Task assignment optimization in SAP Extended Warehouse Management

Akos Monori
2008
DEGREE PROJECT
In Computer Engineering

Programme
Master in Computer Engineering

Reg. number
E3600D

Extent
30 ECTS

Name of student
Ákos Monori

Year-Month-Day
2008.08.20

Supervisor
Pascal Rebreyend

Examiner
Mark Dougherty

Company
SAP Hungary KFT

Supervisor at the company
István Gellért

Title
Task assignment optimization in SAP Extended Warehouse Management

Keywords in English
Warehouse management, task assignment, generalized assignment problem, SAP, EWM
Abstract

Nowadays in the world of mass consumption there is big demand for distribution centers of bigger size. Managing such a center is a very complex and difficult task regarding to the different processes and factors in a usual warehouse when we want to minimize the labor costs. Most of the workers’ working time is spent with traveling between source and destination points which cause deadheading. Even if a worker knows the structure of a warehouse well and because of that he or she can find the shortest path between two points, it is still not guaranteed that there won’t be long traveling time between the locations of two consecutive tasks. We need optimal assignments between tasks and workers.

In the scientific literature Generalized Assignment Problem (GAP) is a well-known problem which deals with the assignment of m workers to n tasks considering several constraints. The primary purpose of my thesis project was to choose a heuristics (genetic algorithm, tabu search or ant colony optimization) to be implemented into SAP Extended Warehouse Management (SAP EWM) by with task assignment will be more effective between tasks and resources.

After system analysis I had to realize that due different constraints and business demands only 1:1 assignments are allowed in SAP EWM. Because of that I had to use a different and simpler approach – instead of the introduced heuristics – which could gain better assignments during the test phase in several cases. In the thesis I described in details what were the most important questions and problems which emerged during the planning of my optimized assignment method.
Abstract .................................................................................................................................................. 3

1. INTRODUCTION .......................................................................................................................... 8

1.1 Foreword ........................................................................................................................................ 8

1.2 The problem to be solved ............................................................................................................... 8

1.3 Generalized Assignment Problem .............................................................................................. 9

1.4 Complexity of the Generalized Assignment Problem .............................................................. 10

2. WAREHOUSE MANAGEMENT SYSTEMS AND SAP EWM ........................................ 12

2.1 Why do we need warehouses? ..................................................................................................... 12

2.2 Warehouse Management Systems ............................................................................................. 14

2.3 What is SAP EWM? .................................................................................................................. 17

2.4 The structure of a warehouse in EWM ...................................................................................... 18

2.4.1 Storage types ........................................................................................................................ 19

2.4.2 Activity areas ........................................................................................................................ 21

2.4.3 Storage sections .................................................................................................................... 22

2.4.4 Storage bins .......................................................................................................................... 23

2.4.5 Quants .................................................................................................................................... 25

2.5 Warehouse tasks and orders ...................................................................................................... 26

2.6 Attributes of a resource .............................................................................................................. 27

3. OPTIMIZATION METHODS FOR SOLVING THE GENERALIZED
ASSIGNMENT PROBLEM .................................................................................................................. 29

3.1 Ant colony optimization ............................................................................................................. 29

3.1.1 Analogy with the real ants ..................................................................................................... 30

3.1.2 Applications of the Ant Colony Optimization ....................................................................... 31

3.1.3 Solving GAP with ACO ........................................................................................................ 31

3.2 Evolutionary algorithms ............................................................................................................ 32

3.2.1 Solution representations in genetic algorithms ................................................................. 33
3.2.2 Operators: mutation and crossover ................................................................. 33
3.2.3 Initial population and candidate selection ................................................... 35
3.2.4 Performance of the GA .................................................................................. 36
3.2.5 Solving GAP with GA ..................................................................................... 37

3.3 Local search methods ....................................................................................... 39
    3.3.1 A local search method - tabu search .......................................................... 40
    3.3.2 Tabu lists .................................................................................................... 42
    3.3.3 The aspiration criteria of the tabu search .................................................... 42

4. THE CURRENT WAY OF TASK/RESOURCE ASSIGNMENT IN SAP EWM
   VERSION 7.0 AND THE NEW PROPOSED METHOD ........................................ 43

4.1 Route optimization or task assignment optimization? ..................................... 43

4.2 Task interleaving .............................................................................................. 44
    4.2.1 The purpose of task interleaving ................................................................. 45
    4.2.2 Practical example of task interleaving ....................................................... 46

4.3 Task assignment in the latest SAP EWM release 7.0 ....................................... 46
    4.3.1 Connecting queues to resources via resource groups .................................. 47
    4.3.2 Task interleaving in SAP EWM 7.0 .............................................................. 49
    4.3.3 Special cases during the assignment process ............................................. 51

4.4 The new model for the problem ..................................................................... 53
    4.4.1 Number of warehouse orders to be assigned to the resource ...................... 53
    4.4.2 A problem – estimating the execution time of a warehouse order ............... 57
    4.4.3 How to compute the processing time of a specific warehouse order? .......... 59
    4.4.4 The decision about the number of warehouse orders .................................. 60
    4.4.5 Organizing the warehouse orders from different queues ......................... 60
    4.4.6 When and how to search for available warehouse orders? ......................... 62
    4.4.7 A second searching approach ................................................................. 64

4.5 Why don’t use GA, ACO or TS for this scenario? ........................................... 66

5. DEVELOPMENT PHASE – IMPLEMENTATION ISSUES AND TECHNICAL
   DETAILS ............................................................................................................. 69

5.1 Where to place the call of the new function into the old logic? ....................... 69

5.2 How to store the data? ..................................................................................... 72

5.3 Are new data types required and can I reuse existing functions? ................. 73
5.4 What is ABAP? ............................................................................................................................... 74

6. TEST PHASE – ANALYZING THE RESULTS .................................................................................. 75

6.1. The process of the testing................................................................................................................ 75

6.2 Test scenarios .................................................................................................................................... 77
   6.2.1 Test case 1................................................................................................................................. 78
   6.2.2 Test case 2................................................................................................................................. 80
   6.2.3 Test case 3................................................................................................................................. 80
   6.2.4 Test case 4................................................................................................................................. 83
   6.2.5 Test case 5................................................................................................................................ 85
   6.2.6 Test case 6................................................................................................................................ 85
   6.2.7 Test case 7................................................................................................................................ 87
   6.2.8 Test case 8................................................................................................................................ 87

6.3 Analysis of the results and conclusion .......................................................................................... 88
   6.3.1 The running time of the algorithm.......................................................................................... 88
   6.3.2 Comparing the new method to the proposed heuristics ......................................................... 89
   6.3.3 The effectiveness of the new algorithm ................................................................................. 89

6.4 Future Work .................................................................................................................................... 93

APPENDIX A ........................................................................................................................................ 96

1. Endnotes........................................................................................................................................... 96

2. List of abbreviations ......................................................................................................................... 98

3. List of figures .................................................................................................................................... 99

4. List of tables .................................................................................................................................... 102

APPENDIX B ........................................................................................................................................ 112

1. Source code samples ......................................................................................................................... 112
Please handle the contents of this thesis work confidentially.
1. Introduction

1.1 Foreword

First of all I would like to thank István Gellért for his help as a supervisor during the making of my thesis project. His professional advices regarding the ABAP environment and the EWM software both from technical and business aspects helped me a lot when I had to plan and implement my own solution for the resource/task assignment. Also I would like to thank all my colleagues who gave me good advices in ABAP development. Last but not least I would like to thank Krisztina, my girlfriend who supported me all the time and sometimes missed my attention due to the project.

1.2 The problem to be solved

The problem sketched in the abstract part is a very complex one since many constraints shall be considered. Empty travels are not welcome because they need energy and time but don’t denote effective work. This can happen frequently in a warehouse if scheduling lines are not well organized, the location of subsequent tasks are far from each other. If a worker can find out the shortest (or almost the shortest) path between two locations then it’s still not the best situation, since maybe there were another location and task but in shorter distance.

The primary goal of my thesis is to find a solution to be implemented into SAP Extended Warehouse Management by with task assignment will be more effective. This
occurring assignment problem is called General Assignment Problem (GAP) in the scientific literature which consists in assigning a set of tasks to a set of agents with minimum cost. Each agent has a limited amount of a single resource and each task must be assigned to one and only one agent, requiring a certain amount of the resource of the agent. Later I will show how the practical example differs from the original GAP and what kind of approach did I use instead of the different popular solutions such as evolutionary algorithms.

1.3 Generalized Assignment Problem

The objective of the GAP is to find a minimum cost assignment matrix for \( m > 0 \) tasks and \( n > 0 \) workers such that each job is assigned to exactly one worker. Such an assignment requires a cost and resource from the worker’s side. This problem occurs in many fields such as communication networks, vehicle routing, machine scheduling. In my thesis project the application field is task / worker assignment in a warehouse. Formally the problem can be described as follows [13]:

\[
I: \text{set of tasks } (i = 1 \ldots n) \\
J: \text{set of agents } (j = 1 \ldots m) \\
a_j = \text{resource capacity of agent } j \\
b_{ij} = \text{resource needed if task } i \text{ is assigned to agent } j \\
c_{ij} = \text{cost of task } i \text{ if assigned to agent } j
\]
The variables are: \( x_{ij} = 1 \), if task \( i \) is assigned to agent \( j \); 0, otherwise.

1. \[
\text{Min } f(x) \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij}
\]

2. \[
\sum_{i=1}^{n} b_{ij} \cdot x_{ij} \leq a_j \quad j = 1, \ldots, m
\]

3. \[
\sum_{j=1}^{m} x_{ij} \leq 1 \quad i = 1, \ldots, m
\]

4. \[
x_{ij} \in \{0,1\} \quad i = 1, \ldots, n \quad j = 1, \ldots, m
\]

The objective is to find an assignment matrix \( x = x_{ij} \), with \( x_{ij} = 1 \) if agent \( i \) performs job \( j \) and 0 otherwise, which minimizes the total costs. The capacity constraints (2) ensure that the total resource requirement of the jobs assigned to each agent do not exceed its capacity. The assignment constraints (3) guarantee that each job is assigned to exactly one worker.

1.4 Complexity of the Generalized Assignment Problem

When we face computational problems we would like to solve them as fast as we can and step over to the next challenge. Since we say as fast as we can then it is reasonable to classify the different problems based on the required computational time. GAP is a well-known \( NP \)-hard (non-deterministic polynomial hard) problem, therefore efficient algorithms needed, especially for large problem instances.
NP is a problem class which consists of most of the combinatorial problems and some of them are related to two areas: problems of complexity \(NP\)-hardness and problems of complexity \(NP\)-completeness. \(NP\)-hard problems are computationally intractable which means we can’t solve them in polynomial time. [15] describes a list of standard problems which contains the GAP, too:

- Integer programming
- Constraint satisfaction problem
- Resource constrained project scheduling problem
- Generalized assignment problem
- Vehicle routing problem
- Set covering problem
- Maximum satisfiability problem
- 2-dimensional packing problem

Many practical problems are \(NP\)-hard in nature, which means an overall, constructive search is unlikely to satisfy our computational demand. The problem with the GAP and the other combinatorial problems is that they have an extremely huge solution space. Considering the GAP again, if we have more and more tasks and workers then the solution space will grow not in a linear but in an exponential way. Therefore that if we want to solve such a problem we have to toss away the idea of using linear algorithms. Heuristic is necessary to find out which are the feasible regions of the solution space, what could be considered as an initial solution. There are many possibilities: we can use genetic algorithms, simulated annealing, swarm particles, different hybrid methods etc.
2. Warehouse management systems and SAP EWM

Since my thesis is based on a warehouse management software I believe it is necessary to consecrate a few paragraphs to this field to understand the business demands and requirements for WMS softwares.

2.1 Why do we need warehouses?

Running a warehouse makes possible to match better the supply with customer demands. This is a dynamic scenario where the demands change quickly and supply has to adapt for them which is always a slower process. Without stockpiled products it would be impossible to support surge demands in seasons such as December when product shipping happens at a very high rate. Shipping can be also quicker, if the warehouse in near to the place of use. The other reason for running a warehouse is price breaking for bulk purchases. If the customer buys high amount of item from a vendor, the unit price may become low enough to exceed the expense of storing with the savings.

Storing and buffering are not the only benefits of a warehouse. Transportation costs can be pushed lower by consolidation which means the amortization of the fixed transportation cost by filling the total carrier capacity. Consolidation of small shipments into a larger one may allow truck scheduling with a fixed and smaller number of warehouse gates.

Nowadays value added services are also common in warehouses by which product configuration happens closer to the customer. Light assembly is a good example for this, and it’s general in the field of PC manufacturing. Generic parts are assembled in
the warehouse which enables the manufacturer to satisfy many types of customer demand from a limited set of generic items.

![Diagram of warehouse roles in the logistic chain](source: McGraw-Hill Education Group, 2001, Frazelle, Edward H. Supply Chain Strategy)

There may be smaller-bigger differences between warehouses in different aspects, but warehouse operations have a fundamental set of activities in common. The following list contains activities found in most warehouses [4]:

---

Högskolan Dalarna  
Röda Vägen 3  
S-781 88 Borlänge  
Tel:+46-23-778800  
Fax:+46-23-778050  
http://www.du.se
• Receiving
• Prepackaging (optional)
• Putaway
• Storage
• Order picking
• Packaging and/or pricing (optional)
• Sortation and/or accumulation
• Packing and shipping

![Common warehouse activities diagram](image)

*Figure 2.1.2 – Common warehouse activities (source: McGraw-Hill Education Group, 2001, Frazelle, Edward H. Supply Chain Strategy)*

### 2.2 Warehouse Management Systems

[9] “The use of computer-based information technology is now the norm in most warehouses, and is essential for the management of large facilities. Even in conventional
warehouses, for example with reach trucks and ground-level picking, significant advantages can be achieved in terms of productivity, speed and accuracy with the benefit of a good warehouse management system (WMS).”

[10] describes the warehouse management system as a tool which is enabled to manage stock levels, keeps track of the stock movements and supplies RFID and barcode systems. This is a very simple description, but contains the minimum functional requirements. Warehouse management systems like SAP EWM are very complex providing lot more high-end features e.g. quality inspection, value added services etc. The major WMS packages have a wide range of functionality that may be turned on or off for particular applications. For example, in electronics, batch traceability of components in kitting operations may be significant, whereas in food manufacturing the control of sell-by dates may be important.

A fundamental functionality is the recording and handling of inbound and outbound documents. This is essential, since it drives elementary financial transactions: receipt drives the paying of bills to suppliers upstream; and shipping drives the sending of invoices downstream to the consignee.

Tracking stock movement and locating stocks are also essential functionalities of a WMS. This is the ability of managing storage location inventories in addition to an inventory of a product. This makes possible to support different warehouse operations (e.g. picking, internal moves etc.) by directing warehouse activities from source to destination locations. With a stock locator system directed put-way and picking are both possible, since it can track down every place where product can be.
A WMS normally interfaces the customer transaction system (e.g. SAP ERP) to synchronize material data, get purchase orders or send feedback of goods dispatch. The WMS is used to control all the operations in the warehouse and issues instructions to subsidiary systems, for example equipment control systems. Thus, a WMS will issue an instruction to an AS/RS control system for a crane to move a specific pallet from a pick-up and deposit station at the end of the aisle to a particular location in the racking. The equipment control system will then direct the crane and provide feedback and diagnostics if the crane cannot fulfill this operation (e.g. owing to mechanical failure).

Figure 2.2 – Typical system architecture (source: McGraw-Hill Education Group, 2001, Frazelle, Edward H. Supply Chain Strategy)
2.3 What is SAP EWM?

[8] In 2002, Cat Logistics, together with SAP and Ford, decided to undertake a massive challenge: design a supply chain management software solution specifically tailored to meet the demands of the service parts or aftermarket businesses. After thousands of man-hours of work, EWM engine was born as addition to the mySAP SCM (Supply Chain Management) footprint. Cat Logistics began implementing the solution for one of its Indianapolis, Ind., Caterpillar parts facilities in January 2006. Over time, Cat Logistics planned to have all of the facilities that it operates on behalf of its internal client, Caterpillar Inc., up and running on the system.

The development of the solution focused on warehouse operations. Warehouse Management Systems (WMS) are brand new under SAP. EWM is the abbreviation of Extended Warehouse Management which has to offer flexible, automated support for processing various goods movements and for managing stocks in warehouses, furthermore it has to support planned and efficient processing of all logistics processes in the customers’ warehouses. EWM maps an entire warehouse complex in detail in the system, down to storage bin level. The user can always determine exactly where a certain material currently is in the warehouse complex.

In addition, in September 2007 Evobus, the bus and coach manufacturer put into operation a new logistics centre of which miniload system interfaces with SAP EWM.
2.4 The structure of a warehouse in EWM

In the following section I would like to introduce the hierarchical structure of the warehouses used in EWM, because it is essential to know if we want to understand the methods and processes later.
In EWM it is possible to manage an entire physical warehouse complex using a single warehouse number. The user can define the individual warehouses (high-rack storage area, picking storage area etc.) as storage types within a warehouse complex, and join them together under the same warehouse number.

### 2.4.1 Storage types

In EWM storage zones and facilities are called *storage types* and they are all defined for warehouse numbers. A storage type is characterized by its warehouse technologies, space required, organizational form, or function and consists of one or more storage bins. The standard system for EWM in default case contains several complete storage types, just like:

- High rack storage area
- Bulk storage area
• Fixed bin storage area
• General storage area
• Pallet storage area

Figure 2.4.1.a – Warehouse of Toys R Us – carton-pick modules to the left, bulk storage to the right side (source of image: http://www2.isye.gatech.edu/~jjb/wh/)
For the end-user it is possible to define the individual warehouse facilities or warehouses that make up the warehouse complex, using the technical, spatial, and organizational characteristics as storage types.

