GLOBAL OPTIMIZATION OF LOCAL WORKFLOW MANAGERS USING THE EXAMPLE OF AIRPORT HAMBURG

Yousef Farschtschi[1], Marc Widemann[1], Jochen Wittmann[2], Dietmar P. F. Möller[1]

[1]University of Hamburg, Faculty of Informatics,
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[2]University of Applied Sciences, Faculty of Ecological Informatics,
Wilhelminenhofstr. 75A, 12459 Berlin, Germany

farschtschi@informatik.uni-hamburg.de (Yousef Farschtschi)

Abstract

We present an approach on global optimization of interdependent workflows. The goal was first to create a concept and than a prototype that optimizes these interdependent workflows in any area with its requirements. The requirements for the prototype will be presented in this contribution by using the airport context. The main goal of the optimization is to improve the economic efficiency in these areas like optimizing the time to decrease the costs. Workflow is a term that can be found in different branches of commerce for example in logistic areas like airports. Therefore this contribution will use the context of the airport Hamburg to present the concept and the prototype of this contribution. The prototype will present in an exemplary way the global optimization of the ground handling workflows in the airport. These are for example the transit of the passengers on the apron, the dispatching of the luggage, the catering of an aeroplane and the refuelling. These ground handling workflows contain local optimizers.

The prototype contains a simple GUI (graphic user interface) for the representation. Because of the quick access the interface of the prototype is based on a local database. Furthermore the access to the real existing workflows was not possible. Unfortunately this is a great hurdle and will be justified in this contribution.

Keywords: Workflow, Optimization, Module, Scenario.

Presenting Author's biography

Yousef Farschtschi. He studied computer science at the University of Hamburg and achieved his diploma in 2009 [1] in the field of airport optimization and workflow optimization. On the basis of his experience in the airport area, he has made more research activities in this context see e.g. [2, 3]). Now he is working in the University of Hamburg in the Faculty of Informatics as an employee and Ph.D. student.

1 Introduction

The concept and prototype of this contribution will present the global optimization of interdependent workflows. They will be adoptable in areas with their requirements. The requirements of the prototype are structured in the used data base and will be exemplarily presented in section 6. Other logistic areas can be adapted on this concept by a user. We will use the airport example to present the approach.

For this example we concentrate on the ground-handling in the airport of Hamburg. The optimization of a single workflow of the airport Hamburg, the luggage transport is presented in [3]. Luggage transport is one of many possible service-providers which are responsible for the ground-handling of an aeroplane in the airport of Hamburg. A service-provider has to discharge their tasks in an allowed time. For example the luggage transport has to deliver the first luggage of the business class on the luggage carousel at least in 7 minutes after the shut-down of the engines. This is a contractually established agreement.

The next section will discuss the basic problem of the global optimization. Afterwards the used terms will be explained. Then the optimization task for the concept will be described and the prototype will be introduced.

2 Basic structure of the problem

The main instance is the global optimizer that executes a task. This task can execute any number of subtasks. The concept has a hierarchical workflows and a task could be a subtask. The hierarchical structure of the workflows is a result of their properties like priority and interdependency. Therefore two terms will be introduced module and scenario. In a nutshell modules present subprograms like the different ground handling workflows and a scenario is an optional number of these modules or accordingly ground handling workflows with their interdependency. They will be explained in the next sections.

The technical challenge is the modular concept of the prototype. The prototype should be deployable in any logistic area like an airport with its requirements. The prototype works as a higher ranked instance for the chosen modules like the different ground handling workflows. Because of the deregulation of the airports, the service-providers are independent subcontractors. The deregulation had an impact on the competition, but these subcontractors have to open their interfaces to higher ranked managing software to implement this concept. Because of their data privacy this intention was blocked. This problem affects, that the prototype works exemplary data for the modules or accordingly ground handling workflows. Furthermore there is regrettably no connection to a real service provider and its workflow.

3 Term definition

A local optimizer like the one in [3] is used by the equivalent terms “service provider workflow” and “ground handling workflow”. If a local optimizer works as a subtask for the global optimizer the term module can be used.