![Different storage types in one warehouse complex](image)

**Figure 2.4.1.b – Different storage types in one warehouse complex**

### 2.4.2 Activity areas

Activity areas are used as logical subdivisions in a warehouse and as logical grouping of storage bins. In these areas the workers execute different task, for example putaway and picking. It can refer to a storage bin, or can concatenate bins from several storage types.
It is possible to assign one or more storage bins manually to the activity areas, or if there is a 1:1 relationship between storage type and activity area, the system generates the assignment.

For the definition of an assigned storage bin in EWM the following attributes are needed:

- Aisle
- Stack
- Level
- Bin subdivision
- Depth
- Information about the storage type
- Any assigned deconsolidation groups

The user can sort these storage bins into any desired sequence, and assign as many activities to them as required.

2.4.3 Storage sections

A storage section is an organizational subdivision of a storage type, which joins together storage bins with similar attributes for the purpose of putaway. The criteria for joining these bins together can be defined in any way, for example, heavy parts, bulky parts, fast-moving items, slow-moving items.
Storage sections can be used as an organizational aid for putting away goods in the warehouse. Defining the organizational aims clearly is necessary for its use. The physical location is the main organizational factor. It is not absolutely mandatory for the user to subdivide a storage type into two or more storage sections. High rack storage areas frequently consist of many storage bins that vary in size. For example, in many such storage areas, the bins in the lower level are larger for especially large and heavy parts, while those in the upper levels are smaller.

2.4.4 Storage bins

Storage bins are different storage spaces that make up a storage type; they are the smallest spatial units in a warehouse. Therefore, the storage bin represents the exact position in the warehouse where products are and/or can be stored and the coordinates of the storage bin tell the user the exact position in the warehouse, where the user can store products.

Each storage type and storage section consists of a selection of storage compartments that in EWM are called storage bins. The following figure shows a high-rack storage area divided into two storage sections. The end-user uses the front storage section for fast-moving items, and the back storage section for slow-moving items.
Since the address of a storage bin is frequently derived from a coordinate system, a storage bin is often referred to as a coordinate. For example, the coordinate 01-02-03 could be a storage bin in aisle 1, stack 2, and level 3. The following additional attributes can be defined for a storage bin:

- Maximum weight
- Total capacity
- Fire-containment section
- Storage bin type (for example, for large or small pallets)

In some putaway strategies, the storage bin type plays an important role during optimization of the automatic search for a storage bin, in connection with the pallet type.
For example, the user can define the putaway strategy in such a way that you put away large industrial pallets into special large bins and smaller pallets into small bins.

2.4.5 Quants

A quant is a stock of a specific product with the same characteristics in one storage bin, resource, and transportation unit. Different batches of a product are handled as different quants. In EWM quants can only be created and deleted using movements and the user can increase the quantity of a quant by adding to the existing stock.

If the user puts a product away into an empty bin, the system generates a quant in the storage bin and when she/he removes the quantity from storage, the system automatically deletes that quant. The quant record includes the following information about the collected products:

- Quant identification
- Number
- Batch number
- Stock category type
- Stock category
- Stock usage
2.5 Warehouse tasks and orders

These documents will be very important later when describing the assignment method in details. Warehouse tasks are documents created with or without reference documents and used to execute goods movements. We differentiate product warehouse tasks (product WT) and handling unit warehouse tasks (HU WT). The former contains all the information required to execute the physical transfer of products into the warehouse, out of the warehouse, or within the warehouse from one storage bin to another storage bin while the latter contains information only for physical transfers for handling units within the warehouse.

![Diagram of warehouse tasks](image.png)

*Figure 2.5 – Different warehouse tasks*

Warehouse orders represent an executable work package that a warehouse employee should perform at a specific time. The warehouse orders consist of warehouse tasks or physical inventory items. If the user wants to move products, the EWM creates warehouse tasks, then groups these tasks together into warehouse orders, and makes them available for processing. If the user performs a physical inventory, (creates physical
inventory items) EWM also groups these physical inventory items together into warehouse orders. Warehouse orders have an attribute, the so-called Latest Starting Date (LSD) which describes when the processing of the WO should be started at the latest time. This is also a very important constraint that should be considered during the assignment process.

2.6 Attributes of a resource

As mentioned before a resource is a warehouse worker and equipment (e.g. a forklift) together. The resources can be typed in EWM by customizing setting based on some attributes such as horizontal and vertical velocity. The first one obviously determines how fast a resource can travel from one location to another. The vertical velocity is called as Z-velocity in EWM since it describes how fast a resource can move its fork to reach higher storage spaces. These attributes both affect traveling time but only the former is considered when the user creates a new resource type in the customizing section of EWM. In some cases Z-velocity has no meaning, consider for example a platform truck.

![Platform truck](image)

*Figure 2.6.1 – Some equipment in warehouses cannot be used for high-level storage spaces*
There is no restriction for a resource how to handle flammable, hazardous or other materials. Handling instruction is added to the material as additional text but there is no setting in the software now which strictly determines for a resource type which kind of material can be transferred and handled by it. Practically hazardous and flammable materials are located in separated areas of the warehouse (storage areas) and EWM is customized in such a way that only a specified resource type can receive tasks from that area. Other resources can travel there, too, but the system won’t assign tasks to them which are connected with dangerous materials.

To log into the SAP RF environment the worker needs an RF-device. With a small handheld terminal he can select from several options e.g. how to assign a task to him, etc.

*Figure 2.6.2 – Handheld RF-device*
3. Optimization methods for solving the generalized assignment problem

In this chapter I would like to describe several approaches for finding optimal solutions. I would like to pin down in the beginning, that the real problem which I had to solve in SAP EWM differs in some aspects to this problem and I had to face many obstacles, too. Finding an optimal solution in theory is one thing, and find a good assignment for a resource in admissible response time (max 2-3 seconds) in a real-time scenario is another thing; I had to make some trade-offs since there are business demands and existing techniques in the current software to which I had to adapt my method. At the same time I believe it’s not certainly a bad thing; if I can gain valuable results and show why things work differently in a business scenario then it is still useful thing. We can’t use always the old methods or their improved versions but that’s why we have to always create new solutions.

3.1 Ant colony optimization

Dorigo and Stützle describe the origin of ant colony optimization in [11]: Ant System (AS) was the first proposed algorithm which could deal well with small traveling salesman problem (TSP) instances, but other approaches mastered it when the instances became larger.
3.1.1 Analogy with the real ants

This probabilistic technique was inspired by the behavior of the ants as they try to find paths between the food and their colony. When ants start searching for food they wander randomly. If they find food, then during the way back to the colony they make pheromone trails so next time they will know where to go back. The pheromone trails in ACO serve as a distributed, numerical information which the ants use to probabilistically construct solutions to the problem being solved and which the ants adapt during the execution of the algorithm to reflect their search experience. If one ant created a pheromone trail the others can follow it also and reinforce it. This pheromone trail will be evaporated after a time; and this is the strength of the method. If there is a short path between the food and the path the ants need less time to march it and they can strengthen the trail more frequently than a very long path. This will result that ants will tend to use the shorter paths.

The role of an ant in this metaheuristic is described in [12]: “An artificial ant in ACO is a stochastic constructive procedure that incrementally builds a solution by adding opportunely defined solution components to a partial solution under construction. Therefore, the ACO metaheuristic can be applied to any combinatorial optimization problem for which constructive heuristic can be defined.”

Ants can be characterized as stochastic construction procedures which build solutions moving on the construction graph G. Ants do not move arbitrarily on G, but rather follow a construction policy which is a function of the problem constraints. In general, ants try to build feasible solutions, but, if necessary, they can generate infeasible solutions [8]. Since we use a full colony of ants instead of only one agent, the method is more robust and the continuous interaction between them makes the searching for good
solutions more easier. This is particularly true for geographically distributed problems, because the differential path length effect exploited by ants in the solution of this class of problems can only arise in presence of a colony of ants. It is also interesting to note that in routing problems ants solve many shortest path problems in parallel (one between each pair of vertices) and a colony of ants must be used for each of these problems.

3.1.2 Applications of the Ant Colony Optimization

Big strength of this method is its capability for adaptation; when the graph is changed then the algorithm will run continuously and adapt to the changes. Because of this ability, current applications of these algorithms fall into the two classes of static and dynamic combinatorial optimization problems. The topology and costs in the case of static problems do not change while the algorithm is running (e.g. Travelling Salesman Problem). In contrast, in dynamic problems the topology and costs can change while solutions are generated (see networking problems).

3.1.3 Solving GAP with ACO

The Generalized Assignment Problem can be represented by a graph in which the set of components contains the set of tasks and workers, that is $C = I \cup J$ and the set of connections fully connect the graph. Each task/worker assignment, which consists of $n$ (number of tasks) couplings (task $i$ - worker $j$) of tasks and workers, corresponds to an ant’s walk on this graph. Such a walk has to take into consideration the constraints 3) and 4) (see chapter 1.3) since these guarantee a valid assignment. The ant creates a coupling by walking iteratively switching from task vertices (from set $I$) to worker vertices (from
set \( J \) without repeating any task vertex (constraint 3) but possibly using the same agent vertex several times (that is, several tasks may be assigned to the same agent).

At each step of the construction process, an ant has to make one of the following two basic decisions: firstly it has to decide which task to assign next and secondly it has to decide to which agent a chosen task should be assigned. Pheromone trail and heuristic information can be associated with both steps. With respect to the first step the pheromone information can be used to learn an appropriate assignment order of the tasks, that is, \( \tau_{ij} \) gives the desirability of assigning task \( j \) directly after task \( i \), while the pheromone information in the second step is associated with the desirability of assigning a task to a specific agent.

### 3.2 Evolutionary algorithms

In [14] Feltl and Raidl introduced an improved version of the hybrid genetic algorithm proposed by Chu and Weasley\(^4\) with alternative initialization heuristics, modified mutation operator and selection scheme. For the largest and most difficult instances the proposed genetic algorithm yielded on average the best results in shortest time. In a recent paper [20] Majumdar and Bhunia proposed an elitist genetic algorithm to solve the generalized assignment problem with imprecise cost(s)/time(s). Their modified approach used in selections and finding the best/worst solutions, furthermore the new mutation and crossover operators and other new features could help them achieve very good results.

The term genetic algorithm was first used by John Holland, whose book *Adaptation in Natural and Artificial Systems* of 1975 was instrumental in creating what is
now a flourishing field of research and application that goes much wider than the original GA. The original motivation was a biological analogy. In the selective breeding of plants or animals, for example, offspring are sought that have certain desirable characteristics chromosomes combine [8].

3.2.1 Solution representations in genetic algorithms

In the case of genetic algorithms, a population of strings is used, and these strings are often referred as chromosomes. These strings are recombined by using simple analogies of genetic crossover and mutation, and the search is driven by the results of evaluating the objective function $f$ for each string in the population. Chromosomes that have higher fitness values are considered as better solutions and these are given more opportunity to breed.

3.2.2 Operators: mutation and crossover

There are two operators which are essential part of the whole process; crossover and mutation. We can apply one- or more-points crossover, the following example uses the one-point version:
Crossover is a matter of replacing some of the genes in one parent by corresponding genes of the other. Given the parents P1 and P2, with crossover point 3 (indicated by X), the offspring will be the pair 01 and 02. The second operator is mutation which alters the allele values in randomly selected genes.
3.2.3 Initial population and candidate selection

After describing the crossover and mutation operators, the following major questions to consider are the size of the population, and the method by which the individuals are chosen. The size of the population is always of a trade-off between efficiency and effectiveness. A too small a population would not allow sufficient room for exploring the search space effectively, while too large a population would so impair the efficiency of the method that no solution could be expected in a reasonable amount of time.

But which entities should be selected for mutation and crossover? We can assume that if we use promising solutions then the entities in the next generation will be also appropriate. The selection is related to the fitness values of the candidates, and the original scheme for its implementation is commonly known as the roulette-wheel method. It uses a probability distribution for selection in which the selection probability of a given string is proportional to its fitness.
3.2.4 Performance of the GA

Genetic algorithms are stochastic search methods that could run forever but in practice, a stopping criterion is necessary. Such a criterion can be setting a limit on the number of fitness evaluations or the computer clock time, or to track the population’s diversity and stop the algorithm when this diversity value falls below a predefined threshold. The meaning of diversity in the latter case is not always obvious, and it could relate either to the genotype or the phenotype, or even, conceivably, to the fitnesses, but the most common way to measure it is by genotype statistics.

One of the keys to good performance is to maintain the diversity of the population as long as possible. The effect of selection is to reduce diversity, and some methods can

Figure 3.2.3 – Roulette wheel selection
reduce diversity very quickly. This can be mitigated by having larger populations, or by having greater mutation rates, but there are also other techniques that are often employed.

A popular approach, commonly linked with steady-state or incremental GAs, is to use a ‘no-duplicates’ policy [17]. This means that the offspring are not allowed into the population if they are merely clones of existing individuals. The downside, of course, is the need to compare each current individual with the new candidate, which adds to the computational effort needed.

3.2.5 Solving GAP with GA

The whole process begins with the initial solutions. It is our decision if we generate random strings or use some heuristic to generate only valid initial solutions which will not require further validations. If we use the first way then we have to validate the solutions already at the beginning. Of course validation is always necessary later because when we use the crossover and mutation operators there will be many invalid assignments which should be dropped or corrected.

A string may look like this if we have 8 workers:

<table>
<thead>
<tr>
<th>Task10</th>
<th>Task34</th>
<th>Task11</th>
<th>Task02</th>
<th>Task03</th>
<th>Task30</th>
<th>Task13</th>
<th>Task22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worker1</td>
<td>Worker2</td>
<td>Worker3</td>
<td>Worker4</td>
<td>Worker5</td>
<td>Worker6</td>
<td>Worker7</td>
<td>Worker8</td>
</tr>
</tbody>
</table>

*Figure 3.2.5–An example solution*
In this example one task is assigned to only one worker. If the constraints are satisfied, too, then we can consider this string as a valid one. At this point we still don’t know if it is an optimal one or not.

Another important question is the validation of the solutions. Firstly we have to check that one task is assigned to only one resource. If there are multiple assignments then the solution is invalid. Secondly, if there are no multiple assignments then the algorithm has to check if the constraints are satisfied or not. If the worker has less capacity than it’s required for the execution of the assigned task then the assignment is invalid again. We have to decide if we want to correct an invalid solution or drop it and use two parent strings from the current generation to create a new solution.

We have to consider well our operators: shall we use multiple cross-over or just single, and how the mutation should work? Is it reasonable to change only one assignment in the string or change more? There are many questions regarding the implementation which can’t be answered at the very beginning. This chapter doesn’t contain the best algorithm for the GAP but raises the most important questions if someone would use evolutionary algorithm to solve this problem. We have to perform several tests and change the operators and other parameters if necessary.

Computing the fitness value is easier since we know the constraints, the costs, the capacities and the assignments. If the assignments are economical then we should award it by adding higher fitness values while assignments with too much free capacity should be rewarded with less fitness points.

The last step is the selection. We can use the mentioned roulette-wheel selection but there are other strategies, too. For example we can sort the assignments by fitness
values and choose the best n solutions. Whatever we choose we can’t forget that the diversity of the solutions is important.

3.3 Local search methods

Local search methods are improvement search techniques for computationally hard combinatorial problems. In order to derive a local search method, it is necessary to define a neighborhood structure, that is a function that associates a set of solutions \( N(x) \) with each solution \( x \). The neighborhood is usually obtained by specific modifications on \( x \), called a \textit{move}. The local search starts with an initial solution and searches the neighborhood defined before for one solution with some characteristics, as for example the one with lower cost. Then this neighbor solution replaces the current solution if it verifies some properties that depend on the acceptance strategy as, for example, if it has a lower cost than the current solution. The search continues until a stopping criterion is verified. The algorithm returns the best solution found with respect to the cost function.

The search space consists of only complete solutions; this is a big difference between local search algorithms and construction algorithms [16]. The latter works on partial solutions and tries to extend them in the best way. Regarding the performance of local search the neighborhood structure is a very important factor; this defines a set of solutions which can be reached from the current point in one single step.

The most basic version of the local search method is known as \textit{iterative improvement} which searches in the neighborhood for an improved solution. If the new solution is better than the current one then the algorithm replaces it. The algorithm will run until there is no better solution in the neighborhood than the current one which means the algorithm reached a local optimum. Local optimum is an optimal solution within a
neighboring set of solutions. In contrast to that global optimum is an optimal solution within the whole search space. The locality of the optimum is dependent on the neighborhood structure as defined by the metaheuristic that is used for optimizing the solution.

3.3.1. A local search method - tabu search

In [18] Diaz and Fernandez proposed a flexible heuristic to solve the GAP. Due to its applied recent and medium-term memory for dynamic penalty weight adjusting they could gain very good quality solutions in feasible computational time. In [19] a metaheuristic was introduced to solve a variant of GAP, the MRGAP (Multi-Resource Generalized Assignment Problem) with an algorithm based on tabu search. It uses a very large-scale neighborhood search, where the resulting neighborhood is searched by improvement graph and automatically adjusts search parameters, to maintain a balance between visits to feasible and infeasible regions. Considering the results their solution proved to be efficient on very hard problem instances.

In [21] the birth of the tabu search heuristics is described as follows: “The steepest descent\textsuperscript{3} approach has been extended by Glover to tabu search, which basically boils down to a steepest descent/ mildest ascent approach. As in the steepest descent approach, all solutions in the neighborhood of the current solution are evaluated and the solution with the smallest project makespan is chosen. However, tabu search is continuing the search even if the new solution does not improve the previous one. “
The fundamental principle of this method is that it can overcome local optimum by allowing non-improving moves. The algorithm can’t go back to previously visited solutions because the so-called tabu lists don’t allow this. Gendreau describes the tabu lists in detail in [8]:

“Tabus are stored in a short-term memory of the search (the tabu list) and usually only a fixed and fairly limited quantity of information is recorded. In any given context, there are several possibilities regarding the specific information that is recorded. One could record complete solutions, but this requires a lot of storage and makes it expensive to check whether a potential move is tabu or not; it is therefore seldom used. The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations … others are based on key characteristics of the solutions themselves or of the moves.”

The tabu search is a powerful algorithmic approach that has been applied with great success to many difficult combinatorial problems. The search space of an LS or TS heuristic is simply the space of all possible solutions that can be considered (visited) during the search. A particularly nice feature of TS is that, like all approaches based on local search; it can quite easily handle the complicating constraints that are typically found in real-life applications. In fact, basic TS can be seen as simply the combination of LS with short-term memories. It follows that the two first basic elements of any TS heuristic are the definition of its search space and its neighborhood structure. Choosing a search space and a neighborhood structure is by far the most critical step in the design of any TS heuristic. It is at this step that one must make the best use of the understanding and knowledge he/ she has of the problem at hand.
3.3.2 Tabu lists

Tabus are stored in a short-term memory of the search (the tabu list) and usually only a fixed and fairly limited quantity of information is recorded. In any given context, there are several possibilities regarding the specific information that is recorded. One could record complete solutions, but this requires a lot of storage and makes it expensive to check whether a potential move is tabu or not; it is therefore seldom used. The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations; others are based on key characteristics of the solutions themselves or of the moves.

3.3.3 The aspiration criteria of the tabu search

Meanwhile tabus mean the basis of this heuristics, they are sometimes too powerful which means they may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process. That is the reason why it necessary sometimes to cancel a tabu. To avoid this situation we use the so-called aspiration criteria. The simplest and most commonly used aspiration criterion consists in allowing a move, even if it is tabu, if it results in a solution with an objective value better than that of the current best-known solution (since the new solution has obviously not been previously visited). Much more complicated aspiration criteria have been proposed and successfully implemented, but they are rarely used. The key rule in this respect is that if cycling cannot occur, tabus can be disregarded.
4. The current way of task/resource assignment in SAP EWM version 7.0 and the new proposed method

In this chapter I describe the method and principles used in EWM 7.0 to give a better picture what I used for starting point. In the second part I describe my own solution and try to find answers for the emerging questions.

4.1 Route optimization or task assignment optimization?

It is obvious that the goal is to avoid the non-necessary traveling within the warehouse since it doesn’t add value but requires time. At the very beginning of my thesis it was a big question what should be optimized, the question “How?” had secondary priority. In my situation the system to which I wanted to add plus functionality had already existed just like business demands from the partners. On the grounds of this there was no need for route optimization for warehouses, but this is not true for every situation. Totally automated warehouses already exist without human resources, only controlled vehicles travel within the warehouse and perform different tasks. This is a different scenario than a warehouse with human and other resources.

In a totally automated environment it is necessary to generate optimal paths between two points and send them to the RF-enabled robot vehicles. In addition, the environment may change, maybe after sending the route somewhere an obstacle emerges in the path which must be avoided somehow, so we need continuous feedback and
updates, too. In my case the software is used for ordinary warehouses with human and non-human (e.g. a forklift) resources.

Visualizing complete routes on RF-devices is currently not possible due to the limitations of the display, so even if we generate optimal paths for human resources, it is still problematic to use. There are other difficulties with pick-path optimization mentioned in [6], too. For optimization the algorithm must know the structure of the warehouse and the distances between all pairs of storage locations. This is a huge amount of data, and the content of a warehouse is always changing, so the data must be always updated which requires extra computational efforts. The last and most important counter-reason against complete optimal route generation is the lack of business demand. After a few weeks of practicing and training the worker can find his or her way about the warehouse complex and doesn’t need to be told where to turn at the next corner. In SAP EWM we don’t tell the worker which way to go, only where to go. Irrespectively of this there is distance calculation in the system because it is necessary when sorting the different warehouse orders (see later chapters for more details).

Of course the lack of need for pick-path optimization doesn’t mean that here is no more possibility to make the performance better. Since the previous chapters of the thesis concentrated on task assignment then evidentially my extension will focus on this area.

4.2 Task interleaving

In the beginning I would like to introduce a principle which is used in warehouses and implementing this can make the overall performance better. This is the task interleaving (TI).
4.2.1 The purpose of task interleaving

Task interleaving is not a very new method in warehousing, since the need for creating round trip payloads for forklift trucks is more than 20 years old now. Referring to [5] task interleaving is the term used in describing a warehouse management system to mix tasks to reduce travel time; it is a more advanced implementation of directed activity. Task interleaving provides a continuous flow of work to resources, replacing one-way work assignments and travel with bi-directional work assignments. It alleviates the problem of deadheading, which occurs when a forklift, for example, travels in one direction to perform a task and returns empty. Used primarily in full-pallet-load operations, task interleaving directs a lift truck operator to put away a pallet on his/her way to the next pick. In large warehouses this can greatly reduce travel time, not only increasing productivity, but also reducing wear on the lift trucks and saving on energy costs by reducing lift truck fuel consumption. It is a tremendously powerful concept that has been the catalyst for many companies to move from locator level warehouse software systems to higher level WMS packages.
4.2.2 Practical example of task interleaving

Figure 4.2.2 – Example scenario of task interleaving

In this situation the forklift driver is sent to a storage type where he scans a handling unit then he is assigned to a picking task. He loads the handling unit onto his forklift then travels to the destination, being the goods issue area. After placing the handling unit in the goods issue area, the driver confirms execution of the task. Now, he is assigned to a putaway task, the source of which is at the goods receipt area (it’s near by the goods issue area). The driver scans the putaway handling unit, loads it on to his forklift and moves it to the destination bin.

4.3 Task assignment in the latest SAP EWM release 7.0

Briefly I’d like to introduce the current model of resource/task assignment in the latest version of EWM. My new solution will be compared to this one.
4.3.1 Connecting queues to resources via resource groups

First of all I’d like to introduce the queues: they are logical files to which warehouse orders (WOs) for processing are assigned. Queues define movements by which work in the warehouse is managed and assigned. Generally we can talk about inbound, internal and outbound queues, but this a very flexible concept. Since different companies may have different warehouses therefore different strategies and structuring needed for every location. The management and distribution of work is based on the assignment of warehouse tasks and resources to the different queues.

In chapter 1.6 I’ve already mentioned warehouse tasks and orders. Warehouse tasks are atomic; they represent movements from a source location from a destination location within the warehouse. Warehouse orders consist of several tasks based on filter criteria e.g. weight, number of pieces etc. If an outbound request arrives for ten different products maybe the new tasks will be assigned to the same order and finally processed by the same resource, but it depends on the filtering conditions. A resource may execute only those orders that belong to one of its allowed queues (but there may be situations when orders are assigned manually by the warehouse supervisor). This means a queue to which the resource or resource group is assigned, and in which the resource is authorized to work. In addition, a resource working within a particular operating environment may execute only those WOs that belong to a queue in the same operating environment.

It’s possible to assign a sequence of queues to a resource group and this setting is strongly customizing dependent. Since warehouses are different it’s not good to hardcode everything but let the user (or the consultant) scale it for the needs. This means the overall performance of the software depends not only on the code but on the customizing
settings, too. This is a totally different concept than I’ve described in the previous chapters regarding GA, ACO and TS.

Now assigning orders can happen basically in two different ways. In the first the warehouse supervisor assigns manually an order to the resource. In this case the supervisor’s decision will determine if the assignment is optimal or not, the algorithm efficiency doesn’t matter. For my thesis this is not relevant scenario.

The second way is to leave the system decide what to assign and how (this is the so-called system-guided mode). In this case we have one resource and we have to find an
appropriate warehouse order for it (1:1 assignment). The system loops through the first queue (which is assigned to the resource group) and selects the first open (not locked by other resources) WO and assigns it to the worker. If there is no open WO in the queue then the algorithm will check the next queue in the sequence. If there is no more WO in the queue then the resource has to do nothing.

4.3.2 Task interleaving in SAP EWM 7.0

The system-guided mode was extended with the optional task interleaving functionality to create better assignments. This means practically that all assigned queues have a new attribute which describes their type: there are *inbound*, *internal* and *outbound* types but it’s possible to create new types if the customer needs it. This is not an obligatory option; this can be enabled or disabled from a corresponding customizing section.
Let’s consider a practical example to understand how it is working in the real life with the following variables:

**R** - A forklift truck and its driver

**Q1** (inbound) = \{WO12, WO14; WO16; WO34; WO36; WO70\}

**Q2** (internal) = \{WO13, WO43; WO45\}
**Q3** (outbound) = \{WO17, WO18; WO27; WO40\}

R and the queues are assigned through a resource group. Q1 is the first in the queue sequence therefore that the system will search for unlocked WOs from that queue. When TI is activated the system will check Q2 right after R has finished its WO execution from Q1. The reason for doing that is simple: it requires less time for R to travel to the adjacent internal area than travel back much more with empty forks. This queue switching will generate circular travels, since after Q3 the Q1 queue will be inspected again.

Without TI the resource R will perform tasks only from the inbound queue if open orders still exist. If there are no more open warehouse orders then it will continue with the second queue. Getting tasks from the same queue will result that the worker will always travel back to the same area with empty forks instead of traveling to a nearby location.

### 4.3.3 Special cases during the assignment process

There are some special cases to be considered, too: if there are more WOs in the same queue with the same LSD value (see chapter 2.5) then a distance-calculation function will be used to decide which location is the nearest to the current location of the resource.

The second special case when there are two WOs with small difference between their LSD values (few seconds).
In this case the system will give WO1 to the resource even if it requires much more time to travel into the direction of WO1 than WO2. This is not definitely a good behavior, it should be checked how long would be the traveling times to execute WO1 and WO2 then compare these times to the difference of the LSD values. With this small comparison the system could make better assignments. Since this option is missing from the current release, I can add it to my solution.

By all accounts the whole assignment process in the latest release depends mainly on the customizing settings (grouping of resources, queues assigned to the groups etc.) which are set up by consultants and professionals. Because of that no algorithms are used for assignment which are computationally expensive (long running time), but a bad setting can corrupt the overall performance. In the following I would like to talk about the multiple and single task assignment. Both concepts have pros and cons and I will reason my decision.
4.4 The new model for the problem

After introducing the standard assignment method it is time to think about a new approach which may be better from some aspects. Again, I would like to clarify that my thesis is based on the SAP EWM software which means I had to accept its architecture, the business logic behind it and its other constraints. My original plan was implementing one of the most promising heuristics (GA, ACO or TS) into EWM then test if it makes better assignments. After a deeper analysis (considering software architecture and business logic) I came to the conclusion that the original plan wouldn’t work and I had to think about a new method considering the software environment. Now I’ll introduce my ideas and answers for important questions regarding this issue from the beginning.

4.4.1 Number of warehouse orders to be assigned to the resource

GAP is about multiple task assignment so the first question is the maximum number of warehouse orders to be assigned to a resource. Is there a rational limit then what is that? This question leads us immediately to the locking problem in EWM. If we assign too much order to a resource, these will be all locked by a flag in the database for the other resources. This means other workers won’t be assigned to those orders since they are not available for them. But why is it a problem if a resource can solve tasks by itself reliably?

In a warehouse the following situation can occur: if several resources are in the same area or in the same aisle then it can cause collision and the resource has to wait without working. During waiting the time is ticking but there is no effective work and the latest starting date of the other orders will be the same. This means I would have to make
possible for the resource to unlock its assigned task and make it available again in the
database for the other resources, too. There should be a possibility to skip an order, or
change the execution sequence of the orders, if possible. Of course changing the
optimized sequence is not always possible, but we would have to offer this possibility,
too. This means if I implement an assignment heuristic then it is not enough, other
extensions are needed in the system.

The question regarding the number of assigned orders is not answered yet; is it
sure that allowing multiple assignment will be always acceptable? Consider the following
example; what happens if we use an own order list for the resource which can contain n
warehouse orders in a sequence. The first problem is the reloading: when should a
background process add new warehouse orders to the list? If we want to add immediately
a new one when an order is executed (so the number of unprocessed orders in the list will
be n-1) then the problem will be similar to the second example detailed later. This would
work in a FIFO (First In First Out) way normally. If we want to push new orders into the
list when only one order remained then the running time of the background search will
take longer time, because n-1 orders will be needed in a sequence where the LSD value
determines the order.

Another problem with this multiple-order list is its inflexibility. If suddenly a
truck arrives to the warehouse with plenty of products and the resources are assigned to
long order sequences then they won’t be able to ignore their lists and the truck with the
products will have to wait for the unloading process in the warehouse door. This is a
possible business scenario which should be taken into account when planning the new
method. Unnecessary waiting without working is not welcomed.
Now forget the idea of the multiple order lists for a while and think about single assignments (1:1). In the second example we use a one-element list; a single resource/WO assignment. If the resource begins the execution of the order then a background search has to be started immediately for the next order. This is similar to the method mentioned at the first example: if a WO is executed then list reloading immediately happens. Now this idea seems to be similar as I mentioned in an earlier chapter when I described the standard assignment. But it can be more effective! Don’t forget the special case mentioned before which gives a good opportunity for optimization, and searching for the next available order still can be optimized.
Until now in this chapter I described two examples for 1:n and 1:1 assignment, but in the GAP chapter the problem was an m:n assignment. What is the reason for the difference? The main reason is the software architecture itself. After its analysis I’ve realized that the software doesn’t assign the warehouse orders centrally at the same time for every resource. This means when a worker logs into the system and asks for a job then the system will assign a warehouse order from the related queues. This is one thread, if more than one worker enter the system at the same time and ask for a next job then separate threads will run for the assignments.

![Figure 4.4.1.b– Assignments are performed separately](image)

This is the basic behavior which can’t be changed for the sake of the thesis project. This means practically that the problem is simpler because from this point I had
to think about only 1:1 or 1:n assignment. In the following I’ll clarify the question of the number of warehouse orders.

![Diagram of warehouse orders and resources](image)

*Figure 4.4.1.c– Which assignment is better from business point of view?*

### 4.4.2 A problem – estimating the execution time of a warehouse order

If multiple-task assignment (1 worker: n tasks) would be accepted from business point of view then we need the execution time of an order to find a consequent one for it.

![Diagram of warehouse order execution timeline](image)

*Figure 4.4.2– Which WO should be the next candidate?*
Study this figure to understand the problem with multiple-task assignment. WO 01 is the first assigned warehouse order in the sequence and we can see that the worker can start its execution before its LSD. As a first step it is a good assignment, the worker isn’t in late. The problem with the second order is that its LSD value is before the end of the execution time of the first order. It doesn’t make sense to assign WO02 to the same resource because when it will finish its first order it will be already in late. That would be an infeasible solution so we have to find a more suitable warehouse order. If we don’t know the execution time then we can’t predict if the next order to be assigned could be executed without delay or not. Warehouse order 03 seems to be a good solution in this case.

Therefore the required processing time of an order is a necessary parameter during the optimizing process. Obviously every task requires a processing time and there is a traveling time between two adjacent tasks (every warehouse order contains 1…n tasks). This can be formulated as it follows:

\[ T_{req} = \sum_{i=1}^{n} p_i + t_i \]

where \( T_{req} \) is the total length of the warehouse order processing, \( p_i \) is the required time to perform the \( i \)th task and \( t_i \) is the traveling time between the \( i \)th and \( i+1 \)th task. This formula reflects the ideal case; the order of the tasks is given, the resource travels from one to one then it will finish the processing at the end. Traveling distances can be calculated easily with an already implemented algorithm which uses Manhattan distance\(^5\) metrics. There is also a possibility to define in customizing settings...
the processing time of different tasks. With all of these it would be easy to compute $T_{req}$ as I’ve mentioned in an ideal case. The problem is that these values are not always set.

I’ve mentioned earlier that there are sudden accidents when the worker may change the order of task processing which may lead to additional travels (e.g. the fork-lift truck can break down, there is collision in an aisle, a high amount of material arrives to a door etc.). At this point the originally calculated $T_{req}$ is not reflecting the real circumstances anymore; even it seemed to be optimal earlier.

4.4.3 How to compute the processing time of a specific warehouse order?

We have to notice, that in most of the cases we rely on customizing settings which indicate how much time it takes to perform a specific task (but it is not always customized). Since the algorithm generates the WO sequence before executing the tasks it doesn’t know and not able to estimate these values.

First we can use the original formula, find an optimal sequence (with a predefined length) and make the software adaptive by handling different sudden situations. Pros and cons are described in the previous subchapters.

Secondly, we can expect that something will happen and therefore that more traveling time will be needed but we don’t know, how much. Expecting the worst case scenario the algorithm chooses the sequence with the biggest processing time. This will require definitely more computational resources than the first version. This means, if we have 10 tasks in an order, there are 10! possible routes. Routing problems like this can be
solved with Bellman-Ford\textsuperscript{6} or Dijkstra’s\textsuperscript{7} graph search algorithms (since there are no negative edges in this example, Dijkstra’s algorithm would be a better choice). Of course in this situation we search for the longest, not for the shortest path. But what will this kind of approach guarantee? If suddenly the original order of the tasks is changed, the changed total execution time won’t bother us since the algorithm works with the worst case scenario execution time. At the same time this kind of approach is not a lucky choice: anomalies may occur but not always so calculating the longest path every time will waste CPU power and memory space. It’s more feasible to calculate the $T_{req}$ based on the original WT sequence and make the program flexible to handle such situations.

4.4.4 The decision about the number of warehouse orders

After several discussions with my colleagues we have agreed that the best option is still the 1:1 assignment. M:n doesn’t work due to the system architecture and 1:n is also problematic due to the execution time processing and it is not flexible enough (remember the collision examples or immediate urgent jobs). This problem doesn’t remind anymore to the original GAP problem which I wanted to solve in EWM. But this doesn’t mean that there is nothing to do, as I have written before there is a weakness in the original method, and there is a possibility to use the queues in a different way!