3.1 Terms in the airport context

The usage of the airport example was a result of the participation in the WFF-Project (in German “Wett-bewerbsfähiger Flughafen”/ in English “competitive airport”). As mentioned before the paper [3] gives a general view of the activities and the approach. In a nutshell an optimizer was implemented for the luggage transport of the subcontractor GROUND-STARS [4]. This local optimizer creates automatically tasks for the luggage cars by checking their status (like proposed, free or failure) and the path length.

The ground handling of an aeroplane is subdivided into several workflows. These are for example: boarding and accordingly deboarding of the passengers, cleaning of the cabin, catering, loading and accordingly unloading of the luggage, waste and portable water service and refuelling. The prototype uses exemplary four workflows: passengers, luggage, catering and refuelling.

3.2 Module and scenario definition

The mentioned service-provider workflows were considered as modules. The module concept was first published by Parnas [5]. A module is a complete component of software. It contains a chain of pass. Because of the separation of interface and implementation, modules allow casing. A module is a subprogram and delivers after a pass the result to the higher instance. This could be a module like the luggage transport that optimize its single workflow and delivers its task start times and task end times to the global instance. There are different reasons using modules:

- The program logic is reusable without creating redundant code
- They can be compiled separately in different programming languages and can be allocated as libraries of programs; for example different service provider use different program languages for the optimization of their single workflow
- Complex programs can be arranged by modules and functionalities can be included by the modular conception
- Different developers are able to edit and test independently some modules

The used term “scenario” is a description of an event or series of actions and events. It comprises an environment with a specific state. Today there are a lot of areas of application like economic or society based areas, in which this term finds usage:

- Preparation of decisions

- Orientation concerning future deployment
- Creating and checking of strategies

In our context a scenario can be described as an environment in which our modules are interdependent and have to be run in a specific time. Every scenario has an own environment and can be run any time.

The next section will give a closer definition by using the airport example.

4 Module and scenario administration

The different ground handling workflows on an airport could be seen as modules, if they use a global instance. Any numbers of modules can be combined in a scenario. The prototype is responsible for the optimization of the chosen modules in the scenario.

4.1 Module administration

To keep the complexity of the first prototype on a lower level and as mentioned because of the difficulty to retrieve a connection to the interface of the service-provider workflows, the decision was made to use a local data base as data input for the modules. The prototype uses exemplary four ground handling workflows: the transit of the passengers on the apron, the dispatching of the luggage, the catering of an aeroplane and the refuelling. For each of these different service provider workflows a table in the data base was created. The following data is used for the optimization. It is statically kept in the database table, except for the priorities, which may be dynamically altered.

- priority: every module or ground handling workflow has its own priority in the chain of the global workflow; a lower number is a higher priority
- common dimension: the event of the optimization needs a common dimension; for example the aeroplane Id in the airport context; this is important for the connection of the data set of the different modules
- resource: every module has its own resources; for example the resources Ids of the passenger transport are busses and of the refuelling are tank lorries
- start and end time: this describes the start- and end-time of a task in one workflow

Due to the fact that modules work as subprograms, the module data have to be saved locally to ensure the consistency. The following data will be saved on the hard disk after the creation of a module: module name, priority, database name, task table name, common dimension column name, resource column, task start time column name and task end time column.

Section 6 will show the procedure of the module creation by using an example.

4.2 Scenario administration

The scenarios are in the context of the prototype responsible for interdependence of the modules. This can be set by a user and will be exemplary introduced in section 6. For creating a scenario the prototype requires the following information.

- Chosen modules: describes chosen modules for this scenario
- Time frame: the length of a scenario is defined by the start and end times of the modules and either they run simultaneously or not; the timeframe defines the time limit for a scenario (in minutes)
- Module combination: describes the interdependence of the modules; there can be chosen at least one or none

The consistency of a scenario has to be ensured. Therefore the scenario data will be saved locally, too. It contains the same field as before: “chosen modules”, “timeframe” and “module combinations”.

4.3 Adaptation in the airport context

These terms now will be concretized in the airport context.

Every ground handling workflow can be represented by a module and every module uses a local optimizer. These modules have to be synchronized by a global instance.

A scenario presents a number of modules and their interdependency. Some service-providers are not able to work their task concurrently. Therefore the user has to define which module is able to run concurrently with another. The used time frame ensures the observance of time in a scenario. After the execution of a scenario by the global optimizer the data of the modules will be connected by using the common dimension ID. The optimization task will be described in the next section.