4.4.5 Organizing the warehouse orders from different queues

In the original EWM model the resources were assigned to resource groups which were also assigned to different queues. Since it’s a rational and necessary organization of
resources and orders I’ve started the planning from the same idea but in a different way. If there are more than one queues assigned to the resource group then the system should consider them as one global queue containing several tasks. Priorities don’t matter. The standard solution uses separated and prioritized queues for execution and both ways have pros and cons.

For example, if a truck arrives into the goods receipt zone then probably there will be a lot of items to be picked and taken to final destination bins. In this situation the supervisor can tell the worker to pick orders only from the inbound queue to disencumber that area. Against this my solution uses one global queue which is made of the several separate queues assigned to the resource group. The logic is that the emphasis is on giving the nearest warehouse order to a worker. If the worker executed the tasks of an order which was originally in an inbound queue and the nearest order is in the internal queue then it will get that. It doesn’t matter which was the original queue. If I want to be able to handle situations described previously then I can still keep the queue numbering but considered only in exceptional situations.

Already at the beginning it’s obvious that I can’t rely only on optimization methods and I have to adapt my solution to the existing one. Since the thesis is about a new assignment method in SAP EWM I have to integrate it by accepting some rules and constraints which may give less possibility for different optimizing heuristics. I had to think about response time, too. Workers walk in the warehouse with small RF-devices (radio-frequency enabled tools) by which they can log on, ask for tasks and accept them, and there are strict requirements regarding the response time of the system. If every worker shall wait for long minutes to get his own warehouse order list then it is disturbing and totally unacceptable for the customers. Already at this point we can see
why it is usual for WMSs to rely on mainly customizing settings and using general algorithms for solving problems: customer needs may be different and a warehouse is a dynamical environment which means always there is a possibility for accidental events and a rigid system will be hardly able to respond to the sudden challenges.

4.4.6 When and how to search for available warehouse orders?

When the resource begins the execution of a warehouse order the system automatically starts a background process which will call a search mechanism based on the LSD and the required traveling time for the first task of the warehouse order. When the appropriate order is found then it is assigned to the resource. This sounds very simple and logical but there are some technical issues to be considered. For example what happens if the resource requires the new WO but it’s still not found because of the longer running time of the optimized algorithm? In this case the system runs the standard solution, which will find the next WO faster but in a less optimal way.

Queues are considered as a global one, all open warehouse orders could be chosen as a subsequent one. All the warehouse orders have the LSD property so the WOs are put into a list in ascending order ordered by the LSD (this is not the own order list of the worker, it is only for temporary purposes). It would be the easiest way to select the first WO with the lowest LSD value but it’s still not enough. If there are many WOs with the same or almost the same LSD values then the algorithm. Let’s see the main steps of the new algorithm in a pseudo code after the resource requests for a warehouse order:
Collect the queues assigned to the resource group of the resource.

Unite the WOs in the different queues into one global list.

Sort the order items by LSD.

Keep only the first n warehouse order items.

Get the LSD timestamp of the first WO from the global list.

Loop through every WO

  Calculate Manhattan distance from the current position to the WO start location.
  Calculate traveling time based on the Manhattan distance
  Calculate penalty based on the difference of LSD of the current and the first WO
  Calculate a fitness value based on the penalty and distance values

End of the loop

Sort the list again by the fitness value.

Loop through every WO

  Try to assign the WO to the resource.
  If it was successful, then exit from the loop.

End of the loop

One remark for the fourth step: we do not consider all open warehouse tasks, since it’s not necessary. In the current version it is hardcoded that the system will check only the first 50 warehouse orders from the queues. My solution allows higher numbers.
But how is the penalty calculated and what is its role? Penalty is the difference of the timestamp of the first warehouse order (the lowest value) and the current order in the loop. For example if WO01 is 2008:08:10 15:20:00 and WO10 is 2008:08:10 15:21:00 then the difference will be 60 seconds. As we loop through the list the penalties will be higher and higher. When the distance is computed then we can compute the traveling time which is measured in seconds. The final fitness value is the sum of the penalty and distance values. The optimal order has the lowest fitness values at the end. Giving more and more penalty points to the later orders we can avoid choosing a warehouse order in the far future.

With this solution the traveling distance is used in a different way since until now it was considered as a sorting parameter if the subsequent orders had the same LSD value.

### 4.4.7 A second searching approach

This solution was tossed finally but it’s important to talk about that. Basically there is no big difference between the algorithms:

- Collect the queues assigned to the resource group of the resource.
- Unite the WOs in the different queues into one global list.
- Sort the order items by LSD.
Keep only the first n warehouse order items.

Get the LSD timestamp of the current time.

Loop through every WO

Calculate the difference of the current timestamp and the LSD of the order

Calculate Manhattan distance from the last position of the current WO to the next WO start location.

Calculate traveling time based on the Manhattan distance

Subtract the traveling time from the calculated difference

Use the new value as the fitness value

End of the loop

Sort the list again by the fitness value.

Loop through every WO

Try to assign the WO to the resource.

If it was successful, then exit from the loop.

End of the loop

The goal of this algorithm was to find the warehouse orders which are the most urgent. First it would be reasonable to think that the most urgent is the one with the lowest latest starting date, but processing the first order with the lowest LSD value is not surely the best option since there may be other warehouse orders with highest LSD values but much longer traveling times.
After several discussions I’ve tossed the idea because the most important problem is to minimize the traveling. If the previous situation occurs we should just get the lowest LSD order and let the other workers to be assigned to the other ones. The problem is that if the ulterior orders require longer traveling then they will be prioritized instead of the orders which need less traveling in the warehouse. This is against the principle of choosing an order with the lowest or almost the lowest LSD but with less traveling time.

4.5 Why don’t use GA, ACO or TS for this scenario?

My original idea when I started the thesis was to apply one of the proposed heuristics for GAP to EWM and perform several tests. But during analysing the system architecture and the business logic I found out that this can't work as I had originally planned. As I've described earlier the GAP is about multiple assignment (m workers : n tasks) with different constraints but this problem doesn't occur in EWM in the same way. Why is that? The main reason for that is the business logic and software architecture behind EWM. Again, if a resource logs into the SAP EWM system by using his handheld RF-device then the system assigns and asks for a next job then the system assigns a warehouse order to him (and this warehouse order is not assigned at this time to other workers, too). This happens in one process. If there are 10 workers in the warehouse, and they log into the system at the same time then 10 different threads will run for the specific resources resultin a successfull/unsuccesfull assignment. This means that assignment is not centralized the logic considers 1:1 assignment. This is a logic which shouldn't be changed so I had to create a better assignment method which fits this logic.
After all it would be still interesting to think about the proposed methods if the application would support the multiple assignments. First of all, when the GA begins its running we cannot predict if it will give us good, optimal, or bad result. The second question is when it will give a good result if it will be good anyway. If this thesis focuses on a timetable optimization or the vehicle routing problem then the situation would be different, since it’s enough to generate an optimal solution – even if it would require long computational time – then we could use it later. But this scenario is real-time; the system has to give an optimal solution in less than 2 seconds. Can a GA guarantee always optimal solutions in such a small time interval if we have hundreds of warehouse orders? The other problem is that we need only one order, not a whole sequence, so it doesn’t make any sense to create strings which contain only one order and perform crossover on them. For this situation the genetic algorithm is not useful.
The other thing is that the effectiveness if the genetic algorithm depends on the operators, initial solutions and selections, too. Consider that in one warehouse the maximum number of warehouse orders in a queue is 10 and in a different one it is 5. This means the strings will be different considering the length so different settings are required to run the algorithm properly. We can’t expect from a WMS consultant to know the theory of genetic algorithms and customize how big should be the initial population, where to put the crossover point etc. From business point of view this approach is not a lucky one.

The previous problems also occur at tabu search and the ant colony optimization at the beginning. The solution is only one warehouse order, not a warehouse order sequence. This makes the mentioned heuristics unnecessary.

For multiple assignments in EWM I’d use the ACO of the introduced heuristics because of the following reason: this method constructs the final solution partially from step by step meanwhile a genetic algorithm would generate full length warehouse order sequences. Tabu search does the same and that is the reason why I wouldn’t use GA and TS for assignment in EWM.
5. Development phase – implementation issues and technical details

This chapter gives some insight into the implementation of the new solution. The most important questions were:

- Where to place the call of the new function into the old logic?
- How to store data?
- Are new data types required and can I reuse existing functions?

I’d like to briefly show the answers without being enmeshed into too much technical detail since it’s not a SAP software design document. Also I will introduce the ABAP programming language briefly since almost everything is developed in ABAP at SAP (just like EWM). For ABAP source code samples check the appendix.

5.1 Where to place the call of the new function into the old logic?

The positive side is that the new solution fits tightly the old logic so I didn’t have to write everything from the scratch but concentrate only the implementation of the algorithm. Hereby I’d like to show the pseudo code of the order assignment process which is evidently only a small piece of the whole process but it’s enough to block in the original standard logic.
Check manually assigned warehouse orders

If there are no manually assigned orders

    Try to find a suitable one from the queues

    If there is a suitable order

        Assign it to the resource and quit

    Else

        Quit, there is no suitable warehouse order

    Endif

Else

    Assign it to the resource and quit
Endif

Now check the new modified old logic with the inserted new codes. The indented lines are together the new logic.
If there are no manually assigned orders

Check if the background process found something earlier

If it found a suitable order

Assign it to the resource

Delete the warehouse number from the corresponding database table entry

Begin a new background search process

Assign the order to the worker and quit

Else

Ignore the result of the background process by setting a flag in the corresponding database table entry

Try to find a suitable one from the queues

If there is a suitable order

Assign it to the resource

Start a background search process with the new method

Assign the order to the worker and quit

Else

Quit, there is no suitable warehouse order

Endif

Endif
Else

    Give the manually assigned order to the worker

    Start a background search process with the new method then quit

Endif

5.2 How to store the data?

The only data we need to store is the resource/warehouse order assignment, the number of the warehouse and a flag. Warehouse number is necessary, since EWM can manage several warehouses at the same time, and there may be resources with the same name in different warehouses. One flag is necessary to indicate, if the result of the currently running background function is still required or not. Here is an example of the database table ZRSRC_WO with three assignment entries.

<table>
<thead>
<tr>
<th>WAREHOUSE NUMBER</th>
<th>RESOURCE NAME</th>
<th>WAREHOUSE ORDER NUMBER</th>
<th>DISABLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>WH01</td>
<td>SMITH</td>
<td>W02345</td>
<td></td>
</tr>
<tr>
<td>WH01</td>
<td>ROBERTS</td>
<td>W03456</td>
<td>X</td>
</tr>
<tr>
<td>WH02</td>
<td>SMITH</td>
<td>W03456</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2 – An example of the assignment table

When the program finds an entry in the table for the given resource, it will assign the corresponding warehouse order to it and deletes the order number from the database.
table entry. It may happen that the background search starts after that step, but it can’t find a new warehouse order in time so the warehouse number for the resource in the database table won’t be updated. In this case when the program finds an empty entry, it will start immediately the old search logic and sets the DISABLED flag to TRUE. This will result that when the background search is finished will not update the table entry since it was disabled.

5.3 Are new data types required and can I reuse existing functions?

Since I was improving an existing part of the software I didn’t have to create new data types only one database table (its schema is introduced in the previous chapter). I’ve created only two function modules:

- Z_RSRC_ASSIGNED_WO_TO_RESOURCE: this function module runs in the background and tries to find the most appropriate warehouse order for the resource
- Z_RSRC_CHECK_ASSIGNMENT: this function module checks in a database table what should be assigned to the resource.

The first letter indicates that the function modules in customer namespace (not standard SAP solution). For pseudo code please check chapter 5.4.5. Inside Z_RSRC_ASSIGNED_WO_TO_RESOURCE I’ve used the existing function modules to query the warehouse orders from the queues and a BAdI (Business Add-In) to calculate distance. A BAdI in SAP terminology means a special enhancement to the standard software which makes possible for the different customers to implement their own code inside the existing source code without modifying that.
5.4 What is ABAP?

ABAP (Advanced Business Application Programming) is an application-specific fourth-generation language which was originally a report language. ABAP was one of the first languages to include the concept of logical databases, which provides a high level of abstraction from the basic database level. ABAP was also intended to be used by SAP customers to enhance the standard SAP applications. As computer hardware evolved through the 1990s, more and more of SAP's applications and systems were written in ABAP. In 1999, SAP released an object-oriented extension to ABAP called ABAP Objects.

All ABAP programs reside inside the SAP database. In the database all code exists in two forms: source code, which can be viewed and edited with the ABAP workbench, and generated code, which is loaded and interpreted by the ABAP runtime system. ABAP programs run in the SAP application server, under control of the runtime system, which is part of the SAP kernel. The runtime system is responsible for processing ABAP statements, controlling the flow logic of screens and responding to events (such as a user clicking on a screen button). A key component of the runtime system is the Database Interface, which turns database-independent statements (Open SQL) into statements understood by the underlying DBMS (Native SQL). The database interface handles all the communication with the relational database on behalf of ABAP programs; it also contains extra features such as buffering of frequently accessed data in the local memory of the application server.
6. Test phase – analyzing the results

The most important part is to try out if the new solution works or not and what is the performance. I’ve created different test cases to simulate different real life situations to find out the truth about the effectiveness and weaknesses.

6.1. The process of the testing

I’ve created different test cases to simulate different real life situations to find out the truth about the effectiveness. I would like to pin down at the beginning that the most reliable test would be a real-life test with a live warehouse instead of a virtual SAP test warehouse. I’ve used smaller/larger warehouses to see how the shorter and longer distances affect the performance and how will the fitness values change when varying the LSD values.

I’ve used the transaction /SCWM/ADHOC in the SCM system to create warehouse orders. All of these orders contained only one task because it was not necessary to use more.

Since it was just a virtual test, I didn’t use any RF devices in a warehouse but entered the transaction /SCWM/RFUI. This is exactly the same environment what a worker could see on the display on the RF device, the following figure shows the screens and the consequent steps.
1. After the worker enters the RF environment he can see the following menu. He has to choose the first option.

2. Again, he has to enter the first option then the WO searching algorithm will run in the background and assign the most suitable order to the worker.

3. Very soon a screen appears with the number of the order offered by the system.

4. If the worker is happy with the job then he can check the most important information and start working.

Figure 6.1.1 – RF User Interface Screen sequence during warehouse order assignment
One remark for the test scenarios: the path from the current position of the worker to the first bin of the warehouse order is calculated via Manhattan distance metric.

6.2 Test scenarios

In the following test scenarios I've logged in the system always with the same resource of which resource type was assigned to two queues and its velocity was 3 m/s. In both cases 50-50 candidate warehouse orders were preselected for testing the searching algorithm. I’ve used a test warehouse three different test warehouse layouts. I wanted to
simulate smaller and bigger warehouses to observe the cost reduction and traveling times id different warehouses.

Of course a comparison to the original standard solution was necessary to examine if the new solution has a better performance in some or all cases. For that purpose I’ve also used the standard method for the same test data in every test case and I’ve made a short comparison, too.

6.2.1 Test case 1

In this test case the warehouse orders are scattered around within the warehouse. Please check the table 6.2.1.a in Appendix A for the detailed results.
At this test case the shortest traveling time was 3.33 seconds long meanwhile the longest one was 52 (in average it was 26.42). The original solution proposed the first line item from the table:

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Traveling time in (seconds)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>13:11:00</td>
<td>34.66</td>
<td>0</td>
<td>34.66</td>
</tr>
</tbody>
</table>

*Table 6.2.1.b – Proposed WO by the original approach*

The new method proposed the third line item:

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Traveling time in (seconds)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO03</td>
<td>13:11:28</td>
<td>03.33</td>
<td>28</td>
<td>31.33</td>
</tr>
</tbody>
</table>

*Table 6.2.1.c – Proposed WO by the new approach*

As we can see there is not so much difference between the fitness values but in the second case the worker spared 31, 33 seconds. If the warehouse order had been locked by a different worker in the database then the system would have chosen from this list (the first 10 orders sorted by the fitness value):

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Traveling time in seconds</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO03</td>
<td>13:11:28</td>
<td>03.33</td>
<td>28</td>
<td>31.33</td>
</tr>
<tr>
<td>WO01</td>
<td>13:11:00</td>
<td>34.66</td>
<td>0</td>
<td>34.66</td>
</tr>
<tr>
<td>WO02</td>
<td>13:11:10</td>
<td>42.66</td>
<td>10</td>
<td>52.66</td>
</tr>
<tr>
<td>WO04</td>
<td>13:11:45</td>
<td>25.33</td>
<td>45</td>
<td>70.33</td>
</tr>
</tbody>
</table>
6.2.2 Test case 2

In this test case I didn’t rearrange the first bins so I used the same layout. The difference is in the LSD values since there are bigger time gaps than in the previous test case. I wanted to know what the difference is if only LSD values are different, since the fitness values mainly depend on these parameters. The expected result is that due the higher penalties the new solution will propose the same warehouse order as the standard one. Please check table 6.2.2 in Appendix A for the detailed results.

Finally as it was expected, after the end of the test case the new method found the WO with the lowest LSD value the most appropriate choice.

6.2.3 Test case 3

The warehouse layout and the bin arrangement is still the same, I’ve changed only the LSD values again. Now I created groups of WOs which had the same LSD values, because of the following logic:

- There is a shipment which contains several products picked from different storage areas of the warehouse.

Table 6.2.1.d – First 10 proposed WOs sorted by the fitness value

<table>
<thead>
<tr>
<th>WO</th>
<th>Time</th>
<th>LSD</th>
<th>Distance</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO05</td>
<td>13:12:00</td>
<td>19.33</td>
<td>60</td>
<td>79.33</td>
</tr>
<tr>
<td>WO07</td>
<td>13:12:38</td>
<td>17.33</td>
<td>68</td>
<td>85.33</td>
</tr>
<tr>
<td>WO06</td>
<td>13:12:11</td>
<td>16.00</td>
<td>71</td>
<td>87.00</td>
</tr>
<tr>
<td>WO08</td>
<td>13:13:23</td>
<td>18.66</td>
<td>143</td>
<td>161.66</td>
</tr>
<tr>
<td>WO09</td>
<td>13:13:56</td>
<td>25.33</td>
<td>174</td>
<td>199.33</td>
</tr>
<tr>
<td>WO10</td>
<td>13:14:30</td>
<td>20.00</td>
<td>210</td>
<td>230.00</td>
</tr>
</tbody>
</table>
If the execution of the warehouse orders are independent from each other in this situation then the LSD values of the orders can be the same.