5 Optimization task

This section will describe the target of the optimization of the concept. Some of the now described properties of the concept are not implemented in the prototype. This will be discussed later.

5.1 Optimization

Now the optimization procedure of the concept will be discussed.

The goal of this contribution is the optimization of interdependent workflows. After the creation of some modules, any number of them has to be bundled in a scenario. All modules have a common dimension and this has to have a unique Id. This is an important requirement for the adaptation in other contexts. On this condition the procedure of the optimization can be started.

The global optimizer will first upload the start and end times from all modules and sort them by the common dimension. It can now choose the start time for each common dimension by searching the earliest one.

The modules have a priority and interdependence. Both properties are important for the optimization. A higher ranked module has to be started earlier, even though the interdependence does not allow a concurrent start with other modules. The time frame is the constraint of the optimization. If the time frame passes over and the global optimizer does not found another constellation, because of the priorities and specialy the interdependence of the modules, the modules have to be advised. The modules now have to adjust the periods of time to the changes of the global optimizer. If a resource of a module finishes a task, it will inform the module and the new optimized times will be proposed to the global optimizer. The modules are responsible for the optimization of the available resources by using the time constraints of the global optimizer. The real-time requirements are the untroubled transmission of the data between the modules and the global optimizer. The global optimizer now has to restart the optimization. There is a constant exchange between the modules and the global optimizer. After the optimization task of the global instance the modules will be shown in an optimal order.

The adaptation in the airport context will be shown in the next section. An example will also show the optimized procedure and the non optimized procedure.

5.2 Interface

This concept emerged assuming that the modules were created interdependently. Therefore an interface has to be created that is compatible with different systems. Two possibilities for the implementation will now be represented. The first one is to implement for the global optimizer an interface to each module and the second one is to implement for each module an interface to the global optimizer. These two options depend on the factors time and cost and have to be decided by the module creators or the creators of the global optimizer. The experiences at the airport showed us that the handling with such data streams can be handled by using data local bases for modules.

As mentioned the adaptation in the airport context uses a local database with test data, too.

6 Implementation

After defining and explaining the terms and the concept of this contribution the prototype now will be specified.

6.1 Prototype

As mentioned before the prototype should be deployable in any area of commerce. The first decision was to use a platform independent programme language. Java achieves this condition and delivers an expandable object-oriented platform. The prototype uses a simple GUI (graphic user interface) for the interaction. It contains the wizard principle for creating the modules and scenarios. This principle ensures the usability of the prototype. The module and scenario creation requires several data and these steps are shown in the figures 1 and 4. This will be discussed in subsection 3.

6.2 Algorithm

While implementing the prototype the attention was to the algorithm. It is based in parts on the “mergesort”-algorithm [6, 7]. The algorithm of the prototype is complex and the following marks few important steps for the better understanding:

1. Modules and a scenario have to be created. The priority and interdependence have to be set and chosen.
2. After the start of a scenario the modules will be sorted by the common dimension
3. The earliest start time for each common dimension will be searched
4. All chosen modules will be sorted by their priority. A lower value is a higher priority.
5. The interdependence of the modules will be compared. Considering the priorities, the module combinations with the lowest duration will be picked.
6. Changes concerning the start and end time will be made, if
    a. the interdependence of two or more modules enforce it.
    b. the resource of one module in task chain is used in the next task chain of the same module and the buffer time does not suffice.
7. If the time frame is exceeded a message will be shown in the specified row.
8. Create a list of the synchronized modules.

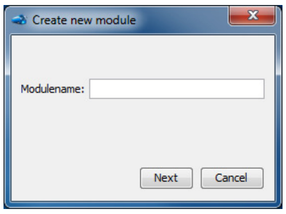
6.3 Example

An example in the airport context will now show the creation of the modules and scenarios but also the result of the optimization by using the prototype components and diagrams. Material data for the modules was caught from the work in [3].

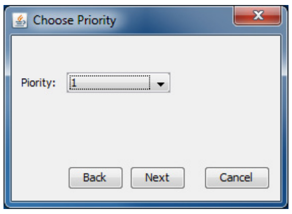
The prototype contains a wizard for each step to create a module. This is mapped in figure 1. In the first step (Figure 1a) the user has to enter the module name. The next step (Figure 1b) there has to be chosen the priority of the module. As mentioned a lower number is a higher priority. The steps after (Figure 1c-e) are caused by the test environment of the prototype. First the data base has to be chosen and then the table and at the end required values like: common dimension, resource and start and end time.

For example there have been created four modules by a user with the following priority:

- Passenger transport with the priority one
- Luggage transport with the priority one
- Catering with the priority two
- Refuelling with the priority three

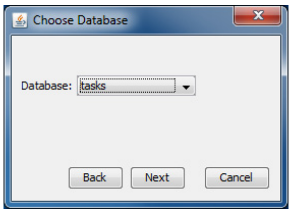


a) module name



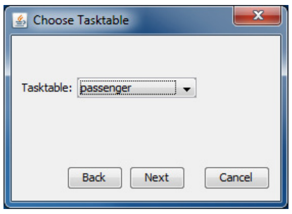
Piority: 1

b) priority



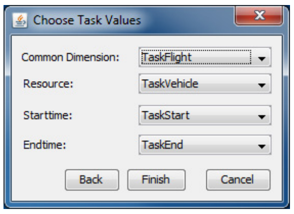
Database: tasks

c) data base



Tasktable: passenger

d) task table



Common Dimension: TaskFlight
Resource: TaskVehicle
Starttime: TaskStart
Endtime: TaskEnd

e) task values

Fig. 1 Module Wizard

Figure 2 shows the constellation before the optimization for one common dimension.

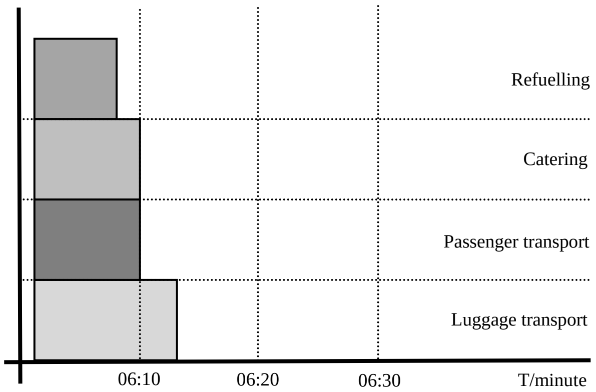


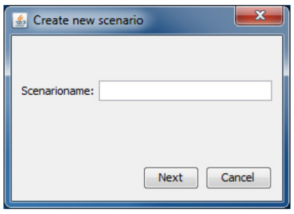
Fig. 2 Modules before optimization

```
CreatedModule = passenger
transport
Priority = 1
DB = Passenger
Table = tasks
CommonDim = TaskFlight
Resource = TaskVehicle
TaskStart = TaskStart
TaskEnd = TaskEnd
```

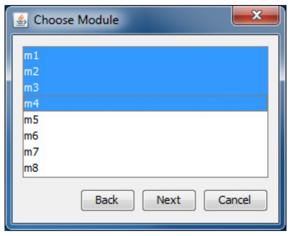
Fig. 3 Module data file for passengers

For each module a data file with the mentioned requirements will be created. This is important because of the reusability of a module. A module file for the passenger transport is shown exemplarily in figure 3.

For the scenario creation the prototype contains a scenario wizard shown in Figure 4. There are four steps to take.