If the previous situation occurs then the warehouse orders in the same group have the same penalty points.

Now consider the results (table 6.2.3.a in APPENDIX A): the standard solution proposed the third order from the candidate list just like the new approach. The reason is that the LSD values are the same, so the next sorting parameter is the distance (the priority wasn’t used). Since the warehouse order with the smallest distance in the group has the smallest fitness value, the proposals are the same from both approaches.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO03</td>
<td>13:11:00</td>
<td>03,33</td>
<td>0</td>
<td>03,33</td>
</tr>
</tbody>
</table>

Table 6.2.3.b – The third candidate warehouse order is the best choice

The fitness value is much less due to the zero penalties. This property has changed the whole sorted candidate list of which first ten items are the following.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO03</td>
<td>13:11:00</td>
<td>03,33</td>
<td>0</td>
<td>03,33</td>
</tr>
<tr>
<td>WO06</td>
<td>13:11:00</td>
<td>16,00</td>
<td>0</td>
<td>16,00</td>
</tr>
<tr>
<td>WO05</td>
<td>13:11:00</td>
<td>19,33</td>
<td>0</td>
<td>19,33</td>
</tr>
<tr>
<td>WO04</td>
<td>13:11:00</td>
<td>25,33</td>
<td>0</td>
<td>25,33</td>
</tr>
<tr>
<td>WO01</td>
<td>13:11:00</td>
<td>34,66</td>
<td>0</td>
<td>34,66</td>
</tr>
<tr>
<td>WO02</td>
<td>13:11:00</td>
<td>42,66</td>
<td>0</td>
<td>42,66</td>
</tr>
</tbody>
</table>
It is not always necessary that the two methods will propose the same values just like in this situation. This happened because the warehouse order with the smallest fitness value was in the first order group. It is also a possible situation that the order with the smallest fitness value is in a latter group (which has higher penalty than zero but not so much) but due to its very short traveling time it is a better solution than any other orders within the group with zero penalties. Consider this example:

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMPLE_ORDER_01</td>
<td>10:00:00</td>
<td>48.33</td>
<td>0</td>
<td>48.33</td>
</tr>
<tr>
<td>EXAMPLE_ORDER_02</td>
<td>10:00:00</td>
<td>34.66</td>
<td>0</td>
<td>34.66</td>
</tr>
<tr>
<td>EXAMPLE_ORDER_03</td>
<td>10:00:00</td>
<td>42.66</td>
<td>0</td>
<td>42.66</td>
</tr>
<tr>
<td>EXAMPLE_ORDER_04</td>
<td>10:00:30</td>
<td>3.33</td>
<td>30</td>
<td>33.33</td>
</tr>
</tbody>
</table>

Table 6.2.3d – An example candidate list

In this example the standard method would choose the second order from the list since EXAMPLE_ORDER_01, EXAMPLE_ORDER_02 and EXAMPLE_ORDER_03 have the same LSD value but EXAMPLE_ORDER_02 has the lowest distance value. In my solution I use traveling times instead of distances but the shortest traveling time evidentially means the shortest distance. The new method proposes
EXAMPLE_ORDER_04 because in overall it requires the least time to travel and it has the smallest fitness value, too.

### 6.2.4 Test case 4

In this test case the size of the warehouse is smaller than in the previous one. Because of that I’ve created only 25 warehouse orders of which first bins are scattered around within the warehouse. Since it is a smaller layout the traveling distances are also shorter. Please check table 6.2.4.a in Appendix A for the detailed results.

![Figure 6.2.4 – Layout of the second warehouse](image)

At this test case the shortest traveling time was 4.66 seconds long meanwhile the longest one was 27.33 (in average it was 16.05). The original solution proposed the first line item from the table:
<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>14:00:00</td>
<td>20,66</td>
<td>0</td>
<td>20,66</td>
</tr>
</tbody>
</table>

*Table 6.2.4.b – Proposed WO by the original approach*

The new method proposed the third line item:

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO03</td>
<td>14:00:14</td>
<td>5,33</td>
<td>14</td>
<td>19,33</td>
</tr>
</tbody>
</table>

*Table 6.2.4.c – Proposed WO by the new approach*

Now in the second case the worker spared 15, 33 seconds. If the warehouse order had been locked by a different worker in the database then the system would have chosen from this list (the first 10 orders sorted by the fitness value):

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO03</td>
<td>14:00:14</td>
<td>5,33</td>
<td>14</td>
<td>19,33</td>
</tr>
<tr>
<td>WO01</td>
<td>14:00:00</td>
<td>20,66</td>
<td>0</td>
<td>20,66</td>
</tr>
<tr>
<td>WO02</td>
<td>14:00:10</td>
<td>12,66</td>
<td>10</td>
<td>22,66</td>
</tr>
<tr>
<td>WO04</td>
<td>14:00:35</td>
<td>16,00</td>
<td>35</td>
<td>51,00</td>
</tr>
<tr>
<td>WO06</td>
<td>14:01:05</td>
<td>10,66</td>
<td>65</td>
<td>75,66</td>
</tr>
<tr>
<td>WO05</td>
<td>14:00:55</td>
<td>23,33</td>
<td>55</td>
<td>78,33</td>
</tr>
<tr>
<td>WO07</td>
<td>14:01:15</td>
<td>16,66</td>
<td>75</td>
<td>91,66</td>
</tr>
<tr>
<td>WO08</td>
<td>14:01:35</td>
<td>22,66</td>
<td>95</td>
<td>117,66</td>
</tr>
<tr>
<td>WO09</td>
<td>14:01:55</td>
<td>12,66</td>
<td>115</td>
<td>127,66</td>
</tr>
</tbody>
</table>
6.2.5 Test case 5

The arrangement of starting bins is exactly the same as the previous one in the fourth test case. The only difference is only made in the LSD values again. Both methods proposed the same warehouse order as the most appropriate choice due to the incremented time gaps. More detailed results can be found in table 6.2.5 in Appendix A.

6.2.6 Test case 6

In this test case the most of the first bins are concentrated to the right side of the warehouse. Please check table 6.2.6.a in Appendix A for the detailed results.
At this test case the shortest traveling time was 7.33 seconds long meanwhile the longest one was 82 (in average it was 51.11 – it is higher due to the arrangement). The original solution proposed the first line item from the table:

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>16:00:00</td>
<td>54.66</td>
<td>0</td>
<td>54.66</td>
</tr>
</tbody>
</table>

*Table 6.2.6.b – Proposed WO by the original approach*

The new method proposed the third line item:

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO06</td>
<td>16:00:47</td>
<td>7.33</td>
<td>47</td>
<td>54.33</td>
</tr>
</tbody>
</table>

*Table 6.2.6.c – Proposed WO by the original approach*

With WO06 the worker can spare 47, 33 seconds. If the warehouse order had been locked by a different worker in the database then the system would have chosen from this list (the first 10 orders sorted by the fitness value):

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO06</td>
<td>16:00:47</td>
<td>7.33</td>
<td>47</td>
<td>54.33</td>
</tr>
<tr>
<td>WO01</td>
<td>16:00:00</td>
<td>54.66</td>
<td>0</td>
<td>54.66</td>
</tr>
<tr>
<td>WO02</td>
<td>16:00:08</td>
<td>57.33</td>
<td>8</td>
<td>65.33</td>
</tr>
</tbody>
</table>
6.2.7 Test case 7

I’ve used big time gaps again with unchanged layout and bin arrangement. Since the penalties were higher than before then both methods choose the first warehouse order from the candidate list. Please check table 6.2.7 in Appendix A for the detailed results.

6.2.8 Test case 8

At the last test case I’ve used the grouping principle again, so several warehouse orders have the same penalty points. The same situation occurred like in the third test case: both methods proposed the same warehouse order, but not the first one. Because of that there is no time spare. Please check table 6.2.8.a in Appendix A for the detailed results.
6.3 Analysis of the results and conclusion

I’ve tested my method on batches of warehouse orders. The batch size was 25/50 in different cases and in every case only one result was accepted as the best solution since in EWM only 1:1 assignment is accepted.

6.3.1 The running time of the algorithm

Since the problem was reduced to a 1:1 assignment therefore the applied algorithm is simpler. The running time is polynomial since the algorithm always works with the warehouse orders in a sequential way; for n orders in the main loop it requires n pieces of distance, penalty and fitness calculations. If there are more warehouse orders to be sorted the running time will not grow in exponential way. Sorting of the lists in ABAP also doesn’t require exponential time, these operations is very fast in the ABAP environment since ABAP was written to handle large amount of business data stored in database and internal tables (tables can be considered as special arrays). We can say that the complexity of the algorithm is $O(n)$ just like in the case of the original approach but since my modification allows working on maximum 200 warehouse orders instead of 50 (original method) then the running time can be 4 times longer. This approach is not faster than the original one (but not slower either) so I’ve improved it by only finding better solutions.
6.3.2 Comparing the new method to the proposed heuristics

If someone wants to compare two different methods then it is reasonable if they solve the same problem. There is no reason to compare a genetic algorithm which solves the TSP problem to a taboo search approach which solves the bin-packing problem. In my case the basis is an assignment problem, but the literature deals with m:n assignments and I have 1:1 in EWM. Furthermore, constraints are different and handled differently. Because of that I can’t do any comparison but at least I can compare the new method with the current standard one I can show what are the possibilities in the future.

6.3.3 The effectiveness of the new algorithm

As it turned out the new approach won’t guarantee always the shortest path but it wasn’t the original goal. If the purpose has been finding the shortest path then the only necessary thing would have been sorting the warehouse order list by distance instead of LSD values. The original goal was to keep the LSD as the sorting parameter with the highest priority and try to find a better warehouse order to travel less at the same time without letting the algorithm prioritize the latter orders. The reason behind that is simple: since LSD (which indicates urgency) is the most important sorting parameter it is not good if the algorithm assigns only less urgent WOs in every case just because the travelling time is much sorter. To find a good balance the computation of the penalty values is very important.

This solution applies the principle of the task interleaving, too. This doesn’t mean better performance regarding the running time but it has a good effect on the overall work distribution in the warehouse.
The following figure represents a warehouse with three queues and two warehouse orders which are available for the resource. If the priority of the queues is like that: Prio(Inbound) > Prio(Internal) > Prio(Outbound) then the system assigns the WO01 to the resource even it has to travel much more to the corresponding location. That was one problem which I wanted to eliminate by using the new algorithm. Even so for the test cases I’ve created only one queue for the test warehouses. I had to do this because by this way the standard method could use the same warehouse orders just like the new one with its global queue (which consisted of one queue in this case) and priorities didn’t bother the test. It made easier the comparison of the two methods to decide which finds the better resource/warehouse order assignments.

Figure 6.3.a – Three queues in a warehouse where the inbound has the biggest priority for the resource
So against the multi-queue principle I used one global queue which means all queues are merged into one, so there is no queue priority, only the distance and the LSD value matter. It means in this case that warehouse orders with smaller traveling distances will be considered as a possible option even if they were generated into other queues. Of course if the penalty for WO 02 is too high then my solution would propose WO 01 as the best option, too.

Figure 6.3.b – With using a global queue only distance and LSD matter – task interleaving is implemented by this way

I could show in the test cases the difference (sometimes there was no difference) between the two proposals so it was possible to define in percentage the gains. It is not possible to say that the new method is better in N percentage than the other one, because it highly depends on the current candidate order list, the time gaps between the
consequent orders and the distances. We could see that in its current state the fitness value mainly depends on the penalty values. Since one second equals to one penalty point a few seconds mean a lot of penalties therefore the latter orders will not be considered as an optimal solution.

So when does the new method find a better assignment than the standard one? If the ulterior orders in the sorted WO list (ulterior means that these WOs are less urgent) requires less traveling than the most urgent WOs and the time gap between them is not too much then then the new algorithm will prefer the ulterior warehouse orders.

But what is ‘not too much’? It depends on the way of calculating penalties. For the test I used this conversion: 1 second is 1 penalty point. After checking the result tables of the test cases we can confirm that sometimes even a minute difference (60 penalty points) is enough not to consider ulterior orders as best solutions. So sometimes it was a drawback but with further testing it can be decided which is the best conversion rate. Luckily this feature can be fine-tuned easily; it would be possible to create customizing options to define a different conversion e.g. 10 seconds should be 1 penalty point. I have shown that nowadays these applications are mainly based on the customizing settings due to the different demands. The test cases introduced warehouse layouts of different sizes, so different customizing values would be a reasonable option later. To decide this much more tests are needed, the best option would be a live test in a middle or larger warehouse.

The results also shown that the best solutions were selected from the first ten items of the list. This is ok, because the LSD has the biggest priority.
On the right hand I can say I’m satisfied but on the left hand I’d say there are a lot of things to do. During the planning and development process I learned a lot about warehousing, the different processes and methods, how IT is connected with real life with the business demands. The new solution could improve the warehouse order selection in cases which is a good point. But I also think there may be different fields in order assignment where further optimization is possible.

6.4 Future Work

One thing is sure now: further testing is needed because if the new method works properly without bugs and performance subsidence then it could be introduced in the new EWM 7.01 release. With more testing it will be decided how I could compute the penalty points in the best way. By using lower penalty points it would be possible to assign latter warehouse orders from the candidate list to the resource.

The current solution may get maximum 200 preselected warehouse orders to find the most appropriate one. During the tests it wasn’t justified for me that the constant 200 as maximum order number if too much, enough or too small. I’ve always created 50 (or 25) warehouse orders for the own queues and this amount is not enough to clarify this question.
References

[1] SAP Library – SAP EWM


[5] Warehousing Forum (February, 2001), Volume 16, Number 3,

   http://www.warehouse-science.com (2007.11.28)

   http://logistics.cat.com/cda/components/fullArticle?m=115228&x=7&id=382143 (2008.02.08)


Appendix A

1. Endnotes

1 Functions (function modules) in ABAP belonging to the so-called function groups (function pools) differ from are self-contained and do not belong to a specific program. ABAP functions accept as input any number of input parameters, return as output any number of output parameters, and raise exceptions if an error condition occurs. A very important feature of ABAP is the ability to call function modules in another SAP system or in an external application using the RFC (Remote Function Call) mechanism. It is also possible to call functions asynchronously; the ABAP program then does not wait for the function to return but instead continues immediately, while the function executes in a separate context.

2 SAPgui is the GUI client in SAP R/3's 3-tier architecture. It is software that runs on a Microsoft Windows, Apple Macintosh or UNIX desktop, and allows a user to access SAP functionality in SAP.

3 Steepest descent is an optimization algorithm for finding the nearest local minimum of a function which presupposes that the gradient of the function can be computed. It’s also called the gradient descent method.


5 The distance between two points measured along axes at right angles. In a plane with p$_1$ at (x$_1$, y$_1$) and p$_2$ at (x$_2$, y$_2$), it is |x$_1$ - x$_2$| + |y$_1$ - y$_2$|.

6 The Bellman–Ford algorithm computes single-source shortest paths in a weighted digraph (where some of the edge weights may be negative).
Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, outputting a shortest path tree. This algorithm is often used in routing.