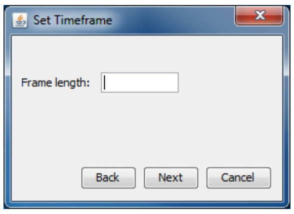


a) scenario name

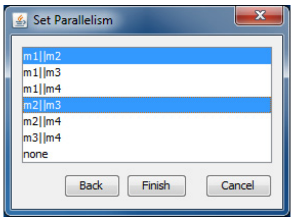


m1
m2
m3
m4
m5
m6
m7
m8

b) choose the modules



c) timeframe



m1||m2
m1||m3
m1||m4
m2||m3
m2||m4
m3||m4
none

d) timeframe

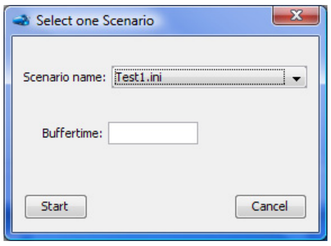
Fig. 4 Scenario Wizard

First of all the name has to be entered (Figure 4a). Then modules have to be chosen and the time frame of the scenario has to be set (Figure 4b/c). Finally the interdependence of the modules has to be chosen (Figure 4d). Figure 5 shows a scenario data file for the current example. This file contains all data for execution. “notMod_Combo” will be created automatically and describes the not chosen module combinations. In this example the passenger transport and luggage transport can be handled together. Furthermore the luggage transport and catering can be handled in parallel. The other combinations are not allowed. The catering cannot be done during the passenger transport. Because of the security the refuelling has to be done after all other tasks. The start window in the prototype allows the start of the created scenarios shown in figure 6. The buffer time is set as a buffer between the task chains. This can be described by using the output table of table 1.

```
chosen_modules = passenger, luggage,
catering, refuelling
time_frame = 45
module_combo = passenger||luggage,
luggage||catering
notMod_combo = passenger||catering,
passenger||refuelling,
luggage||refuelling, catering||refuelling
```

Fig. 5 Scenario data file

If the used resource is the same, at least the buffer time has to be granted between the task “end time” of a module and the task "start time" of the same module in the next row.



Scenario name: Test1.ini

Buffertime:

Fig. 6 Start window

The algorithm will now pre-sort the modules by their priority. All possible combination will be built depending on the interdependence. The time duration of these combinations will be compared and the shortest one will be chosen. Figure 7 shows the constellation after the optimization for the first task chain. The optimization is now done the prototype shows the optimized modules in a simple table.

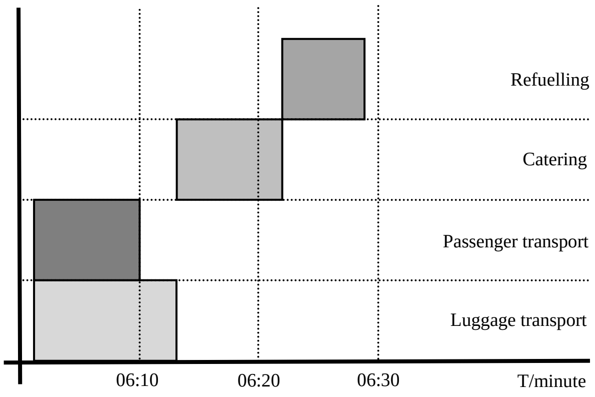


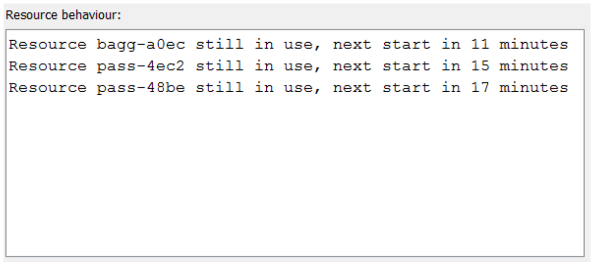
Fig. 7 Modules after optimization

This is exemplarily shown in table 1. Certainly the other rows contain different task chains for the following task. The prototype has a resource check functionality to simulate the interface to a local optimizer. It compares the resources of one module with the resource of the same module in the next task chain. If the same resource was used and the set buffer time passes over the time difference of the two task chains, the start time has to be changed. In a real environment the local optimizer has to change the resource of the following task chain.

Table 1 Output of main table

CommonID	luggage	passenger	catering	refuelling
D68D6B13	06:01-06:13	06:01 - 06:10	06:13 - 06:22	06:22 - 06:29
A40326B4	06:23 - 06:35	06:23 - 06:32	06:35 - 06:54	06:54 - 07:01
94CF5424	06:13 - 06:28	06:13 - 06:20	06:28 - 06:48	06:48 - 06:54
4863A15B	07:32 - 07:41	07:32 - 07:40	07:41 - 07:58	07:58 - 08:04

Figure 8 shows another part of the main window with the resource time change information.