The normal way of executing ABAP code in the SAP system is by entering a transaction code. Transactions can be accessed via system-defined or user-specific, role-based menus. They can also be started by entering their transaction code (a mnemonic name of up to 20 characters) in the special command field, which is present in every SAP screen. Transaction codes can be linked to screen elements or menu entries. Selecting such an element will start the transaction. The term "transaction" must not be misunderstood here: in the context just described, a transaction simply means calling and executing an ABAP program, not an indivisible operation on data, which is either committed as a whole or undone (rolled back) as a whole.
2. List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>Advanced Business Application Programming (originally Allgemeiner Berichts-Aufbereitungs-Prozessor = general report creation processor)</td>
</tr>
<tr>
<td>ACO</td>
<td>Ant Colony optimization</td>
</tr>
<tr>
<td>AS</td>
<td>Ant System</td>
</tr>
<tr>
<td>AS/RS</td>
<td>Automated Storage and Retrieval Systems</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management System</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>EWM</td>
<td>Extended Warehouse Management</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>GAP</td>
<td>Generalized Assignment Problem</td>
</tr>
<tr>
<td>HU</td>
<td>Handling Unit</td>
</tr>
<tr>
<td>LS</td>
<td>Local Search</td>
</tr>
<tr>
<td>LSD</td>
<td>Latest Starting Date</td>
</tr>
<tr>
<td>MRGAP</td>
<td>Multiple-Resource Generalized Assignment Problem</td>
</tr>
<tr>
<td>NP</td>
<td>Non-deterministic polynomial</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>SAP</td>
<td>Systems, Applications and Products in Data Processing (originally Systemanalyse und Programmentwicklung, later Systeme, Anwendungen und Produkte in der Datenverarbeitung)</td>
</tr>
<tr>
<td>SCM</td>
<td>Supply Chain Management</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TI</td>
<td>Task Interleaving</td>
</tr>
<tr>
<td>TS</td>
<td>Tabu Search</td>
</tr>
<tr>
<td>TSP</td>
<td>Traveling Salesman Problem</td>
</tr>
<tr>
<td>TU</td>
<td>Transport Unit</td>
</tr>
<tr>
<td>WO</td>
<td>Warehouse Order</td>
</tr>
<tr>
<td>WT</td>
<td>Warehouse Task</td>
</tr>
<tr>
<td>WMS</td>
<td>Warehouse Management System</td>
</tr>
</tbody>
</table>
3. List of figures

Figure 2.1.1 – Roles of the warehouse in the logistic chain (source: McGraw-Hill Education Group, 2001, Frazelle, Edward H. Supply Chain Strategy) ..................................................................................13

Figure 2.1.2 – Common warehouse activities (source: McGraw-Hill Education Group, 2001, Frazelle, Edward H. Supply Chain Strategy)..........................................................................................14

Figure 2.2 – Typical system architecture (source: McGraw-Hill Education Group, 2001, Frazelle, Edward H. Supply Chain Strategy) .............................................................................................16

Figure 2.3 – SAP User Interface............................................................................................................18

Figure 2.4 – Mapping of the physical warehouse ..................................................................................19

Figure 2.4.1.a – Warehouse of Toys R Us – carton-pick modules to the left, bulk storage to the right side (source of image: http://www2.isye.gatech.edu/~jjb/wh/) .......................................................................20

Figure 2.4.1.b – Different storage types in one warehouse complex .......................................................21

Figure 2.4.4 – Storage bins....................................................................................................................24

Figure 2.5 – Different warehouse tasks..................................................................................................26

Figure 2.6.1 – Some equipment in warehouses can not used for high-level storage spaces ..................27

Figure 2.6.2 – Handheld RF device .......................................................................................................28

Figure 3.2.2.a – An example of crossover ..............................................................................................34

Figure 3.4.2.b – An example of mutation ..............................................................................................35
Figure 3.2.3 – Roulette wheel selection .................................................................36

Figure 3.2.5 – An example solution .....................................................................37

Figure 4.2.2 – Example scenario of task interleaving .........................................46

Figure 4.3.1 – Assignment of resources to warehouse orders in SAP EWM 7.0 ....48

Figure 4.3.2 – Queues assigned to the same resource type .................................50

Figure 4.4.1.a – How congestions affect the warehouse sequences? .................55

Figure 4.4.1.b – Assignments are performed separately .....................................56

Figure 4.4.1.c – Which assignment is better from business point of view? .........57

Figure 4.4.2 – Which WO should be the next candidate? .................................57

Figure 4.5.5 – Central multiple assignment is not working in EWM .................67

Figure 6.1.1 – RF User Interface Screen sequence during warehouse order assignment ....77

Figure 6.1.2 – Manhattan distances ...................................................................78

Figure 6.2.1 – Layout of the first warehouse .......................................................79

Figure 6.2.4 – Layout of the second warehouse ..................................................84

Figure 6.2.6 – Layout of the first warehouse with different bin arrangement .......86

Figure 6.3.a – Three queues in a warehouse where the inbound has the biggest priority for the resource. .................................................................90
Figure 6.3.b – With using a global queue only distance and LSD matter – task interleaving is implemented by this way.
4. List of tables

Table 6.2 – An example of the assignment table. ................................................................. 74

Table 6.2.1.a – This table contains the selected warehouse orders for the first test case. Orders are sorted by their LSD values. Columns with gray background color are computed with the new search method. The entry of the best candidate proposed by the new method has a broader border.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Traveling time in seconds</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>13:11:00</td>
<td>34.66</td>
<td>0</td>
<td>34.66</td>
</tr>
<tr>
<td>WO02</td>
<td>13:11:10</td>
<td>42.66</td>
<td>10</td>
<td>52.66</td>
</tr>
<tr>
<td>WO03</td>
<td>13:11:28</td>
<td>03.33</td>
<td>28</td>
<td>31.33</td>
</tr>
<tr>
<td>WO04</td>
<td>13:11:45</td>
<td>25.33</td>
<td>45</td>
<td>70.33</td>
</tr>
<tr>
<td>WO05</td>
<td>13:12:00</td>
<td>19.33</td>
<td>60</td>
<td>79.33</td>
</tr>
<tr>
<td>WO06</td>
<td>13:12:11</td>
<td>16.00</td>
<td>71</td>
<td>87.00</td>
</tr>
<tr>
<td>WO07</td>
<td>13:12:28</td>
<td>17.33</td>
<td>68</td>
<td>85.33</td>
</tr>
<tr>
<td>WO08</td>
<td>13:13:23</td>
<td>18.66</td>
<td>143</td>
<td>161.66</td>
</tr>
<tr>
<td>WO09</td>
<td>13:13:56</td>
<td>25.33</td>
<td>174</td>
<td>199.33</td>
</tr>
<tr>
<td>WO10</td>
<td>13:14:30</td>
<td>20.00</td>
<td>210</td>
<td>230.00</td>
</tr>
<tr>
<td>WO11</td>
<td>13:14:48</td>
<td>10.00</td>
<td>228</td>
<td>238.00</td>
</tr>
<tr>
<td>WO12</td>
<td>13:16:00</td>
<td>11.33</td>
<td>300</td>
<td>311.33</td>
</tr>
<tr>
<td>WO13</td>
<td>13:16:15</td>
<td>12.66</td>
<td>315</td>
<td>327.66</td>
</tr>
<tr>
<td>WO14</td>
<td>13:16:20</td>
<td>15.33</td>
<td>320</td>
<td>325.33</td>
</tr>
<tr>
<td>WO16</td>
<td>13:17:00</td>
<td>08.66</td>
<td>360</td>
<td>368.66</td>
</tr>
<tr>
<td>WO17</td>
<td>13:19:30</td>
<td>20.66</td>
<td>510</td>
<td>530.66</td>
</tr>
<tr>
<td>WO18</td>
<td>13:19:37</td>
<td>22.66</td>
<td>517</td>
<td>539.66</td>
</tr>
<tr>
<td>WO19</td>
<td>13:19:45</td>
<td>32.00</td>
<td>525</td>
<td>557.00</td>
</tr>
<tr>
<td>WO20</td>
<td>13:19:55</td>
<td>38.00</td>
<td>535</td>
<td>573.00</td>
</tr>
<tr>
<td>WO21</td>
<td>13:20:00</td>
<td>36.66</td>
<td>540</td>
<td>576.66</td>
</tr>
<tr>
<td>WO22</td>
<td>13:20:15</td>
<td>47.33</td>
<td>555</td>
<td>602.33</td>
</tr>
<tr>
<td>WO23</td>
<td>13:20:20</td>
<td>37.33</td>
<td>560</td>
<td>597.33</td>
</tr>
<tr>
<td>WO24</td>
<td>13:20:45</td>
<td>20.66</td>
<td>585</td>
<td>605.66</td>
</tr>
<tr>
<td>WO25</td>
<td>13:20:50</td>
<td>23.33</td>
<td>590</td>
<td>613.33</td>
</tr>
<tr>
<td>WO26</td>
<td>12:21:00</td>
<td>17.33</td>
<td>600</td>
<td>617.33</td>
</tr>
<tr>
<td>WO27</td>
<td>12:21:05</td>
<td>36.66</td>
<td>605</td>
<td>641.66</td>
</tr>
<tr>
<td>WO28</td>
<td>13:21:10</td>
<td>38.66</td>
<td>610</td>
<td>648.66</td>
</tr>
<tr>
<td>WO29</td>
<td>13:21:35</td>
<td>20.66</td>
<td>635</td>
<td>655.66</td>
</tr>
<tr>
<td>WO31</td>
<td>13:22:00</td>
<td>34.00</td>
<td>660</td>
<td>694.00</td>
</tr>
<tr>
<td>WO32</td>
<td>13:22:05</td>
<td>52.00</td>
<td>665</td>
<td>717.00</td>
</tr>
<tr>
<td>WO33</td>
<td>13:22:30</td>
<td>29.33</td>
<td>690</td>
<td>719.33</td>
</tr>
<tr>
<td>WO34</td>
<td>13:22:35</td>
<td>22.00</td>
<td>695</td>
<td>717.00</td>
</tr>
<tr>
<td>WO35</td>
<td>13:22:40</td>
<td>42.66</td>
<td>700</td>
<td>742.66</td>
</tr>
<tr>
<td>WO36</td>
<td>13:22:50</td>
<td>43.33</td>
<td>710</td>
<td>755.33</td>
</tr>
<tr>
<td>WO37</td>
<td>13:23:00</td>
<td>21.33</td>
<td>720</td>
<td>741.33</td>
</tr>
<tr>
<td>WO38</td>
<td>13:22:05</td>
<td>27.33</td>
<td>725</td>
<td>752.33</td>
</tr>
<tr>
<td>WO39</td>
<td>13:22:15</td>
<td>14.00</td>
<td>735</td>
<td>749.00</td>
</tr>
</tbody>
</table>
Table 6.2.1.b – Proposed WO by the original approach

Table 6.2.1.c – Proposed WO by the new approach

Table 6.2.1.d – First 10 proposed WOs sorted by the fitness value

Table 6.2.2 – This table contains the selected warehouse orders for the second test case. Orders are sorted by their LSD values.
<table>
<thead>
<tr>
<th>Warehouse Order</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>13:11:00</td>
<td>34,66</td>
<td>0</td>
<td>34,66</td>
</tr>
<tr>
<td>WO02</td>
<td>13:11:00</td>
<td>42,66</td>
<td>0</td>
<td>42,66</td>
</tr>
<tr>
<td>WO03</td>
<td>13:11:00</td>
<td>03,33</td>
<td>0</td>
<td>03,33</td>
</tr>
<tr>
<td>WO04</td>
<td>13:11:00</td>
<td>25,33</td>
<td>0</td>
<td>25,33</td>
</tr>
<tr>
<td>WO05</td>
<td>13:11:00</td>
<td>19,33</td>
<td>0</td>
<td>19,33</td>
</tr>
<tr>
<td>WO06</td>
<td>13:11:00</td>
<td>16,00</td>
<td>0</td>
<td>16,00</td>
</tr>
<tr>
<td>WO07</td>
<td>13:13:00</td>
<td>17,33</td>
<td>120</td>
<td>137,33</td>
</tr>
<tr>
<td>WO08</td>
<td>13:13:00</td>
<td>18,66</td>
<td>120</td>
<td>138,66</td>
</tr>
<tr>
<td>WO09</td>
<td>13:14:00</td>
<td>25,33</td>
<td>180</td>
<td>205,33</td>
</tr>
<tr>
<td>WO10</td>
<td>13:16:00</td>
<td>20,00</td>
<td>300</td>
<td>320,00</td>
</tr>
<tr>
<td>WO11</td>
<td>13:16:00</td>
<td>10,00</td>
<td>300</td>
<td>310,00</td>
</tr>
<tr>
<td>WO12</td>
<td>13:16:00</td>
<td>11,33</td>
<td>300</td>
<td>311,33</td>
</tr>
<tr>
<td>WO13</td>
<td>13:16:00</td>
<td>12,66</td>
<td>300</td>
<td>312,66</td>
</tr>
<tr>
<td>WO14</td>
<td>13:16:00</td>
<td>15,33</td>
<td>300</td>
<td>315,33</td>
</tr>
<tr>
<td>WO15</td>
<td>13:16:00</td>
<td>26,66</td>
<td>300</td>
<td>326,66</td>
</tr>
</tbody>
</table>

Table 6.2.3.a – This table contains the selected warehouse orders for the third test case. Orders are sorted by their LSD values.
Table 6.2.3.b – The third candidate warehouse order is the best choice. .................................................82

Table 6.2.3.c – First 10 proposed WOs sorted by the fitness value..........................................................82

Table 6.2.4.a – This table contains the selected warehouse orders for the fourth test case. Orders are sorted by their LSD values.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>14:00:00</td>
<td>20,66</td>
<td>0</td>
<td>20,66</td>
</tr>
<tr>
<td>WO02</td>
<td>14:00:10</td>
<td>12,66</td>
<td>10</td>
<td>22,66</td>
</tr>
<tr>
<td>Warehouse Order Number</td>
<td>Latest Starting Date</td>
<td>Distance (traveling time)</td>
<td>Penalty (seconds)</td>
<td>Fitness Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td>---------------------------</td>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>WO01</td>
<td>14:00:00</td>
<td>20,66</td>
<td>0</td>
<td>20,66</td>
</tr>
<tr>
<td>WO02</td>
<td>14:01:00</td>
<td>12,66</td>
<td>60</td>
<td>72,66</td>
</tr>
<tr>
<td>WO03</td>
<td>14:01:00</td>
<td>5,33</td>
<td>60</td>
<td>65,33</td>
</tr>
<tr>
<td>WO04</td>
<td>14:01:30</td>
<td>16,00</td>
<td>90</td>
<td>106,00</td>
</tr>
<tr>
<td>WO05</td>
<td>14:01:30</td>
<td>23,33</td>
<td>110</td>
<td>133,33</td>
</tr>
<tr>
<td>WO06</td>
<td>14:02:20</td>
<td>10,66</td>
<td>140</td>
<td>150,66</td>
</tr>
<tr>
<td>WO07</td>
<td>14:02:55</td>
<td>16,66</td>
<td>175</td>
<td>191,66</td>
</tr>
<tr>
<td>WO08</td>
<td>14:05:00</td>
<td>22,66</td>
<td>300</td>
<td>322,66</td>
</tr>
<tr>
<td>WO09</td>
<td>14:05:00</td>
<td>12,66</td>
<td>300</td>
<td>312,66</td>
</tr>
</tbody>
</table>

Table 6.2.4.b – Proposed WO by the original approach

Table 6.2.4.c – Proposed WO by the new approach

Table 6.2.4.d – First 10 proposed WOs sorted by the fitness value

Table 6.2.5 – This table contains the selected warehouse orders for the fifth test case. Orders are sorted by their LSD values.
Table 6.2.6.a – This table contains the selected warehouse orders for the sixth test case. Orders are sorted by their LSD values.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>16:00:00</td>
<td>54.66</td>
<td>0</td>
<td>54.66</td>
</tr>
<tr>
<td>WO02</td>
<td>16:00:08</td>
<td>57.33</td>
<td>8</td>
<td>65.33</td>
</tr>
<tr>
<td>WO03</td>
<td>16:00:15</td>
<td>59.33</td>
<td>15</td>
<td>75.33</td>
</tr>
<tr>
<td>WO04</td>
<td>16:00:20</td>
<td>53.33</td>
<td>20</td>
<td>73.33</td>
</tr>
<tr>
<td>WO05</td>
<td>16:00:35</td>
<td>48.00</td>
<td>35</td>
<td>83.00</td>
</tr>
<tr>
<td>WO06</td>
<td>16:00:47</td>
<td>7.33</td>
<td>47</td>
<td>54.33</td>
</tr>
<tr>
<td>WO07</td>
<td>16:00:58</td>
<td>62.66</td>
<td>58</td>
<td>120.66</td>
</tr>
<tr>
<td>WO08</td>
<td>16:01:13</td>
<td>25.33</td>
<td>73</td>
<td>98.33</td>
</tr>
<tr>
<td>WO09</td>
<td>16:01:36</td>
<td>68.00</td>
<td>96</td>
<td>164.00</td>
</tr>
<tr>
<td>WO10</td>
<td>16:02:00</td>
<td>16.00</td>
<td>120</td>
<td>136.00</td>
</tr>
<tr>
<td>WO11</td>
<td>16:02:27</td>
<td>54.00</td>
<td>147</td>
<td>201.00</td>
</tr>
<tr>
<td>WO12</td>
<td>16:02:50</td>
<td>56.66</td>
<td>170</td>
<td>226.66</td>
</tr>
<tr>
<td>WO13</td>
<td>16:03:20</td>
<td>68.66</td>
<td>200</td>
<td>268.66</td>
</tr>
<tr>
<td>WO14</td>
<td>16:03:47</td>
<td>62.66</td>
<td>227</td>
<td>289.66</td>
</tr>
<tr>
<td>WO15</td>
<td>16:04:00</td>
<td>46.66</td>
<td>240</td>
<td>286.66</td>
</tr>
<tr>
<td>WO16</td>
<td>16:04:23</td>
<td>52.00</td>
<td>263</td>
<td>315.00</td>
</tr>
<tr>
<td>WO17</td>
<td>16:04:44</td>
<td>9.33</td>
<td>284</td>
<td>293.33</td>
</tr>
<tr>
<td>WO18</td>
<td>16:05:06</td>
<td>54.66</td>
<td>306</td>
<td>360.66</td>
</tr>
<tr>
<td>WO19</td>
<td>16:05:40</td>
<td>46.66</td>
<td>340</td>
<td>386.66</td>
</tr>
<tr>
<td>WO20</td>
<td>16:06:00</td>
<td>56.00</td>
<td>360</td>
<td>416.00</td>
</tr>
<tr>
<td>WO21</td>
<td>16:06:03</td>
<td>42.00</td>
<td>363</td>
<td>405.00</td>
</tr>
<tr>
<td>WO22</td>
<td>16:06:35</td>
<td>70.66</td>
<td>395</td>
<td>465.66</td>
</tr>
<tr>
<td>WO23</td>
<td>16:06:40</td>
<td>25.33</td>
<td>400</td>
<td>425.33</td>
</tr>
<tr>
<td>WO24</td>
<td>16:06:55</td>
<td>37.33</td>
<td>415</td>
<td>452.33</td>
</tr>
<tr>
<td>WO25</td>
<td>16:07:30</td>
<td>64.00</td>
<td>460</td>
<td>524.00</td>
</tr>
</tbody>
</table>
Table 6.2.6.b – Proposed WO by the original approach