Resource behaviour:

```
Resource bagg-a0ec still in use, next start in 11 minutes
Resource pass-4ec2 still in use, next start in 15 minutes
Resource pass-48be still in use, next start in 17 minutes
```

Fig. 8 Resource behaviour

7 Results

The result of this contribution is the optimization of any number of local optimized workflows. Using a global optimizer would lead to the following situation: after each optimization of the local instances the global instance has to be started and vice versa. This linking results a high rate of computation and communication. To concretize this in the airport context it will be exemplarily described. The experiences of work in the airport Hamburg result in the statement that each workflow has about 20 planned tasks and our example uses 4 modules. A system that forces each local optimizer after the synchronisation of a task to optimization has to handle 100 optimization procedures in a very short time (4 times 20 and adding 20 executions of the global optimizer; see also equation 1). The implemented prototype handles only 5 optimization procedures. Figure 9 shows the number of calculation steps depending on the number of modules and task chains. Equation (1) shows how to receive the calculation.

$$y = x(z + 1) \quad (1)$$

The main target of this contribution was to implement an approach that can be used in different area. Therefore there might be situations with for example 1000 task chains and by using ten modules 11000 calculation steps are required. This has to be handled by the local optimizers and global optimizer. With an exaggerated duration of one second for each step, the procedure would need three hours.

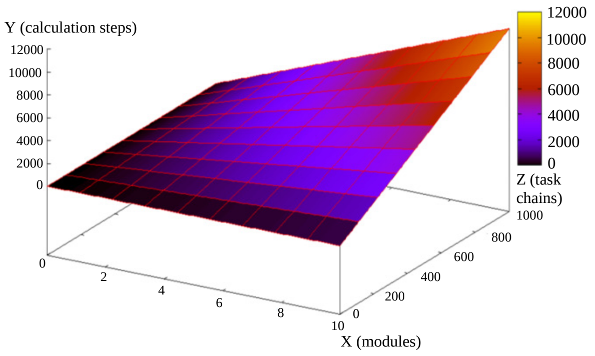


Fig. 9 Optimization by higher utilization

8 Conclusion and Outlook

The concept of the global workflow manager is to make optimization of interdependent workflows possible in different areas of commerce. The procedure of the optimization was described for the concept and the procedure. The prototype uses the example of the airport Hamburg and gives an insight of the concept. The prototype can be adopted in each area with the mentioned requirements. This concept could be established in logistic contexts to optimize interdependent workflows. Our result shows the problems.

The next step would be to build a connection to real existing workflows. It would be desirable to establish this system in the airport Hamburg and build a connection to the existing local optimizers. Because of liberalisation of the airport Hamburg, this is regrettably not possible. This means that the subcontractors, who are working for the airport, do not want to open their interface for higher ranking managing software. The reasons are as mentioned data security, but also to save the company security. To establish such a system, the structure at the airports has to change. This is a decision that has to be made by the airport operators. But there are other interesting logistic areas in which this system would be useful: harbours or hubs of logistic companies.

The upcoming goals are the research of solid data, the adaption in other areas and the technical improvement of the prototype. In the last section the problem of this concept was discussed. This could be solved by extensive simulation. The runtime of the global optimizer and its local optimizers could be proved. The adaptation in other logistic area would validate the concept. The GUI based module and scenario editing would improve the usability. The use of a bar chart for the output like a histogram would improve the visualisation.

9 Acknowledgments

This work is financially supported in part by the BMBF via the Cluster of Excellence in Aviation Hamburg project L3 and the University of Hamburg. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding bodies.

10 References

[1] Y. Farschtschi and M. Widemann. diploma thesis. *Modulares Workflow Management und Optimierungssystem am Beispiel des ground handling am Flughafen.* Hamburg : s.n., 2009.

[2] Y. Farschtschi, K. Himstedt, J. Wittmann and D. P. F. Möller et al. Macroscopic modelling of passenger streams on the airport and its adaptation in matlab simulink. *EUROSIM 2010.* Mai 2010.

[3] M. Widemann, Y. Farschtschi, J. Wittmann and D. P. F. Möller Workflow management of the ground handling at the airport through modular system optimizing. *EUROSIM 2010.* Mai 2010.

[4] GROUNDSTAR Ltd. [Online] 2010. http://www.groundstars.de/e_index.html.

[5] D. Hoffman and D. Weiss. Software fundamentals: collected papers by David L. Parnas. s.l. : Addison-Wesley, 2001, pp. 9-28.

[6] R. H. Güting and S. Dieker. *Datenstrukturen und Algorithmen.* 2004. 978-3519221210.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. *Introducing to Algorithms(2nd ed.).* 2001. 0-262-03293-7.