<table>
<thead>
<tr>
<th>WO</th>
<th>Start Time</th>
<th>Distance</th>
<th>Penalty</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO26</td>
<td>16:20:00</td>
<td>24.00</td>
<td>1200</td>
<td>1224.00</td>
</tr>
<tr>
<td>WO27</td>
<td>16:20:00</td>
<td>28.00</td>
<td>1200</td>
<td>1228.00</td>
</tr>
<tr>
<td>WO28</td>
<td>16:20:30</td>
<td>28.00</td>
<td>1230</td>
<td>1258.00</td>
</tr>
<tr>
<td>WO29</td>
<td>16:20:45</td>
<td>58.00</td>
<td>1245</td>
<td>1303.00</td>
</tr>
<tr>
<td>WO30</td>
<td>16:20:55</td>
<td>61.33</td>
<td>1255</td>
<td>1316.33</td>
</tr>
<tr>
<td>WO31</td>
<td>16:22:00</td>
<td>51.33</td>
<td>1320</td>
<td>1371.33</td>
</tr>
<tr>
<td>WO32</td>
<td>16:22:30</td>
<td>31.33</td>
<td>1350</td>
<td>1381.33</td>
</tr>
<tr>
<td>WO33</td>
<td>16:22:50</td>
<td>56.66</td>
<td>1370</td>
<td>1426.66</td>
</tr>
<tr>
<td>WO34</td>
<td>16:24:10</td>
<td>46.66</td>
<td>1450</td>
<td>1496.66</td>
</tr>
<tr>
<td>WO35</td>
<td>16:24:23</td>
<td>32.66</td>
<td>1463</td>
<td>1495.66</td>
</tr>
<tr>
<td>WO36</td>
<td>16:24:50</td>
<td>53.33</td>
<td>1490</td>
<td>1543.33</td>
</tr>
<tr>
<td>WO37</td>
<td>16:24:50</td>
<td>65.33</td>
<td>1490</td>
<td>1555.33</td>
</tr>
<tr>
<td>WO38</td>
<td>16:24:50</td>
<td>46.66</td>
<td>1490</td>
<td>1536.66</td>
</tr>
<tr>
<td>WO39</td>
<td>16:24:50</td>
<td>73.33</td>
<td>1490</td>
<td>1563.33</td>
</tr>
<tr>
<td>WO40</td>
<td>16:27:00</td>
<td>39.33</td>
<td>1620</td>
<td>1718.33</td>
</tr>
<tr>
<td>WO41</td>
<td>16:27:23</td>
<td>75.33</td>
<td>1643</td>
<td>1717.33</td>
</tr>
<tr>
<td>WO42</td>
<td>16:27:46</td>
<td>51.33</td>
<td>1666</td>
<td>1717.33</td>
</tr>
<tr>
<td>WO43</td>
<td>16:27:50</td>
<td>82.00</td>
<td>1670</td>
<td>1752.00</td>
</tr>
<tr>
<td>WO44</td>
<td>16:28:12</td>
<td>66.00</td>
<td>1692</td>
<td>1758.00</td>
</tr>
<tr>
<td>WO45</td>
<td>16:28:40</td>
<td>66.66</td>
<td>1720</td>
<td>1786.66</td>
</tr>
<tr>
<td>WO46</td>
<td>16:28:50</td>
<td>48.00</td>
<td>1730</td>
<td>1778.00</td>
</tr>
<tr>
<td>WO47</td>
<td>16:45:00</td>
<td>62.66</td>
<td>2700</td>
<td>2762.66</td>
</tr>
<tr>
<td>WO48</td>
<td>16:45:20</td>
<td>68.00</td>
<td>2720</td>
<td>2788.00</td>
</tr>
<tr>
<td>WO49</td>
<td>16:45:20</td>
<td>79.33</td>
<td>2720</td>
<td>2799.33</td>
</tr>
<tr>
<td>WO50</td>
<td>16:45:20</td>
<td>64.00</td>
<td>2720</td>
<td>2784.00</td>
</tr>
</tbody>
</table>

Table 6.2.6.c – Proposed WO by the new approach

Table 6.2.6.d – First 10 proposed WOs sorted by the fitness value

Table 6.2.7 – This table contains the selected warehouse orders for the seventh test case. Orders are sorted by their LSD values.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling time)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO01</td>
<td>14:11:00</td>
<td>54.66</td>
<td>0</td>
<td>54.66</td>
</tr>
<tr>
<td>WO02</td>
<td>14:12:10</td>
<td>57.33</td>
<td>70</td>
<td>127.33</td>
</tr>
<tr>
<td>WO03</td>
<td>14:12:40</td>
<td>59.33</td>
<td>100</td>
<td>159.33</td>
</tr>
<tr>
<td>WO04</td>
<td>14:12:40</td>
<td>53.33</td>
<td>100</td>
<td>153.33</td>
</tr>
<tr>
<td>WO05</td>
<td>14:13:00</td>
<td>48.00</td>
<td>120</td>
<td>168.00</td>
</tr>
<tr>
<td>WO06</td>
<td>14:13:00</td>
<td>16.00</td>
<td>120</td>
<td>136.00</td>
</tr>
<tr>
<td>WO07</td>
<td>14:13:30</td>
<td>62.66</td>
<td>150</td>
<td>212.66</td>
</tr>
</tbody>
</table>
### Table 6.2.8.a

This table contains the selected warehouse orders for the eighth test case. Orders are sorted by their LSD values.

<table>
<thead>
<tr>
<th>Warehouse Order Number</th>
<th>Latest Starting Date</th>
<th>Distance (traveling)</th>
<th>Penalty (seconds)</th>
<th>Fitness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WO08</td>
<td>14:13:30</td>
<td>25.33</td>
<td>150</td>
<td>175.33</td>
</tr>
<tr>
<td>WO09</td>
<td>14:14:50</td>
<td>68.00</td>
<td>150</td>
<td>218.00</td>
</tr>
<tr>
<td>WO10</td>
<td>14:16:00</td>
<td>7.33</td>
<td>180</td>
<td>187.33</td>
</tr>
<tr>
<td>WO11</td>
<td>14:16:00</td>
<td>54.00</td>
<td>180</td>
<td>234.00</td>
</tr>
<tr>
<td>WO12</td>
<td>14:17:00</td>
<td>56.66</td>
<td>360</td>
<td>416.66</td>
</tr>
<tr>
<td>WO13</td>
<td>14:17:30</td>
<td>68.66</td>
<td>390</td>
<td>458.66</td>
</tr>
<tr>
<td>WO14</td>
<td>14:18:00</td>
<td>62.66</td>
<td>420</td>
<td>482.66</td>
</tr>
<tr>
<td>WO15</td>
<td>14:19:00</td>
<td>46.66</td>
<td>480</td>
<td>526.66</td>
</tr>
<tr>
<td>WO16</td>
<td>14:19:45</td>
<td>52.00</td>
<td>525</td>
<td>577.00</td>
</tr>
<tr>
<td>WO17</td>
<td>14:20:00</td>
<td>9.33</td>
<td>540</td>
<td>549.33</td>
</tr>
<tr>
<td>WO18</td>
<td>14:20:00</td>
<td>54.66</td>
<td>540</td>
<td>594.66</td>
</tr>
<tr>
<td>WO19</td>
<td>14:20:30</td>
<td>46.66</td>
<td>570</td>
<td>616.66</td>
</tr>
<tr>
<td>WO20</td>
<td>14:20:45</td>
<td>56.00</td>
<td>585</td>
<td>641.00</td>
</tr>
<tr>
<td>WO21</td>
<td>14:21:23</td>
<td>42.00</td>
<td>623</td>
<td>665.00</td>
</tr>
<tr>
<td>WO22</td>
<td>14:21:55</td>
<td>70.66</td>
<td>655</td>
<td>725.66</td>
</tr>
<tr>
<td>WO23</td>
<td>14:23:00</td>
<td>25.33</td>
<td>660</td>
<td>665.33</td>
</tr>
<tr>
<td>WO24</td>
<td>14:23:42</td>
<td>37.33</td>
<td>702</td>
<td>739.33</td>
</tr>
<tr>
<td>WO25</td>
<td>14:23:50</td>
<td>64.00</td>
<td>710</td>
<td>774.00</td>
</tr>
<tr>
<td>WO26</td>
<td>14:26:00</td>
<td>24.00</td>
<td>900</td>
<td>924.00</td>
</tr>
<tr>
<td>WO27</td>
<td>14:26:10</td>
<td>28.00</td>
<td>910</td>
<td>938.00</td>
</tr>
<tr>
<td>WO28</td>
<td>14:26:45</td>
<td>28.00</td>
<td>945</td>
<td>973.00</td>
</tr>
<tr>
<td>WO29</td>
<td>14:30:00</td>
<td>58.00</td>
<td>1140</td>
<td>1198.00</td>
</tr>
<tr>
<td>WO30</td>
<td>14:30:00</td>
<td>61.33</td>
<td>1140</td>
<td>1201.33</td>
</tr>
<tr>
<td>WO31</td>
<td>14:30:00</td>
<td>51.33</td>
<td>1140</td>
<td>1191.33</td>
</tr>
<tr>
<td>WO32</td>
<td>14:32:12</td>
<td>31.33</td>
<td>1272</td>
<td>1303.33</td>
</tr>
<tr>
<td>WO33</td>
<td>14:32:50</td>
<td>56.66</td>
<td>1310</td>
<td>1366.66</td>
</tr>
<tr>
<td>WO34</td>
<td>14:33:30</td>
<td>46.66</td>
<td>1340</td>
<td>1386.66</td>
</tr>
<tr>
<td>WO35</td>
<td>14:33:45</td>
<td>32.66</td>
<td>1365</td>
<td>1397.66</td>
</tr>
<tr>
<td>WO36</td>
<td>14:38:00</td>
<td>53.33</td>
<td>1620</td>
<td>1673.33</td>
</tr>
<tr>
<td>WO37</td>
<td>14:40:00</td>
<td>65.33</td>
<td>1740</td>
<td>1805.33</td>
</tr>
<tr>
<td>WO38</td>
<td>14:45:00</td>
<td>46.66</td>
<td>2063</td>
<td>2109.66</td>
</tr>
<tr>
<td>WO39</td>
<td>14:45:00</td>
<td>73.33</td>
<td>2040</td>
<td>2113.33</td>
</tr>
<tr>
<td>WO40</td>
<td>14:45:00</td>
<td>39.33</td>
<td>2040</td>
<td>2079.33</td>
</tr>
<tr>
<td>WO41</td>
<td>14:45:23</td>
<td>75.33</td>
<td>2063</td>
<td>2138.33</td>
</tr>
<tr>
<td>WO42</td>
<td>14:46:30</td>
<td>51.33</td>
<td>2130</td>
<td>2181.33</td>
</tr>
<tr>
<td>WO43</td>
<td>14:46:45</td>
<td>82.00</td>
<td>2145</td>
<td>2227.00</td>
</tr>
<tr>
<td>WO44</td>
<td>14:50:10</td>
<td>66.00</td>
<td>2350</td>
<td>2416.00</td>
</tr>
<tr>
<td>WO45</td>
<td>14:50:20</td>
<td>66.66</td>
<td>2360</td>
<td>2426.66</td>
</tr>
<tr>
<td>WO46</td>
<td>14:50:20</td>
<td>48.00</td>
<td>2360</td>
<td>2408.00</td>
</tr>
<tr>
<td>WO47</td>
<td>14:50:50</td>
<td>62.66</td>
<td>2390</td>
<td>2452.66</td>
</tr>
<tr>
<td>WO48</td>
<td>15:00:00</td>
<td>68.00</td>
<td>2940</td>
<td>3008.00</td>
</tr>
<tr>
<td>WO49</td>
<td>15:00:00</td>
<td>79.33</td>
<td>2940</td>
<td>3019.33</td>
</tr>
<tr>
<td>WO50</td>
<td>15:10:00</td>
<td>64.00</td>
<td>2940</td>
<td>3004.00</td>
</tr>
<tr>
<td>WO01</td>
<td>11:00:00</td>
<td>54,66</td>
<td>0</td>
<td>54,66</td>
</tr>
<tr>
<td>WO02</td>
<td>11:00:00</td>
<td>57,33</td>
<td>0</td>
<td>57,33</td>
</tr>
<tr>
<td>WO03</td>
<td>11:00:00</td>
<td>59,33</td>
<td>0</td>
<td>59,33</td>
</tr>
<tr>
<td>WO04</td>
<td>11:00:00</td>
<td>53,33</td>
<td>0</td>
<td>53,33</td>
</tr>
<tr>
<td>WO05</td>
<td>11:00:00</td>
<td>48,00</td>
<td>0</td>
<td>48,00</td>
</tr>
<tr>
<td>WO06</td>
<td>11:05:00</td>
<td>54,66</td>
<td>300</td>
<td>316,00</td>
</tr>
<tr>
<td>WO07</td>
<td>11:05:00</td>
<td>57,33</td>
<td>300</td>
<td>362,66</td>
</tr>
<tr>
<td>WO08</td>
<td>11:05:00</td>
<td>59,33</td>
<td>300</td>
<td>325,33</td>
</tr>
<tr>
<td>WO09</td>
<td>11:10:00</td>
<td>68,00</td>
<td>600</td>
<td>668,00</td>
</tr>
<tr>
<td>WO10</td>
<td>11:12:30</td>
<td>7,33</td>
<td>750</td>
<td>757,33</td>
</tr>
<tr>
<td>WO11</td>
<td>11:25:10</td>
<td>54,00</td>
<td>1510</td>
<td>1574,00</td>
</tr>
<tr>
<td>WO12</td>
<td>11:25:10</td>
<td>56,66</td>
<td>1510</td>
<td>1566,66</td>
</tr>
<tr>
<td>WO13</td>
<td>11:25:10</td>
<td>68,66</td>
<td>1510</td>
<td>1578,66</td>
</tr>
<tr>
<td>WO14</td>
<td>11:25:10</td>
<td>62,66</td>
<td>1510</td>
<td>1572,66</td>
</tr>
<tr>
<td>WO15</td>
<td>11:25:10</td>
<td>48,00</td>
<td>1510</td>
<td>1556,00</td>
</tr>
<tr>
<td>WO16</td>
<td>11:25:10</td>
<td>52,00</td>
<td>1510</td>
<td>1562,00</td>
</tr>
<tr>
<td>WO17</td>
<td>11:26:30</td>
<td>9,33</td>
<td>1590</td>
<td>1599,33</td>
</tr>
<tr>
<td>WO18</td>
<td>11:30:00</td>
<td>54,66</td>
<td>1800</td>
<td>1854,66</td>
</tr>
<tr>
<td>WO19</td>
<td>11:30:00</td>
<td>46,66</td>
<td>1800</td>
<td>1846,66</td>
</tr>
<tr>
<td>WO20</td>
<td>11:30:00</td>
<td>56,00</td>
<td>1800</td>
<td>1856,00</td>
</tr>
<tr>
<td>WO21</td>
<td>11:30:00</td>
<td>42,00</td>
<td>1800</td>
<td>1842,00</td>
</tr>
<tr>
<td>WO22</td>
<td>11:30:00</td>
<td>70,66</td>
<td>1800</td>
<td>1870,66</td>
</tr>
<tr>
<td>WO23</td>
<td>11:30:00</td>
<td>25,33</td>
<td>1800</td>
<td>1825,33</td>
</tr>
<tr>
<td>WO24</td>
<td>11:35:00</td>
<td>37,33</td>
<td>2100</td>
<td>2137,33</td>
</tr>
<tr>
<td>WO25</td>
<td>11:45:00</td>
<td>64,00</td>
<td>2700</td>
<td>2764,00</td>
</tr>
<tr>
<td>WO26</td>
<td>11:45:00</td>
<td>24,00</td>
<td>2700</td>
<td>2724,00</td>
</tr>
<tr>
<td>WO27</td>
<td>11:45:00</td>
<td>28,00</td>
<td>2700</td>
<td>2728,00</td>
</tr>
<tr>
<td>WO28</td>
<td>11:45:00</td>
<td>28,00</td>
<td>2700</td>
<td>2728,00</td>
</tr>
<tr>
<td>WO29</td>
<td>11:45:00</td>
<td>58,00</td>
<td>2700</td>
<td>2788,00</td>
</tr>
<tr>
<td>WO30</td>
<td>11:45:00</td>
<td>61,33</td>
<td>2700</td>
<td>2761,33</td>
</tr>
<tr>
<td>WO31</td>
<td>11:45:00</td>
<td>51,33</td>
<td>2700</td>
<td>2751,33</td>
</tr>
<tr>
<td>WO32</td>
<td>11:46:00</td>
<td>31,33</td>
<td>2760</td>
<td>2791,33</td>
</tr>
<tr>
<td>WO33</td>
<td>11:47:00</td>
<td>56,66</td>
<td>2820</td>
<td>2876,66</td>
</tr>
<tr>
<td>WO34</td>
<td>11:48:30</td>
<td>46,66</td>
<td>2910</td>
<td>2956,66</td>
</tr>
<tr>
<td>WO35</td>
<td>11:48:30</td>
<td>32,66</td>
<td>2910</td>
<td>2942,66</td>
</tr>
<tr>
<td>WO36</td>
<td>11:48:30</td>
<td>53,33</td>
<td>2910</td>
<td>2963,33</td>
</tr>
<tr>
<td>WO37</td>
<td>11:48:30</td>
<td>65,33</td>
<td>2910</td>
<td>2975,33</td>
</tr>
<tr>
<td>WO38</td>
<td>11:48:30</td>
<td>46,66</td>
<td>2910</td>
<td>2966,66</td>
</tr>
<tr>
<td>WO39</td>
<td>11:48:30</td>
<td>73,33</td>
<td>2910</td>
<td>2983,33</td>
</tr>
<tr>
<td>WO40</td>
<td>11:52:00</td>
<td>39,33</td>
<td>3120</td>
<td>3159,33</td>
</tr>
<tr>
<td>WO41</td>
<td>11:52:00</td>
<td>75,33</td>
<td>3120</td>
<td>3195,33</td>
</tr>
<tr>
<td>WO42</td>
<td>12:00:00</td>
<td>51,33</td>
<td>3600</td>
<td>3651,33</td>
</tr>
<tr>
<td>WO43</td>
<td>12:00:00</td>
<td>82,00</td>
<td>3600</td>
<td>3682,00</td>
</tr>
<tr>
<td>WO44</td>
<td>12:00:00</td>
<td>66,00</td>
<td>3600</td>
<td>3666,00</td>
</tr>
<tr>
<td>WO45</td>
<td>12:00:00</td>
<td>66,00</td>
<td>3600</td>
<td>3666,66</td>
</tr>
<tr>
<td>WO46</td>
<td>12:00:00</td>
<td>48,00</td>
<td>3600</td>
<td>3648,00</td>
</tr>
<tr>
<td>WO47</td>
<td>12:00:00</td>
<td>62,66</td>
<td>3600</td>
<td>3682,66</td>
</tr>
<tr>
<td>WO48</td>
<td>12:20:00</td>
<td>68,00</td>
<td>4800</td>
<td>4868,00</td>
</tr>
<tr>
<td>WO49</td>
<td>12:20:00</td>
<td>79,33</td>
<td>4800</td>
<td>4879,33</td>
</tr>
<tr>
<td>WO50</td>
<td>12:20:00</td>
<td>64,00</td>
<td>4800</td>
<td>4864,00</td>
</tr>
</tbody>
</table>
Table 6.2.8.b – Proposal of the methods.
Appendix B

1. Source code samples

The following source codes are all written in ABAP. They can run only on the application server of the development system which I’ve used.

The function Z_RSRC_ASSIGN_WO_TO_RESOURCE is always called as background process as a different thread. This means the main process is not forced to wait until the function module finds a suitable order and stores it in the database. It only requires two import parameters IV_RESOURCE and IV_LAST_ASSIGNED_WO. The former contains all resource relevant data e.g. which warehouse is it working in, what is its current position etc. The latter parameter is the latest WO assigned to the same resource which will be ignored during the candidate selection.

```abap
FUNCTION Z_RSRC_ASSIGN_WO_TO_RESOURCE.
  "----------------------------------------------------------------------
  "* Local Interface:
  "  IMPORTING
  "    REFERENCE(IS_RESOURCE) TYPE /SCWM/RSRC
  "    REFERENCE(IV_LAST_ASSIGNED_WO) TYPE /SCWM/DE_WHO
  "----------------------------------------------------------------------

  TYPES: BEGIN OF s_wo_distance.
    INCLUDE STRUCTURE /scwm/wo_rsrc_ty.
    TYPES   distance TYPE /scwm/de_travel_distance.
    TYPES   travel_time TYPE i.
    TYPES   penalty  TYPE i.
    TYPES   fittness TYPE i.
  TYPES: END OF s_wo_distance.

  DATA: lt_rsrc_grp_queue TYPE /SCWM/TT_RSGR_Q_SQ,
        lt_qualif_wo_rsrc_ty TYPE /scwm/ww_rsrc_ty,
        lt_common_wo_table TYPE /scwm/ww_rsrc_ty,
        ls_common_wo_table LIKE LINE OF lt_common_wo_table,
        lt_lgpla TYPE /scwm/ww_lgpla,
        lt_ordim_o TYPE /scwm/ww_ordim_o,
```

112
lt_distance TYPE TABLE OF s_wo_distance,
lt_path TYPE /scwm/tt_tdc_node,
lt_bapiret TYPE bapirettab,

ls_t340d TYPE /scwm/t340d,
ls_distance LIKE LINE OF lt_distance,
ls_who TYPE /SCWM/S_WHO_INT,
ls_prr TYPE /scmb/s_prr,
ls_rsrec_who TYPE S_RSREC_WO,
ls_attributes TYPE /scwm/s_who_att,

lv_distance TYPE /scwm/de_travel_distance,
lv_distance_z TYPE /scwm/de_travel_distance,
lv_severity TYPE bapi_mtype,
lv_dist_def TYPE /scwm/de_dist_def,
lv_unit_d TYPE /scwm/de_dist_uom,
lv_velocity TYPE /SCWM/DE_VELOCITY,
lv_velocity_uom TYPE /SCWM/DE_VELOCITY_UOM,
lv_who TYPE /SCWM/DE_WHO,
lv_lock_success TYPE xfeld,
lv_lm_active TYPE /scwm/de_lm_active,
lv_search_disabled TYPE xfeld,
lv_timezone TYPE TZNZONE,
lv_timestamp_first_wo TYPE T2NTSTMPS,
lv_lsd_date TYPE d,
lv_date_first_wo TYPE d,
lv_lsd_time TYPE t,
lv_difference TYPE t,
lv_time_first_wo TYPE t,
lv_tabix TYPE sytabix.

STATICS so_badi TYPE REF TO /scwm/ex_tdc_start.

FIELD-SYMBOLS:
<ls_rsrc_grp_queue> LIKE LINE OF lt_rsrc_grp_queue,
<ls_common_wo_table> LIKE LINE OF lt_common_wo_table,
<ls_distance> TYPE s_wo_distance,
<ls_ordim_o> LIKE LINE OF lt_ordim_o.

CONSTANTS:
loc_maximum_wo_number TYPE i VALUE 200.

*******************************************************************************
*******************************************************************************
* 0. Warehouse timezone is required later for timestamp conversion
CALL FUNCTION '/SCWM/LGNUM_TZONE_READ'
EXPORTING
IV_LGNUM = IS_RESOURCE-LGNUM
IMPORTING
EV_TZONE = lv_timezone
EXCEPTIONS
INTERFACE_ERROR = 1
DATA_NOT_FOUND = 2
OTHERS = 3
.
IF SY-SUBRC <> 0.
  lv_timezone = 'CET'.
ENDIF.
******************************************************************************
******************************************************************************
* 1. Collect the queues assigned to the resource group of the resource.
CALL FUNCTION '/SCWM/RSRC_RSGRP_QUEUE_GET'
  EXPORTING
    iv_lgnum = is_resource-lgnum
    iv_rsrc_grp = is_resource-rsrc_grp
  IMPORTING
    et_rsgrp_queue = lt_rsrc_grp_queue.
******************************************************************************
******************************************************************************
* 2. Unite the WOs in the different queues into one global list.
CLEAR lt_qualif_wo_rsrc_ty.
LOOP AT lt_rsrc_grp_queue ASSIGNING <ls_rsrc_grp_queue>.
  * Check authorization of user for queue assignment
  CALL FUNCTION '/SCWM/RSRC_QUEUE_AUTH_CHECK'
    EXPORTING
      iv_lgnum = <ls_rsrc_grp_queue>-lgnum
      iv_queue = <ls_rsrc_grp_queue>-queue
    EXCEPTIONS
      no_authorization = 1
      OTHERS = 2.
  * Omit unauthorized queues
  IF sy-subrc <> 0.
    CONTINUE.
  ENDIF.
  * Get warehouse orders records for queue
  CALL FUNCTION 'Z_RSRC_WHO_RSTYP_GET'
    EXPORTING
      iv_lgnum = is_resource-lgnum
      iv_rfnd = is_resource-rfnd
      iv_rsrc_type = is_resource-rsrc_type
      iv_queue = <ls_rsrc_grp_queue>-queue
    CHANGING
      ct_wo_rsrc_ty = lt_qualif_wo_rsrc_ty.
  IF lt_qualif_wo_rsrc_ty[] IS INITIAL.
    CONTINUE.
  ENDIF.
  * add WOs from the current queue to the common WO container table
APPEND LINES OF lt_qualif_wo_rsnc_tty TO lt_common_wo_table.
ENDLOOP.

*******************************************************************************
*******************************************************************************
* 3. Sort the order items by LSD and resource priority.
  SORT lt_common_wo_table BY lsd ASCENDING priority DESCENDING.
  READ TABLE lt_common_wo_table INTO ls_common_wo_table INDEX 1.
  lv_timestamp_first_wo = ls_common_wo_table-lsd.

* save the time and date components of the first timestamp
  CONVERT TIME STAMP lv_timestamp_first_wo TIME ZONE lv_timezone INTO DATE lv_date_first_wo TIME lv_time_first_wo.

*******************************************************************************
*******************************************************************************
* 4. Keep only the first n WO items.
  DELETE lt_common_wo_table FROM lc_maximum_wo_number.
  DELETE lt_common_wo_table WHERE who EQ IV_LAST_ASSIGNED_WO.

*******************************************************************************
*******************************************************************************
* 5. Loop through every WO and
*   Calculate distance from the current position to the WO start location.

lt_distance = lt_common_wo_table.

IF so_badi IS NOT BOUND.
  GET BADI so_badi
    FILTERS
      lgnum = is_resource-lgnum.
ENDIF.

* get warehouse relevant information
CALL FUNCTION '/SCWM/T340D_READ_SINGLE'
  EXPORTING
    iv_lgnum = is_resource-lgnum
  IMPORTING
    es_t340d = ls_t340d
EXCEPTIONS
  OTHERS = 0. "#EC NO_HANDLER"

LOOP AT lt_distance ASSIGNING <ls_distance>.
  lv_tabix = sy-tabix.
  CLEAR lt_lgpla.

  * collect the WTs for the WO to determine the first source bin
  CALL FUNCTION '/SCWM/TO_READ_WHO'
    EXPORTING
      iv_lgnum = is_resource-lgnum
iv_who = <ls_distance>-who
iv_flglock = '
IMPORTING
et_ordin_o = lt_ordin_o
EXCEPTIONS
wrong_input = 1
not_found = 2
foreign_lock = 3
error = 4
OTHERS = 5.
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

* distance calculation between the current position of the resource
* and the the source bin of the first WT of the WO.
IF LINES( lt_ordin_o ) = 0.
  <ls_distance>-distance = wmegc_infinity.
  CONTINUE.
ENDIF.
READ TABLE lt_ordin_o ASSIGNING <ls_ordin_o> INDEX 1.
APPEND <ls_ordin_o>-vlpla TO lt_lgpla.
APPEND is_resource-actual_bin TO lt_lgpla.

* use BAdI for distance calculation
TRY.
  CALL BADI so_badi->calc
  EXPORTING
    iv_lgnum = is_resource-lgnum
    it_lgpla = lt_lgpla
    is_rsrc = is_resource
    iv_dist_first = lv_dist_def
    iv_unit_d = lv_unit_d
    iv_use_euclid = ls_t340d-use_euclid
  IMPORTING
    ev_travel_distance = lv_distance
    ev_travel_distance_z = lv_distance_z
    ev_unit_dist = lv_unit_d
    et_path = lt_path
    et_bapiret = lt_bapiret
    ev_severity = lv_severity.
  CATCH cx_badi.
ENTRY.

<ls_distance>-distance = lv_distance.

* compute penalty points
IF lv_tabix EQ 0.
  <ls_distance>-penalty = 0.
\( <\text{ls\_distance}>\text{-fittness} = lv\_distance. \)

\text{ELSE.}

\text{CONVERT TIME STAMP} \ <\text{ls\_distance}>\text{-}

\text{LSD TIME ZONE} \ lv\_timezone \ INTO \ DATE \ lv\_lsd\_date \ TIME \ lv\_lsd\_time. \)

\( lv\_difference = lv\_lsd\_time - lv\_time\_first\_wo. \)

\( <\text{ls\_distance}>\text{-penalty} = lv\_difference \mod 86400. \ " \text{difference in seconds} \)

\( <\text{ls\_distance}>\text{-fittness} = <\text{ls\_distance}>\text{-penalty} + lv\_distance. \)

\text{ENDIF.}

\text{ENDLOOP.}

*******************************************************************************
*******************************************************************************
* 6. Sort the order items by fittness value -
* the lowest fittness value is the best option
* sort \( lt\_distance \) by fittness \text{ASCENDING.} \)

*******************************************************************************
*******************************************************************************
* 7. check if the proposed WO is computed in time or not.
* loop over the possible candidates
\text{LOOP AT} \ lt\_distance \ \text{ASSIGNING} \ <\text{ls\_distance}>. \)

\text{IF} \ <\text{ls\_distance}>\text{-status} \ \text{IS INITIAL OR}
\text{<ls\_distance>-status EQ wmegc\_wo\_in\_process.}

\text{CLEAR: ls\_who, lt\_ordim\_o.}

\text{TRY.}

* check if it's possible to assign
\text{CALL FUNCTION} 'SCWM/WHO\_GET'
\text{EXPORTING}
\ \ iv\_lgnum \ = \ is\_resource\_lgnum
\ iv\_to \ = \ gc\_x
\ iv\_lock\_who \ = \ gc\_x
\ iv\_lock\_to \ = \ gc\_x
\ iv\_whoid \ = \ <\text{ls\_distance}>\text{-who}
\text{IMPORTING}
\ \ es\_who \ = \ ls\_who
\ et\_ordim\_o \ = \ lt\_ordim\_o.

\text{IF NOT} \ ls\_who\_who \ \text{IS INITIAL. } "\text{Field DB\_LOCK can be field with X}

* Unlock WO
\text{CALL FUNCTION} 'DEQUEUE_/SCWM/WHO'\text{EXPORTING}
\ \ lgnum \ = \ is\_resource\_lgnum
\ \ who \ = \ ls\_who\_who.

* Unlock the WTs also
LOOP AT lt_ordim_o ASSIGNING <ls_ordim_o>. 
   CALL FUNCTION 'DEQUEUE_/SCWM/ELLTAKE' 
      EXPORTING 
          lgnum = is_resource-lgnum 
          tanum = <ls_ordim_o>-tanum. 
   ENDLOOP.

MOVE-CORRESPONDING ls_who TO ls_attributes. 
ls_attributes-rsrc = is_resource-rsrc.

* Check the Labor Management is active or not in the warehouse 
CALL FUNCTION '/SCWM/WHO_CHECK_LM_ACTIVE' 
   EXPORTING 
       iv_lgnum     = is_resource-lgnum 
       iv_who       = ls_who-who 
   IMPORTING 
       ev_lm_active = lv_lm_active. 
IF lv_lm_active = abap_true. 
    LM active 
    * get the processor for the current user 
    CALL FUNCTION '/SCMB/PRR_READ_SINGLE' 
       EXPORTING 
           iv_bname    = sy-uname 
       IMPORTING 
           es_prr      = ls_prr 
       EXCEPTIONS 
           wrong_input = 1 
           not_found   = 2 
           OTHERS      = 3. 
   IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno 
         WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. 
   ENDIF.
   ls_attributes-processor = ls_prr-prr_id.
ELSE. 
    * LM not active 
    ls_attributes-processor = sy-uname.
ENDIF.

* check if the proposed WO is required or not 
SELECT SINGLE search_disabled FROM ZRSRC_WO INTO lv_search_disabled 
   WHERE lgnum = is_resource-lgnum AND 
       rsrc  = is_resource-rsrc. 
IF lv_search_disabled EQ 'X'.
   * the new WO wasn't found in time, we have to drop this solution and 
   use the "old way" 
   CONTINUE.
ELSE. 
   * assign the WO to the resource 
   CALL FUNCTION '/SCWM/WHO_UPDATE' 

EXPORTING
   iv_lgnum      = is_resource-lgnum
   iv_db_update  = 'X'
   iv_who        = ls_who-who
   is_attributes = ls_attributes
   iv_synchron   = 'X'

EXCEPTIONS
   read_error    = 1
   attributes    = 2
   OTHERS        = 3.

IF sy-subrc <> 0.
   CONTINUE.
ELSE.
   *        locking is successful, write the new assignment into the table
   CLEAR ls_rsrc_who.
   ls_rsrc_who-lgnum = is_resource-lgnum.
   ls_rsrc_who-rsrc = is_resource-rsrc.
   ls_rsrc_who-who = ls_who-who.
   ls_rsrc_who-search_disabled = ' '.
   UPDATE ZRSRC_WO FROM ls_rsrc_who.
   EXIT.
ENDIF.
ENDIF.

*         it was not possible to lock the WO, check the next nominee
ELSE.
   CONTINUE.
ENDIF.

*         Exception in case of locking; Continue with the next nominee
CATCH /scwm/cx_core.
   CONTINUE.
ENDTRY.
ENDIF.

* end of loop checking the candidate solutions
ENDLOOP.
ENDFUNCTION.

The function Z_RSRC_CHECK_ASSIGNMENT is a simple function module which checks the database table ZRSRC_WO if is there any warehouse order assigned to the resource or not. If there is no entry then the main process will use the old standard logic to choose a warehouse order and at the same time it will set a switch to deny any new assignment proposal which was found by the background process.
FUNCTION Z_RSRC_CHECK_ASSIGNMENT.
**"----------------------------------------------------------------------
**"Local Interface:
**" IMPORTING
**"     REFERENCE(IS_RESOURCE) TYPE /SCWM/RSRC
**" EXPORTING
**"     REFERENCE(EV_WHO) TYPE /SCWM/DE_WHO
**"----------------------------------------------------------------------

DATA: ls_rsrc_who TYPE S_RSRC_WO.

* get the assigned WO
SELECT SINGLE who FROM ZRSRC_WO INTO ev_who
  WHERE lgnum = is_resource-lgnum AND
        rscc = is_resource-rsrc.

*if there is no entry for the resource then create an empty one
IF sy-subrc NE 0.
  CLEAR ls_rsrc_who.
  ls_rsrc_who-lgnum = is_resource-lgnum.
  ls_rsrc_who-rsrc = is_resource-rsrc.
  ls_rsrc_who-who = ''.
  ls_rsrc_who-search_disabled = ''.
  INSERT ZRSRC_WO FROM ls_rsrc_who.
ENDIF.

CLEAR ls_rsrc_who.

* if there wasn't any proposed WO then indicate that the background
  search shall be ignored for the current resource
IF ev_who IS INITIAL.
  ls_rsrc_who-lgnum = is_resource-lgnum.
  ls_rsrc_who-rsrc = is_resource-rsrc.
  ls_rsrc_who-search_disabled = 'X'.
  UPDATE ZRSRC_WO FROM ls_rsrc_who.
ELSE.
  ls_rsrc_who-lgnum = is_resource-lgnum.
  ls_rsrc_who-rsrc = is_resource-rsrc.
  ls_rsrc_who-who = ''.
  ls_rsrc_who-search_disabled = ''.
  UPDATE ZRSRC_WO FROM ls_rsrc_who.
ENDIF.

ENDFUNCTION.
SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies.