Master's degree thesis

LOG950 Logistics

An Adaptive Large Neighborhood Search Heuristic for Periodic Supply Vessel Planning Problem

Almiashev Rushan

Number of pages including this page: 70

Molde, 24.05.2016

Molde University College
Specialized University in Logistics

Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. Failure to complete the statement does not excuse students from their responsibility.

*Please complete the mandatory statement by placing a mark **in each box** for statements 1-6 below.*		
1.	**I/we hereby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than mentioned in the paper/assignment.**	☒
2.	**I/we hereby declare that this paper** 1. Has not been used in any other exam at another department/university/university college 2. Is not referring to the work of others without acknowledgement 3. Is not referring to my/our previous work without acknowledgement 4. Has acknowledged all sources of literature in the text and in the list of references 5. Is not a copy, duplicate or transcript of other work	Mark each box: 1. ☒ 2. ☒ 3. ☒ 4. ☒ 5. ☒
3.	I am/we are aware **that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the** **Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8** **and** Examination regulations section 14 and 15.	☒
4.	**I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check**	☒
5.	**I am/we are aware that Molde University College will handle all cases of suspected cheating according to prevailing guidelines.**	☒
6.	**I/we are aware of the University College's rules and regulation for using sources**	☒

Publication agreement

ECTS credits: 30

Supervisor: Irina Gribkovskaia

Abstract

In the upstream offshore petroleum logistics supply vessels play the most important role, being the largest cost contributor. For this reason careful supply vessel planning is of vital importance for this industry. In the literature this problem is known as Periodic Supply Vessel Planning Problem (PSVPP). The problem involves determination of the fleet composition, vessels schedules and voyages. For large size instances optimal solutions are unachievable and for this reason we developed meta-heuristic algorithm. For heuristic validation we developed a two-phase approach which provides optimal solutions for small and medium size instances. Experiments show that developed metaheuristic algorithm provides optimal and near optimal solutions within short times.

Keywords: offshore logistics; periodic routing; adaptive large neighborhood search; fleet composition; routing and scheduling; adaptiveness.

Acknowledgment

First of all, I’d like to express gratitude to my supervisor Irina Gribkovskaia for the provided problem that was studied in this thesis and for the data that was used in the experiments. As well I want to thank PhD student Yauheni Kisialiou who taught me combinatorial optimization methods used in this thesis and supplied with ideas during the development phase.

I am also thankful to Norwegian government which provide me with financial support under the quota scheme.

I want also thank the Norwegian government for providing me financial support under the Quota Scheme so that it became possible to study in Norway. In addition I would like to express special gratitude to my teacher from Moscow Dmitriev M. G. and who provided good recommendations to Molde University College authorities.

My special thanks to my classmate Lena Marder, who encouraged me on hard working during the whole study process.

Finally, I am very grateful to my family and my friends for their moral support and believing in me.

Contents

List of figure

List of table

1.0 Introduction

Many oil and gas producers operate offshore installations which require regular supplies of commodities from land. For this purpose, oil companies hire special supply vessels, which deliver cargo to and collect from installations. However, this resource is rather costly both in terms of hiring cost and in terms of operating costs (fuel costs). Therefore, it is of vital importance to define the fleet required to provide regular supply of offshore installations and achieve efficient utilization of all platform supply vessels (PSVs).

In offshore petroleum logistics, supply vessel planning represents an actual problem. The problem considered in this thesis comes from a real-world problem faced by Norwegian oil and gas company Statoil ASA. The oil field located in the North Sea and serviced from the onshore supply base located in Mongstad, is selected as the one for which delivery problem is studied. The planning of deliveries is performed on the tactical level with one week planning horizon. Each installation has a requirement for the number of visits during the week. A weekly sailing plan is used repetitively over several months and then subject to revision. Construction of the sailing plan requires consideration of the following problems: allocation of voyages (a voyage represents a certain sequence of platforms served by a vessel) to vessels, sequencing of voyages (routing problem) and definition of departure times (scheduling problem) for each vessel on its voyages during the week. All these sub-problems should be solved so that the total sailing cost and vessels charter cost are minimized. The problem relates to the class of Vehicle Routing Problems and in the context of its practical aspects, in the literature, is referred to as Periodic Supply Vessel Planning Problem (PSVPP). The term “periodic” means that the problem is solved for a certain planning horizon. Each of the above mentioned sub-problems (packing, routing and scheduling) represents NP-hard combinatorial problem. Problems of a large size cannot be solved by exact methods within a reasonable time. The problem we consider involves planning deliveries for 26 installations during one week, where each installation requires from 1 to 5 visits. Therefore, development of some efficient heuristic approach is required to obtain a good solution within a reasonable time.

In this master thesis we develop an Adaptive Large Neighbourhood Search Heuristic for PSVPP that might be used as a decision support tool by delivery planers at Statoil ASA for organizing of an efficient supply of offshore installations. For validation of the heuristic, we develop two-phase exact approach providing optimal solutions for small and medium size instances within reasonable time.

In chapter 2, we provide description of the studied problem, its main characteristics, constraints and the objective. In chapter 3 we provide the review of the literature relevant to the problem and compare the studied problem to those similar found in the literature. In chapter 4 we review solution methods which are used for similar problems and analyse methodology which could help to solve our problem. In chapter 5 we state the objective of this master thesis which relates to development of the algorithm, its validation and results analysis. In chapter 6 we present developed solution approaches with detailed description of theirs logic. In chapter 7 we perform fine tuning of the developed heuristic approach, described test instances and provide analysis of the conducted experiments. In chapter 8 we summarize the work conducted in this thesis and provide some directions for the futures research. Finally, list of references and tables with complete results of the experiments accomplish the master thesis.

2.0 Project description

Research that we conduct in this thesis is dedicated to real-life problem faced by Statoil ASA. Statoil ASA employs more than 22000 employees, and it's total revenue is more than 600 billions, being the largest oil and gas operator in Norway. Furthermore, Statoil ASA operates in more than 30 countries around the world and is number one in offshore oil and gas extraction in terms of technologies, effectiveness and efficiency (Statoil 2016).

In this master thesis, we study the problem of suppling offshore oil and gas installations, encountered by Statoil on the Norwegian continental shelf. We focus our research on a single supply base located in Mongstad, from which installations, assigned to this base, receive all the necessary materials and equipment. Supply of offshore installations is provided by a special fleet of supply vessels. However, hiring cost of a supply vessel is very expensive and therefore care should be taken when defining the fleet size. In addition to vessels' charter cost, the cost of supply includes vessels fuel costs. As follows, optimal sequences of visiting installations are required to minimize such costs. Furthermore, downtime cost of an offshore installation is enormous and should be avoided. Hence, Statoil strives to construct efficient schedule for suppling offshore installations with minimal costs, while not allowing for the downtime.

The problem of supplying offshore installation, which we study in this thesis, is known as Periodic Supply Vessels Planning Problem (PSVPP). Efficient solution to this problem can reduce the logistical costs drastically and in the same time ensure high service level.

Further, in this section we provide the main elements, characteristics and constraints inherent to this problem. In addition, in the end of the section we provide an example of a vessels schedule.

2.1 Periodic Supply Vessel Planning

The supply vessel planning problem involves identification of the optimal fleet composition, necessary to serve a given set of offshore installations from a single onshore supply depot, and at the same time development of schedules and routes for vessels, so that vessels charter and fuel costs are minimized. In this problem, under routes are understood the voyages, starting and ending at the depot, and sailed by a particular vessel during the planning horizon. Each voyage, in turn, is defined by a set of installations in a certain sequence of visiting them. Each voyage has a specific departure time from the depot. A vessel's schedule is than defined as collection of voyages departing from a supply base at specific times. The objective of PSVPP is to construct a least cost schedule for a fleet of supply vessels, for a given planning horizon.

The studied problem is of a tactical level, where the planning horizon is considered to be one week. Such schedule is repeated for several months and revised with the aim of adaptation to some changes. Such changes involve changes: in demand of installations, incorporation of a new installation into a schedule, changes in the required number of visits for some installations during the week, time windows, etc.

More properly, description of the main problem’s elements and constraints is provided as follows.

2.1.1 Supply Base

The onshore supply base has opening hours from (8:00–18:30), during which loading and unloading operations are performed. In addition, personal availability and limited number of berths set a limit on the number of vessels that can be served during a day. The turnaround of a vessel on the base i.e. the time required for loading and unloading operations is assumed to be 8 hours. There is a specific set of possible departure times for vessels on their voyages. The reason for haven this such departure time options is twofold. On the one hand adjustment of departure time for a voyage may lead to cost reduction in case of installation(s) with time windows on this voyage. The waiting time in this case may be reduced and as follows the cost of a voyage. On the other hand such “flexible departures” allow to exploit more efficiently work force by avoiding performance of the same operations for different vessels in parallel. In our case, the set of possible departure times is 16:00, 17:00 and 18:30.

2.1.2 Voyages

Voyages are defined as a sequence of installations to be visited by a particular vessel. Each voyage starts and ends at the depot and has specific departure time The maximum voyage duration is set to three days or 72 hours (counting the time for loading/unloading in the base), which is explained by maximum lead time requirements. In addition, there is a requirement to the minimum and maximum number of installations per voyage, 1 and 7 respectively. A vessel’s schedule should be constructed so that the voyages it sails are not overlapped in time.

2.1.3 Offshore installations

Offshore installations play the main role for oil and gas production. Each installation has specific visit frequency during the week i.e. the number of visits it should receive from supply base. There are two types of offshore installations. The first type performs drilling operations and is characterized by larger and more variable demand, and higher visits frequency. The second type is represented by platforms performing oil and gas extraction. This type is characterize by relatively stable demand and low visits frequency. The weekly demand of a platform is assumed to be evenly distributed between visits. Furthermore, those installations having more than one visit per week require even spread of departures to them. For example, installations with three visits per week should be assigned to voyages in such a way that departure to these installations is performed at least 1 time during 3 days. Such requirements are set to each visit frequency. Therefore, it's not an easy task to assign installations with different visit frequencies to voyages while maintaining even spread of departures. Commonly in periodic routing problem planners are concerned by even spread of visits to customers rather than spread of departures. In PSVPP requirement to even spread of departures is explained by the fact that installations know when it is the latest to submit demand request before a vessel starts a voyage. Taking is to account that the maximum lead time is assumed to be three days, such system proves to be quite convenient.

Furthermore, all offshore installations are divided in to two categories: with possibility for night service and without it (with time window and without it). For installations without time window, a vessel may come for service at any time during the day. However, for offshore installations, which cannot be serviced at night time (19:00 – 7:00), there are several situations are possible. If a vessel arrives to offshore installation after closing time (and of course before its opening time), then it must wait till the opening time. As well, there may be a situation when a vessel arrives within the time window, but the time required to perform service before installation is closed, is not enough. In this case, the vessel should wait till the next opening time.

2.1.4 Supply vessel

A fleet of supply vessels is performs delivery of equipment and materials to installations and collection of used. Each platform supply vessel (PSV) may have its own sailing speed and different deck capacity. This means that some PSV are unable to sail some voyages. The cost of PSV is composed of two types of costs: vessel weekly charter cost and fuel cost. Fuel cost is a variable cost and depends on the vessel's speed and type of operation performed. There are different fuel consumption rates for loading/unloading operations at the base, during sailing and during loading/unloading at an installation.

2.1.5 Weekly sailing plan

The weekly sailing plan is composed of weekly sailing plan of all vessels. On the example below PSV1, PSV2 and PSV3 (**Error! Not a valid bookmark self-reference.**). Weekly sailing plan of a vessel is defined by a set of voyages, consequently assigned to specific departure times and not overlapping in time. Each voyage starts and ends at the supply base (FMO). If a vessel, for example, is supposed to start its voyage at 17:00 then, taking into account turnaround time, it is assumed start loading operations at 9:00. Therefore, it should have come back from previous voyage (if any) before 9:00. From the perspective of each PSV collection of voyages of a vessel represents vessel’s route. For example, the route for PSV2 involve 3 voyages, starting on Tuesday, Thursday and Saturday. Each voyage in turn represents a consecutive collection of installations, starting from depot. For example first voyage of PSV 1 starts at 16:00 on Monday from FMO and visits installations TRO, COI, OSE in the given sequence. On the schedule below the service at supply base is marked green, sailing times are marked yellow and service it installations mark dark blue. There are situations called “end-of-week” effect when a voyage starts at the end of the week and finishes on the next week. For example the third voyage of PSV2 starts on Saturday, serve installation COI on Sunday, and installations OSE, KVB on Monday.

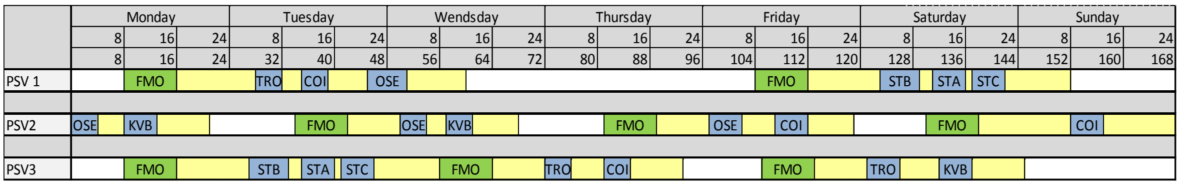


Figure 1 An example of Weekly sailing plan.

There is one thing, on which we have to stress our attention. As it was mentioned above, the planning horizon for installations is assumed to be one week. Nevertheless, the planning horizon for vessels is extended up to two weeks. Such planning is explained by dealing with a specific situation that may happen, called “end-of-week” effect. On the Figure 2 is described the situation when the last voyage of PSV2 starts on Saturday and ends on Monday, while its first voyage starts on Monday. As we see, voyages of the same vessel are overlapped in time, which is not allowed. To circumvent such situation vessels PSV2 and PSV3 may swap voyages on the second week. The only condition for possibility of swapping is that the first voyage of PSV3 should start later than the end of the last voyage of PSV2. In this case, PSV2 and PSV3 exchange by voyages on each week. Such approach may be viewed as the relaxation of the voyage overlap constraint for each vessel in case of the "end-of-week" effect, that may lead to the cost and even fleet size reduction.

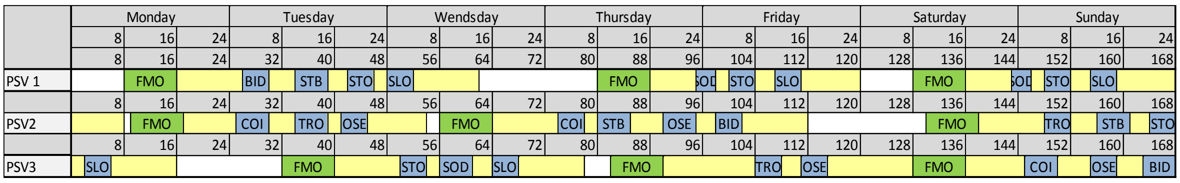


Figure 2 An example of weekly schedule with coupled vessels

2.2 Objective

The objective of the PSVPP is to construct a weekly sailing plan with minimal total vessels charter and sailing cost taking into account base capacity constraint, requirements for the even spread of departures, visit frequencies, vessels capacities, maximum voyage duration and maximum number of installations per voyage, and voyage overlap constraint.

Taking into account problem dimensions and the size we may conclude that some optimization-based design support tool is required which is able to provide the solution of a good quality in a short time. The output of this tool should be weekly vessels schedule. Below are summarized the main characteristics of the problem:

- The planning horizon of 7 working days (one week).
- Set of installations to be served and supply base, and theirs coordinates.
- Number of visits for each installation per week.
- Weekly demand for each installation.
- Average service time of each installation, required for loading/unloading operation.
- Fuel consumption rate of a vessel when sailing, servicing at installation and servicing at supply base.
- Fuel cost per ton.
- Set of available departure times during a day.

The objective of the problem is to:

- Construct weekly delivery schedule for the given set of installations.
- Find the fleet size and build voyages for each PSV.
- Find departure time of each voyage.

- Minimize total vessel charter and fuel cost.

Constraints on solution:
Supply base constraints

- Maximum number of vessel that can departure during a day.
- Supply base working hours.
- A vessel turnaround time.

Voyage constraints

- Maximum number of installations on a voyage.
- Maximum voyage duration.

Vessel constraints

- Capacity of a vessel.

Offshore installations constraints

- Departures to each installation should be evenly spread during the week.
- Working hours of each installation during a day.

3.0 Literature review

The literature dedicated to PSVPP is relatively scarce. Bellow we provide some papers on the PSVPP topic, which appear in the literature for the last several decades. PSVPP relates to vehicle routing problem (VRP) type and namely to periodic VRP, where routes should be constructed for a planning horizon. The main different PSVPP from periodic VRP is that PSVPP routes (or voyages) last more than one day. The literature on the PSVPP topic is as follows.

Fagerholt and Lindstad (2000) presented a paper dedicated to "Optimal policies for maintaining a supply service in the Norwegian Sea". The authors develop two-phase exact method to solve the problem. On the first stage, a feasible set of candidate voyage is generated for each vessel. On the second stage, the set generated of voyages is used as an input to the integer optimization model. In the literature, the model is known as set partition model. However, the authors formulated a simplified version of PSVPP, ignoring constraint on the spread of departures. Their approach does not provide vessel schedule and just handles the problem of assignment voyages to vessel.

Gribkovskaia, Laporte, and Shlopak (2008) represented "A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms." The authors study pickup and delivery problem encounters upstream offshore supply in the Norwegian Sea. The problem is operational planning. A single vessel should provide pickup and deliveries of cargo from several offshore installations. The authors develop tabu search for a single base.

Halvorsen-Weare et al. (2012) considered "Optimal fleet composition and periodic routing of offshore supply vessels", where the authors took into consideration the aspects omitted by the Fagerholt and Lindstad (2000). In this article, the authors as well use the same two-phase exact method to obtain optimal solution accounting spread of departures constraints. In addition, authors deal of the weather uncertain that may result in delays of vessel on theirs voyages. Authors handle uncertainty by adding a slack in the end of a voyage. Developed approach may be applied to medium and large size instances.

Korsvik and Fagerholt (2010) considered a problem in a shipping trade dealing with transportation of bulk products. Shipping companies derive some additional income for optional spot contracts. The authors developed an efficient tabu-search algorithm as a decision support tool, ensuring quick decisions for the planners. The output of their tool represents a schedule with a minimal fleet of vessels required to perform a certain task.

Shyshou et al. (2012) proposed a Large Neighbourhood Search (LNS) heuristic with the aim to solve large instances of PSVPP, taking into account all the constraints handled Halvorsen-Weare et al. (2012). Heuristics is able to define optimal or near-optimal solutions for small-medium size problem and as well is able to construct schedule for large size problem within a reasonable time.

As well, there are several papers dealing with weather uncertainty in PSVPP and as well, those taking into account environmental issues. Halvorsen-Weare and Fagerholt (2011) developed three-phase approach able to define optimal solution of the problem by introducing robust measure to voyages. Norlund and Gribkovskaia (2013) considered the problem of minimization of supply vessel emissions by optimizing using the same, mentioned above, two-phase approach.

As we see, there is only one approach to PSVPP that deals with the problem similar to ours in terms of the problem type and instance size. Namely the approach by Shyshou et al. (2012). This solution approach turned out to be quit efficient for small and medium problem sizes, although the running time for large instances is relatively high. As well, there are several differences between the problem formulated in our thesis and the problem formulated by Shyshou et al. (2012). In Shyshou et al. (2012) vessels departures on voyages are single and fix during a day, while in our problem a vessel have a set of departure options manually defined by a planner. In addition, planning horizon for vessel in our problem is extended to two weeks, with the aim of handling "end-of-week" effect issue.

As we see the LNS heuristic by Shyshou et al. (2012) was quite successful for the PSVPP. Therefore, we may use this heuristic, as a starting point for the development of ours own algorithm. In the next section, we consider methodology related to LNS and some known approaches for improvement of its efficiency.

4.0 Methodology

In this section we cover solution approaches developed for PSVPP and study some methodology that maybe useful for our future metaheuristic.

Halvorsen-Weare et al. (2012) presented two-phase mathematical modelling based approach. On first stage, the authors generate all possible sets of installations satisfied minimum and maximum requirements for the voyage size and capacity. These sets are generated out of the set I of all installations. Then for each set TSP with time windows is solved. The output of the algorithm is the set R of all candidate voyages a vessel may sail, where each voyage represent the optimal permutation for the corresponding set of installations. As well, based on the generated voyages binary parameter $a_{i,r}$ is created. $a_{i,r}$ is equal 1 if installation i is on the voyage r, $i \in I, r \in R$. On the second stage the output of the voyage generation algorithm is used as the set-covering voyage-based model. The model selects a set of voyages out of the whole set R of available voyages taking into account spread of departures, overlap constraints and based capacity constraint. Each voyage is assigned specific departure day. The output of the model is a weekly vessel schedule.

Advantage of this approach compared to the full enumeration is that mathematical model uses only feasible and shortest voyages (provided by the voyage generation algorithm) and thus reducing computational time of the second phase. The solution achieved on the second stage is optimal.

Since we agreed above to develop an algorithm using the Large Neighbourhood Search methodology, we first consider its main principles.

LNS heuristic was first presented by Shaw in (Shaw 1997) and (Shaw 1998). Heuristic was applied to VRPTW (Vehicle Routing Problem with Time Window) and showed very good results. Below we briefly consider the main idea of the LNS. The pseudocode for LNS is as follows (Procedure 3.1).

```
Procedure 3.1 Basic LNS heuristic
1: Function LNS ($s \in \{solutions\}, q \in \mathbb{N}$)
2:     solution $s_{best} = s$;
3:     do
4:             $s' = s$;
5:             remove $q$ request from $s'$
6:             reinsert removed requests into $s'$
7:             if $(f(s') < f(s_{best}))$ then
8:                     $s_{best} = s'$ ;
9:             end if
10:            if $accept(s', s)$ then
11:                    $s = s'$;
12:            end if
13:    while stop-criterion met
14: return $s_{best}$ ;
```

The algorithm starts with the generation of a random initial solution, which is supposed to be further improved. Then, "destroy" operator removes q number of visits (requests) from solution s'. Further, repair operator inserts removed visits (requests) back into solution. If the new solution s' is the better than the best found solution s_{best}, then the solution s' is set as the best s_{best}. Otherwise, if the solution s' is worse than solution s, then solution s' is accepted subject to some accepted criteria (user defined), the algorithm proceeds to the next iteration and so on until stop-criteria is met.

As it was mentioned in the previous section, Shyshou et al. (2012) developed Large Neighbourhood Search heuristic for PSVPP. The algorithm is run for a given number of restarts and iterations. At each restart an initial (feasible) solution is randomly generated. Then, for the given number of iterations an attempt is made to improve this initial solution. At the beginning of each iteration, we make a move from the current solution (at the first iteration the current solution is the initial solution) to one of in its neighbourhood. This is done with the use of *destroy* and *repair* operators. Destroy operator randomly selects a certain number of voyages (user defined) and then in each voyage a random number of visits is removed and placed into pool S of uninserted visits. After that, repair operator tries to insert visits, contained in the pool S, back into the schedule. If the attempt is successful i.e. all visits are inserted back and all constraints are satisfied, then the algorithm proceeds to the improvement phase. On the improvement phase, the solution obtained after a move is tried to be improved by subsequently applying several *improvement* procedures. The first procedure tries to reduce number of voyages in the schedule and thus making the idle time of vessels larger. The second procedure tries to reduce total duration of all voyages by relocation visits between them. The relocation of visit from one voyage to another is accepted if the net duration of both voyages is reduced by at least one day. The third procedure represent a *deep greedy* algorithm, which tries to relocate visits in the schedule while the cost of the schedule reduces. As well, after each of the described above procedures an attempt is made to reduce the fleet size by relocation of voyages between vessels including those ones, which are not in the schedule. The set of improvement procedures is applied in the loop in the predetermined sequence while the cost of the schedule reduces.

After the improvement phase the algorithm attempts to reduce the fleet size again. If the number of vessels in the solution is above the predefined lower bound (the lower bound is defined before the initial solution is created) than the algorithm defines the vessel out of which it is possible to relocate the largest number of visits to other vessels. Those visits, which were not relocated from this vessel, are placed into pool *S* of uninserted visits. If the number of voyages turns out to be lesser than the predefined lower bound then the flag indicating the state of the schedule in terms of the number of voyages relative the lower bound on the number of voyages, is set to “true”. The algorithm proceeds to the next iteration. At the beginning of each iteration, after application of destroy operator, the algorithm cheeks whether the flag was set to “true” at the previous iteration. If the flag is set to “true” then the algorithm creates empty voyages to ensure feasibility of the schedule after the repair operator is applied. And the algorithm proceeds to the improvement phase as described above.

The algorithm proved to be quite efficient on small and medium size instances showing optimal and near optimal results. As regards larger instances, efficiency of the algorithm is not proved due to the absence of the optimal solutions. Solutions for large instances were obtained within high computational times.

Ropke and Pisinger (2006) presented “An Adaptive Large Neighbourhood Search Heuristic for the Pickup and Delivery Problem with Time Windows”. In this article, the authors supplemented the LNS by a new approach to search neighbourhood solutions. Developed heuristic was called adaptive LNS. This approach implies several destroy and repair operators which paschal destroy and then repair the solution that is called “a move”. The main idea of “adaptiveness” is to keep track on the performance on destroy and repair operators. Since there are several destroy and repair operators and only one of each type should be selected at the beginning of an iteration, destroy and repair operators are assigned corresponding probabilities according to which they are supposed to be selected are equal and updated after certain number of iterations (segment). At each iteration selected destroy and repair operators receive certain rates based on their performance. These weight are accumulated within a segment. And in the end of each segment each operator receive the total score and probabilities are updated. Those operators, with good performance and, of course, with higher score are assigned higher probabilities for the next segment of iterations. Thus, the algorithm tries to adapt to better search of neighbourhoods based on the performance of destroy and repair operators within last segment of iterations.

Furthermore, the authors propose an approach to avoid trapping into local minimum. To do this, the authors take the idea from simulated annealing. The idea is to accept solution s' with some probability, if it is worse than the solution s with some probability. Probability is define by $e^{-\frac{f(s')-f(s)}{T}}$, where the temperature $T > 0$ and $0 < c < 1$ is the cooling rate. In each iteration temperature decreases by multiplication by c .So the lower the value of c, the higher the cooling rate and probability of acceptance of worse solution reduces faster as the algorithm proceeds for one iteration to another.

5.0 Research Task

The primary goal of this thesis is to develop an Adaptive Large Neighbourhoods heuristic (ALNS) for the described above PSVPP taking into account all the problem's specific constraints, which is able to solve large size problem instances within a relatively short time. Since we need to validate the algorithm and test its efficiency, we need to compare it to some exact approach. For this purpose we develop two-phase exact solution approach based on the approach by Halvorsen-Weare et al. (2012) and adapted to the specifics of our problem (flexible departures and coupled vessels).

6.0 Solution approach

In this section, we provide the description of our modification of two-phase exact approach and detailed description of developed ALNS heuristic.

6.1 Two-phase method

6.1.1 Voyage generation algorithm

The main idea of the voyage-generation algorithm is based on dynamic programing or the concept of recursive call. The input information involves sets of offshore installations and supply vessels, coordinates of offshore installations, vessels capacities and maximum voyage duration. The algorithm starts from generation of all sets (combinations) of installations. The number of sets is denoted by K and set $N(k)$ represents the collection of installations in a set k, $k \in 1..K$. The size of the sets is limited by minimum and maximum number of installations in a voyage and vessel capacity (Procedure 6.1.1 Line 5). Then, for each combination $N(k)$ the Travel Salesman Problem with Time Window (TSPTW) is solved by dynamic programming (Procedure 6.1.2) i.e. for each set $N(k)$ the shortest permutation is defined.

Procedure 6.1.2 takes one by one all sets of installations from the set $N(k)$ as an input. Each installation in a set $N(k)$ is removed and sent into recursive procedure. In this procedure this installation is added into a voyage which is defined globally in the procedure (initially this voyage is empty and contains only the depot). After the installation is added to the voyage, duration and sailing of this *partial voyage* (since it does not contain all installation) are calculated. And then, for this partial voyage procedure 6.1.2 is called again (recursively). After the partial voyage is sent into recursive procedure, installation is removed from this partial voyage and is placed back into set $N(k)$. For the partial voyage which was send into recursive procedure (and which contains this installation) all this steps are repeated again starting from line 5. When at some recursive call into it turns out that all installations from the set N(k) present in the voyage then duration of the voyage with this sequence is compared to the previously found voyage with the shortest duration. If the duration of the voyage with the new sequence is shorter than the previously best found, then this voyage is stored and set as the best. So, this recursive procedure dynamically enumerates all possible permutations of installations for each set $N(k)$ and returns the shortest. The output of the algorithm is the set of shortest voyages for each vessel.

Procedure 6.1.1 Voyage-generation algorithm. Main code

```
1:Function VoygeGeneration(Vessels $V$, Departure Time $T$, Installations $I$W)
2:    Array $R = \emptyset$: array, which containing all feasible routes;
3:    for each $j$ in $V$
4:          for each $t$ in $T$
5:                do
6:                      Construct unique combination of installations,
                        which satisfy constraints for vessel capacity and
                        maximum and minimum number of visits during one
                        voyage. And it places on set $N(k)$;
7:                      Solve TSPTW for set $N(k)$ by Procedure 6.1.2;
8:                      if solution founded
9:                            place solution on set $R$;
10:                     end if
11:               while(!all combination of Installations not created)
12:         end for
13:   end for
14:return $R$;
```

Procedure 6.1.2. Recursive solve TSPTW.

```
1: Route: $R(k) =$ Globally defined  route;
2: Route $BR(k)$ = Globally defined best route;
3: $Bcost =$ Globally define Best cost;
4: Function SolveTSPTW (PoolOfInstallations $N(k)$, MaximumDuration $maxDur$)
5:     for $i$ = 0 to $N(k)$
6:          Installation: $C = N(k)_i$;
7:          set $R = R \cup C$;
8:          set $N(k) = N(k) \backslash C$;
9:          $cost =$ Compute sailing cost for route $R$;
10          $dur =$ compute route $R$ duration;
11:         if($cost < Bcost$ and $dur < maxDur$)
12:               if($|N|$ == 0)
13:                     $Bcost$ = $cost$;
14:                     $BR$ = R;
15:               end if
16:               else
17:                     Recursively run SolveTSPTW ($N(k), maxDur$);
18:               end If
19:         end if
20:         set $R = R \backslash C$;
21:         set $N(k) = N(k) \cup C$;
22:   end for
23: End Function;
```

6.1.2 Voyage-based model

In this section, we provide mathematical formulation of the PSVPP. We use as the input the set of voyages from the voyage generation algorithm, which was described above, and their corresponding costs. As well based on the set of generated voyages we define binary parameters $E_{v,r,u,t}$ and $A_{v,i,r}$ which are described below. The notations for the sets, parameters and variables are as follows.

Sets:	Notations:
$\boldsymbol{N}$	Set of installations
$\boldsymbol{N_f}$	Set of installations, which require f visits during the week, $\boldsymbol{f} \in \{\mathbf{2..5}\}, \boldsymbol{N_f} \subseteq \boldsymbol{N}$.
$\boldsymbol{V}$	Set of PSV types
$\boldsymbol{W}$	Set of potential departure days, during the week (Working days in Supply base)
$\boldsymbol{T}$	Set of possible departure times during the week
$\boldsymbol{T_w}$	Set of possible departure times during the day w, $\boldsymbol{w} \in \boldsymbol{W}, \boldsymbol{T_w} \in \boldsymbol{T}$.
$\boldsymbol{T_t}$	Set which involve possible departure times, after departure time t, when PSV may not have returned to the supply base from voyages, which have started in time t, $\boldsymbol{t} \in \boldsymbol{T}, \boldsymbol{T_t} \subset \boldsymbol{T}$.
$\boldsymbol{R}$	Set of all possible voyages
$\boldsymbol{R_{v,t}}$	Set of voyages, which may be assigned by a PSV type v, at departure time t, $\boldsymbol{v} \in \boldsymbol{V}, \boldsymbol{t} \in \boldsymbol{T}, \boldsymbol{R_{v,t}} \subseteq \boldsymbol{R}$.
$\boldsymbol{Td_{t,f}}$	Set of departure times after t, which represented time horizon, when PSV may not start or oblige to start from supply base, depended from spread of departure required for visit frequency f,$\boldsymbol{t} \in \boldsymbol{T}, \boldsymbol{f} \in \{\mathbf{2..5}\}, \boldsymbol{T_{t,f}} \subset \boldsymbol{T}$.
Parameters	Notations:
$\boldsymbol{q_v}$	Available quantity of PSVs of type, $\boldsymbol{v} \in \boldsymbol{V}$.
$\boldsymbol{C_v^{TC}}$	Weekly charter cost for PSV of type, v $\boldsymbol{v} \in \boldsymbol{V}$.
$\boldsymbol{C_{vr}^{S}}$	Precalculated sailing, service and base costs for voyage r, which associated on PSV of type v,$\boldsymbol{v} \in \boldsymbol{V}, \boldsymbol{r} \in \boldsymbol{R}$.
$\boldsymbol{B_w}$	Maximum number of departure from the supply base during the day w, $\boldsymbol{w} \in \boldsymbol{W}$
$\boldsymbol{F_i}$	Required number of visits during the week for installation i, $\boldsymbol{i} \in \boldsymbol{I}$.

$\boldsymbol{A_{i,r}}$	Binary parameter equal 1, if and only if installation i have visits by voyage r, $\in \boldsymbol{N}, \boldsymbol{r} \in \boldsymbol{R}$.
$\boldsymbol{E_{v,r,u,t}}$	Binary parameter equal 1 if voyage r, which sailing by PSV of type v, start in time u and will not have returned from voyage to the supply base in possible departure time t, and 0 otherwise, $\boldsymbol{v} \in \boldsymbol{V}, \boldsymbol{r} \in \boldsymbol{R}, \boldsymbol{u} \in \boldsymbol{T}, \boldsymbol{t} \in \boldsymbol{T_t}$.
Variables:	Notations:
$\boldsymbol{x_{v,r,t}}$	Binary variable equal 1, if and only if PSV type v start voyage r in departure time t, and 0 otherwise, $\boldsymbol{v} \in \boldsymbol{V},\ \boldsymbol{r} \in \boldsymbol{R},\ \boldsymbol{t} \in \boldsymbol{T}$.
$\boldsymbol{y_v}$	Integer variable which represent quantity PSV of types $\boldsymbol{v}$, $\boldsymbol{v} \in \boldsymbol{V}$.

Mathematical model:

$$\min \sum_{v\in V} c_v^{TC} y_v + \sum_{v\in V}\sum_{r\in R}\sum_{t\in T} c_r^S x_{vrt} \tag{1}$$

subject to

$$\sum_{v\in V}\sum_{t\in T}\sum_{t\in R_{vt}} A_{vri} x_{vrt} = F_i,\ \forall i \in N \tag{2}$$

$$\sum_{t\in T_t}\sum_{r\in R_{vu}} E_{vrut} x_{vru} \le y_v\ , \forall v \in V, \forall t \in T \tag{3}$$

$$\sum_{v\in V}\sum_{t\in T_w}\sum_{r\in R_{vt}} x_{vrt} \le B_w, \forall w \in W \tag{4}$$

$$\sum_{v\in V}\sum_{t\in T_w}\sum_{r\in R_w} A_{vri} x_{vrt} \le 1, \forall i \in N,\ \forall w \in W \tag{5}$$

$$\sum_{v\in V}\sum_{r\in R_{v,t}}\sum_{u\in T_{t,2}} A_{v,r,i} x_{v,r,u} \le 1, \forall i \in N_2, t \in T \tag{6}$$

$$\sum_{v\in V}\sum_{r\in R_{v,t}}\sum_{u\in T_{t,3}} A_{v,r,i} x_{v,r,u} \ge 1, \forall i \in N_3, t \in T \tag{7}$$

$$\sum_{v\in V}\sum_{r\in R_{v,t}}\sum_{u\in T_{t,4}} A_{v,r,i} x_{v,r,u} \ge 2, \forall i \in N_4, t \in T \tag{8}$$

$$\sum_{v\in V}\sum_{r\in R_{v,t}}\sum_{u\in T_{t,5}} A_{v,r,i} x_{v,r,u} \ge 1, \forall i \in N_5, t \in T \tag{9}$$

$$y_v \le q_v,\ v \in V \tag{10}$$

$$y_v \in Z^+ \tag{11}$$

$$x_{vrt} \in \{0,1\} \tag{12}$$

Constraints description:

The objective function (1) expresses minimization of total vessels charter and sailing costs.

Constraints (2) state that each installation receives the required number of visits during the week. Inequalities (3) ensure that at each time interval t, the number of voyages sailed by vessels of type v is less or equal than the number of PSVs of type v used in the schedule. If, for example, integer variable y_v equals to 2 for vessel type v, then there are two vessels of the same type are used in the schedule, which means that these vessels are coupled and may sail each other voyages. If for each type v of vessels, y_v is equal to one then there are no coupled vessels in the schedule and inequality states that voyages of the same vessel are not overlapped in time. Constraints (4) sets the limit on the number departures from supply base during the day. Constraints (5) grantee, there is possible only 1 departure to each installation, from the supply base during the day. Constraints (6)-(9) grantee, evenly spread of departures for installations with visits frequency f. For example group of constraints (7) ensure even spread of departures for installations with visits frequency 3, stating that there should be at least 1 departure within 3 days is required for each installation in set N_3. Constraints (10) state that number of vessels of type v is can not be more maximum available number of vessels of type q_v.Finally, constraints (11) and (12) set the integer and binary requirements for the y_v and x_{vrt} variables respectively.

6.2 ALNS heuristic.

In this section we provide detailed description of the ALNS heuristic. Heuristic is developed with the use of C# programming language with Microsoft visual studio 2013. Below are provided the pseudo-code of the main algorithm and pseudocodes of the main procedures, and theirs descriptions.

6.2.1 Heuristic overview.

The algorithm is applied for a given number of iterations which are defined by a user. The first phase in the algorithm is generation of initial solution z_0 which is purely randomly. Then, this solution is supposed to be improved over the given number of iterations. At the beginning of the each iteration for the current solution z (at the first iteration $z = z_0$) its neighbourhood $N(z)$ is defended. Under the current solution z is understood the solution with which the algorithm is currently working or tries to improve. As well z^* is defined as the best found solution up to current iteration. Since the aim of the algorithm is to improve the current solution, at the beginning of each iteration transition from the solution z to solution z' in neighbourhood $N(z)$ is performed with the use of *destroy* and *repair* operators.

Transition to the neighbourhood solution z' is called a $move$. The move is performed by the means of removing of some number of visits with the use of some destroy operator and then reinserting these visits back into the schedule with one of the repair operators. The number of visits to be removed is defined randomly between minimal and maximum values defined by a user (average 15% - 20% of the total number of visits). Selection of destroy and repair operators is performed according by assigned to them probabilities, which are initially equal. There are three destroy and three repair operators in the algorithm. The destroy operators are: *worst removal* (Ropke and Pisinger 2006), *Shaw removal* (Ropke and Pisinger 2006) and *random voyage removal*. The repair operators are: *deep greed insertion*, *2-regret insertion* and *3-regret insertion* (Ropke and Pisinger 2006).As it was mentioned above, each destroy and repair operator has its own probability of being selected. The sum of probabilities of destroy operators equal to 1.The same is for the repair operators. These probabilities are recalculated after each (user defined) number of iterations based on their previous performance. Proper description of the adaptiveness mechanism is provided below. For detailed description of the *mechanism of adaptiveness* see Ropke and Pisinger (2006), Pisinger and Ropke (2010). If the move to the neighbourhood solution z' is performed successfully (Pseudocode 6.2.1 line 12) i.e. all the removed visits are inserted back into the schedule, the algorithm proceeds to the improvement phase. At the improvement phase, the solution z' is tried to be improved by the set of improvement operators: *reduce number of voyages*, *reduce number of vessels*, *swap visits between voyages* and *relocate visits between voyages* (Pseudocode 6.2.1 lines 13-20). These procedures are located in certain sequence of the algorithm, and applied cyclically while the solution can be improved. The first procedure *reduce number of voyages* aims to decrease number of voyages in the schedule by relocating visits from some voyage to other voyages in the schedule. This is done with the aim to reduce the sailing cost and the fleet size, since the idle time of some vessels reduced (if some voyages were eliminated). The fleet size reduction is provided by the procedure *reduce number of vessels*, which attempts to reassign all voyages from some vessel to other vessels. If such vessel is found, and voyages are reassigned, then this vessel is marked as unused. Procedure *swap visits between voyages* analyses all combinations of swapping two visits between all voyages. The aim of this procedure is to reduce the cost of the schedule, which is, of course, dependent on the duration of the voyages. Therefore, if the cost is reduced and as follows, durations of voyages, then vessels' idle time is increased. Moreover, there is a possibility to decrease the fleet size again, which is done by the mentioned above procedure *reduce number of vessels*. And, the last procedure – *relocate* *visits* tries to find best relocation of visits between voyages. After this procedure as well applied the procedure *reduce number of vessels.* Detailed pseudocodes of these improvement procedures are provided bellow in section 6.2.6.

After the improvement phase solution z' is compared to solution z (solution which was before the move at the beginning of the current iteration) and z^* (the best found solution which was found over the all iterations). If $z' < z^*$ then, we have found the new best solution, $z^* = z'$. And $z = z'$ i.e. the current solution is made equal to this new solution z' and is supposed to be improved further on the next iteration. If $z' > z^*$ and $z' < z$ then $z = z'$ and z^* remains the same (since it was not improved). If $z' > z^*$ and $z' > z$ then z' is accepted ($z = z'$) with some probability (acceptance criteria), (see Pseudocode 6.2.1 line 25). If z' is not accepted then the algorithm proceeds to the next iteration with the solution that was at the beginning of the current i.e. current solution z remains unchanged. The logic of the acceptance criteria is explained further (see Section 6.2.7). If a certain number of iterations (δ) passed after the last update of weights, than the weights are updated again (see Pseudocode 6.2.1 lines 30-31 and section 6.2.5). The algorithm proceeds to the next iteration. After the last iteration, the algorithm returns the best found solution.

```
Procedure 6.2.1 Main ALNS heuristic for PSVPP
1:$z \leftarrow Construct\ initial\ solution$ (Procedure 6.2.2.);
2:Initialize the weight $\pi$;
3:set the temperature $T$;
4:set $z^* \leftarrow z$;
5:for $i = 0$ to $N$
6:    $z' \leftarrow z$;
7:    $q_{iter}$ ←Select number of visits to be removed;
8:    $Opr_{rem}$ ←Select removal operator;
9:    $z' \leftarrow Opr_{rem}(z', q_{iter,} S)$ (Procedures in section 6.2.3);
10:   $Opr_{ins} \leftarrow$ Select insert operator;
11:   $z' \leftarrow Opr_{ins}(z', S)$ (Procedures in section 6.2.4);
12:   if $S == \emptyset$ and $z'$ is feasible then
13:         do
14:               Reduce number of routes (Procedure 6.2.6.1);
15:               Reduce number of vessels (Procedure 6.2.6.2);
16:               Swap visits between voyages (Procedure 6.2.6.3);
17:               Reduce number of vessels (Procedure 6.2.6.2);
18:               Relocate visits between voyages (Procedure 6.2.6.4);
19:               Reduce number of vessels (Procedure 6.2.6.2);
20:         while z improves;
21:         if $c(z') \leq c(z^*)$ then
22:               $z^* \leftarrow z'$;
23:               $z \leftarrow z'$;
24:         end if
25:         if $Accept(z, z')$ then (Described in section 6.2.7)
26:               $z \leftarrow z'$;
27:         end if
28:   end if
29:   if $i/\delta == 0$ then
30:         Update weights $\pi$ (Described in section 6.2.5);
31:   end If
32:   $T \leftarrow T * c$;
33:Next $i$;
34:return $s^*$;
```

6.2.2 Initial solution

Here we described the procedure for generate randomly feasible initial solution (schedule). The initial solution contains a set of voyages with a certain departure time during the week and sailing by the specific vessels. Represented schedule satisfy spread of departure constraint and contain required number of visits for each installation. The procedure for generation of the feasible initial solution is described below (Procedure 6.2.2).

Procedure 6.2.2. Construct initial solution for ALNS.

```
1: Function ConstructInitialSolution(installations $I$, Vessels $V$, maximum number of departure per day $\mu dep$, maximum installations per route $\mu inst$ )
2: Array: R = an array containing all routes;
3: do
4:     $R = \emptyset$;
5:     Bool Flag: $f$ = $true$;
6:     for each $i \in I$
7:         Randomly generate departure day pattern with respect to visit frequency;
8:         $VisComb_{t,i} = 1$ if installation $i$ is assigned on departure day t,0 otherwise;
9:     end for
10:    for each $t \in T$
11:        Define number of visits per departure day t: $q = \sum_{i \in I} VisComb_{t,i}$
12:        if $q > \mu dep * \mu inst$ then
13:            $f = false$;
14:        end if
15:        Define number of voyages per day: $\rho = \lceil q/\mu inst \rceil$
16:        Define number of visits per route: $\tau = \lceil q/\rho \rceil$;
17:        for k = 0 to $\rho$
18:            Create empty voyages r;
19:            Counter: v = 0
20:            do
21:                Installation: $i = I_v$;
22:                if $VisComb_{t,i} == 1$ then
23:                    Assign visit to installation i on route r;
24:                end if
25:                set $v = v + 1$;
26:            while $v < |I|$ and $|r| < \tau$;
27:            Call procedure 6.2.7.1 for route r;
28:            $R = R \cup r$;
29:        end for
30:    end for
31:    for each $r \in R$
32:        Bool flag: $possible = false$;
33:        for each $v \in V$
34:            if voyage $r$ is possible to assign on vessel $v$ then
35:                voyage r is assigned to vessel v;
36:                $possible = true$;
37:                exit for;
38:            end if
39:        end for
40:        if $possible == false$ then
41:            $f = false$;
42:        end if
43:    end for
44: while $f == false$
45: $z_0 \leftarrow R$;
46: return $z_0$;
```

6.2.3 Destroy operators

This section describes three destroy operators. All three removal operators return the solution z and pool of uninserted visits S as an output. More detailed description of all removal operators is provided below.

6.2.3.1 Shaw removal

Shaw removal heuristic was first presented by in Shaw (1997),Shaw (1998).The main objective of the *Shaw removal operator* is to remove visits which are similar i.e. close in servicing time, location etc. For more detail description, see Ropke and Pisinger (2006). This approach provides easier possibility to insert all visits back into the schedule and perhaps better neighbor solution. For determining somewhat similar visits we define a *related measure R(i, j).* This measure expresses a relatedness between two visits i and j and is computed by a given formula:

$$R(i,j) = \alpha(d_{i,j}) + \beta(|T_i - T_j|) \tag{13}$$

This formula contains two terms:$d_{i,j}$ which denotes the travel distance between installations visits and T_i indicates departure time to installation i. Both terms are weighted by the weights α and β. The procedure for removing visits from schedule by shaw removal is presented below in Pseudocode 6.2.3.1.

Pseudocode 6.2.3.1. Shaw Removal.

```
1:Function ShawRemoval(solution $z$, number of visits $q$)
2:    visit : $v$ = a randomly selected visit from $z$;
3:    pool of visits: $S = \{v\}$;
4:    remove visit $v$ from the soltution $z$;
5:    while $|S| < q$ do
6:          $r$ = a randomly selected request from $S$;
7:          Array : $V$ = an array containing all visits from $z$ not in $S$;
8:          Array : $R$ = an array containing rank for each visit in $z$;
9:          Counter: $i = 0$;
10:         while $i < |V|$ do
11:                $R_i = \alpha(d_{V_i,r}) + \beta(T_{V_i} - T_r)$;
12:                $i = i + 1$;
13:         end while
14:         sort $R$ such that $i < j \Rightarrow R_i < R_j$;
15:         Insert in Pool $S$ first visit in array $R$;
16:   end while
16:return $z,S$;
```

6.2.3.2 Random Voyage removal

The random voyage removal operator simply selects q random voyages from the solution z, and then place all visits from voyages into pool S and remove voyages from the solution. Pseudocode of this procedure is shown below (Pseudocode 6.2.3.1).

Pseudocode 6.2.3.2. Random Voyage removal.

```
Function VoyageRemoval(solution $z$, number of voyages remove $q$)
pool of visits: $S = \emptyset$;
Counter: $i = 0$;
while $i < q$ do
    $r$ = a randomly selected voyage from $z$;
    Array : $V$ = an array containing all visits from $r$;
    $S = S \cup V$;
    remove r from solution z;
    $i = i + 1$;
end while
return $z, S$;
```

6.2.3.3 Worst Removal

The general idea of the *worst removal operator* is to remove visits with the maximum cost reduction values, i.e. remove visits with high cost. In presented pseudocode 6.2.3.3. for each visit one by one cost c' (schedule cost without visit) is computed. Visit with lowest cost c' is removed from the solution z. Algorithm repeats while number of visits in pool S less than q.

Pseudocode 6.2.3.3. Worst removal.

```
Function Worst Removal (solution $z$, number of visits $q$)
    Pool of visits :$S = \emptyset$;
    while $|S| < q$ do
        Array : $V$ = contains all visits from solution $z$;
        Cost: $= c(z)$ ;
        $\vartheta = \emptyset$;
        for each $v \in V$
            remove visit $v$ from solution z;
            Cost: $c' = c(z)$;
            if $c' < c$ then
                $\vartheta = v$;
                $c = c'$;
            end if
            insert visit $v$ back into the solution $z$;
        end for
        $S = S \cup \{\vartheta\}$;
        remove visit $\vartheta$ from solution $z$;
    end while
return $S, z$;
```

6.2.4 Repair operators

General idea of repair operators is insertion back into schedule all visits from the pool of uninserted visits S. We provide below descriptions for 2 repair operators with their pseudocodes: *deep greedy insertion* and *k-regret insertion* (which represent the class of regret operators, depending of the value of the parameter k).

6.2.4.1 Deep Greedy insertion

The deep greedy insertion is a simple construction heuristic. Heuristic contains several number of iterations. An Heuristic contains several number of iterations. At each iteration algorithm tries to insert each visit from pool S into the schedule and if insertion is possible, then procedure computes schedule cost with this insertion. In the end of iteration, an algorithm inserts visit into the schedule with the minimal cost increase and removes visit from S. Pseudocode of the algorithm is showed below (**Procedure** 6.2.4.1).

Procedure 6.2.4.1 Deep greedy insertion

```
1:Function DeepGreedyInsertion(solution $z$, Visits Pool $S$)
2:Best evaluation: $\varepsilon = \emptyset$;
3:Array : $R$ = an array containing all voyages from z;
3:do
4:    $\varepsilon = \emptyset$;
5:    Counter: $i = 0$;
6:    while $i < |S|$ do
7:          Counter $j = 0$;
8:          while $j < |R|$
9:                      Evaluation: $e \leftarrow$ get evaluation by calling
                        procedure 6.2.7.2 for $R_j, S_i$;
10:               if $e \neq \emptyset$ and $c(e) < c(\varepsilon)$ then
11:                     $\varepsilon = e$;
12:               end if
13:               $j = j + 1$;
14:         end while
15:         $i = i + 1$;
16:   end while
17:   if $\varepsilon \neq \emptyset$ then
18:         $r \leftarrow$ target route in evaluation $\varepsilon$;
19:         $v \leftarrow$ insertion visit in evaluation $\varepsilon$;
20:         Insert visit $v$ in voyage $r$;
21:         Call procedure 6.2.7.1. for route $r$;
22:         Remove visit $v$ from pool $S$;
23:   end if
24: while $\varepsilon \neq \emptyset$ and $|S| > 0$;
25: return $z, S$;
```

6.2.4.2 Regret-k insertion heuristic

The *regret* heuristic represents an evolution of the deep greedy heuristic by making a kind of look ahead when selecting a visit for insertion. Let $\Delta f_{i,k}$ the change in the objective function resulted after the insertion of a visit *i* into voyage *k*. We define $x_{ik} \in \{1,..,n\}$ as a variable indicating the route for which insertion of a visit *i* has the *k*'th lowest insertion cost (variables are sorted in increasing order of the value of the objective function). If k < k' then $\Delta f_{i,x_{ik}} \leq \Delta f_{i,x_{ik'}}$. So, we can we define $c_i^* = \Delta f_{i,x_{i2}} - \Delta f_{i,x_{i1}}$ as the difference between the best and the second to the best insertion options for visit *i* or in other words we define c_i^* as a regret value.

During the search for better visit for insertion, the regret heuristic, at each iteration, selects the visit so that:

$$\max_{i \in I} c_i^* \tag{14}$$

In other words, we strive to insert a visit which we would regret if we do not insert it now. When inserting a visit *i* into route *k*, the visit is inserted into the minimum cost position. Perusing this logic the heuristic can be extended by defining a class of regret heuristics. The *k-regret* heuristic aims to insert a visit such that:

$$\max_{i \in I} \left\{ \sum_{j=1}^{k} (\Delta f_{i,x_{ij}} - \Delta f_{i,x_{i1}}) \right\} \tag{15}$$

this means that we take into account insertion options of visit *i* for the first best *k* insertions. If, applying *k-regert*, some visits cannot be inserted into *n*-*k*+1 number of routes then the request with the fewest number of routes for insertion. For this heurist, at least two insertions options required to perform the assessment. Formulation (15) represents *regret-2* heuristic, since it consider the two best routes for insertion of a visit. The k-regret heuristic considerers for each visit *i* the best *k* routes for insertion and selects the one with maximal cost difference of insertion into k-1. Pseudocode (6.2.4.2) presents the logic of k-regret insertion procedure, adopted for our heuristic.

Procedure 6.2.4.2. Regret-k insertion

```
1: **Function** RegretInsertion(solution $z$, number of visits $q$, regret $k$)
2:    **do**
3:          Highest regret value: $\sigma = 0$;
4:          Best evaluation: $\varepsilon = \emptyset$;
5:          Counter: $i = 0$;
6:          **while** $i < |S|$ **do**
7:                Counter: $j = 0$;
8:                Array : $E$ = possible evaluation for visit $i$;
9:                Array:  := containing all voyages in solution $z$;
10:               **while** $j < |R|$ **do**
11:                     evaluation: $e \leftarrow$get evaluation by calling
                        **procedure 6.2.7.2** for $R_j, S_i$;
12:                     **if** $e \neq \emptyset$ **then**
13:                           $E = E \cup \{e\}$;
14:                     **end if**
15:                     $j = j + 1$;
16:               **end while**
17:               sort $E$ such that $a < b \Rightarrow f(E_a) > f(E_b)$;
18:               **if** $|E| \geq k$ **then**
19:                     Counter: $j = 0$;
20:                     Accumulated regret value: $\theta = 0$;
21:                     **while** $j < k$ **do**
22:                           $\theta = \theta + (c(E_j) - c(z))$ ;
23:                           $j = j + 1$;
24:                     **end while**
25:                     **if** $(\theta > \sigma)$ **then**
26:                           $\sigma = \theta$;
27:                           $\varepsilon = E_0$;
28:                     **end If**
29:               **else if** $|E| > 0$
30:                     $\varepsilon = E_0$;
31:                     **break while**;
32:               **end If**
33:               $i = i + 1$;
34:         **end while**
35:         **if** $\varepsilon \neq \emptyset$ **then**
36:               $r \leftarrow$ target route in evaluation $\varepsilon$;
37:               $v \leftarrow$ insertion visit in evaluation $\varepsilon$;
38:               Insert visit $v$ in voyage $r$;
39:               Call **procedure 6.2.7.1.** for route $r$;
40                Remove visit $v$ from pool $S$;
41:         **end if**
42:   **while** $\varepsilon \neq 0$;
43: **return** $z, S$;
```

6.2.5 Selection of destroy and repair operators.

In section 6.2.3. we described three destroy operators (*Shaw*, *random voyage* and *worst* removal) and in section 6.2.4. we provide three repair operators (deep greedy, regret-2 insertion and regret-3). In this section, we provide selection mechanism that is used for both groups of operators. We remind that selection of destroy and repair operator take place at the beginning of each iteration.

6.2.5.1 Probability recalculation.

All operators are selected according to probabilities which depends on their performance during the run of the algorithm. Probabilities are equal at the first iteration. Let K be the set of operators (either destroy or repair), $j \in K$. P_j – probability of selection of operator j. In order to select an operator we assign a weight to each of the operators w_i. The probability of selection of an operator j is then calculated according to the following formulation (see Ropke and Pisinger (2006)):

$$p_j = \frac{w_j}{\sum_{i=1}^{k} w_i} \tag{16}$$

N.B! Probabilities are defined separately for destroy and repair operators.

6.2.5.2 Weights adjustment.

In the above section we described weights and probability calculation for destroy and repair operator. In this section we describe how these weights are adjusted for each operator during the algorithm run. The main idea of weight adjustment is to record performance of each operator and assign different score depending on the performance efficiency. The search is divided into a number of *segments*. Each segment corresponds 50 or 100 iterations (user defined). Since the initial probabilities are equal for each operator, the score for each operator is set to zero. The score of an operator increases by σ_1, σ_2, or σ_3 depending on the following conditions:

Parameter	Description
σ_1	After applying the last remove-insert operators the algorithm found new global best solution.
σ_2	After applying the last remove-insert operators the algorithm found new solution that has not acceptance before, worse than global best solution, but better than current.

σ_3	After applying the last remove-insert operators the algorithm found new solution that has not acceptance before, worse than current solution, but solution was accepted.

Table 1 – Types of parameters that increase the score of a heuristic

At the end of each segment the weight of an operator *i* is recalculated based on its score. Let w_{ij} be the weight of operator *i* and the j_s *segment.* The weight is used in formulation (17) for probability calculation. When the segment *j* is over, the weight of operator *i* within the next segment *j*+1is defined as follows:

$$w_{i,j+1} = w_{i,j}(1 - r) + r\frac{\pi_i}{\theta_i} \tag{17}$$

Where π_i corresponds to the total score of the an operator *i* for the last segment *j.* θ_i represents the number of times the operator *i* was used during the last segment. As well, there is reaction factor *r* which defines the degree of reaction of weights adjustment. For example if we set *r* to 0, the scores are not used at all and the algorithm uses those initial weights. For more information see (Ropke and Pisinger 2006).

6.2.6 Improvement operators

This section describes following set of improvement operators: *reduce number of voyages*, *reduce number of vessels*, *swap visits between voyages* and *relocate visits between voyages*. The general idea of represented operators is schedule cost decrease after made move to the neighborhood by efficient application improvement procedures. Swap visits and relocate visits are aimed to reduce voyages durations and sailing costs. Reduce number of vessels tries to reduce charter cost of the schedule by minimizing fleet size composition. While reduce number of voyages simultaneously reduce sailing and charter costs by minimizing number of routes. More detail descriptions and pseudocodes for each improvement operator are provided below.

6.2.6.1 Reduce number of Voyages.

Procedure 6.2.6.1. tries to reduce number of voyages in the schedule by a relocation all visits from each voyage into another voyages. This procedure allows increasing an idle time between voyages and decrease total schedule cost.

Procedure 6.2.6.1. Reduce number of voyages

```
Function ReduceNuberOfvoyages (solution $z$)
Array : $K$ = set containing all voyages from the solution;
Index of voyage: $\varepsilon = -1$;
do
   $\varepsilon = -1$;
   Schedule Cost Change: $\lambda = 0$;
   Counter $i = 0$;
   while $i < |K|$ do
         Solution: $z'$ = Copy of the solution $z$;
         Array: $S =$ an array containing all visits from voyage $K_i$;
         Remove voyage $K_i$ from the solution $z'$;;
         Try insert visits from $S$ into solution $z'$ (Procedure 6.2.4.1);
         if $S == \emptyset$ and $\lambda < c(z) - c(z')$then
               $\lambda = c(z) - c(z')$;
               $\varepsilon = i$;
         end if
         $i = i + 1$;
   end while
   if $\varepsilon > 0$ then
         Array: $S =$ an array containing all visits from voyage $K_\varepsilon$;
         Remove voyage $K_\varepsilon$ from the solution $z$;
         Insert visits from $S$ into solution $z$ (Procedure 6.2.4.1);
   end if
while $\varepsilon \geq 0$
return z;
```

6.2.6.2 Reduce number of Vessels

This procedure tries to reduce fleet size by reassigning each voyage from one vessel to other vessels. If it is performed then the total schedule cost are sufficiently reduced. Pseudocode of this procedure are provided below. (Procedure 6.2.6.2.)

Procedure 6.2.6.2. Reduce number of vessels

```
1: Function ReduceNumberOfVessels(solution $z$)
2:     $z'$ = copy of the solution $z$;
3:     Set: $V =$ a set containing all vessels from solution $z'$;
4:     $\theta = M$;
5:     Counter: $k = 0$;
6:     while $k < |R|$
7:           $\mu \leftarrow$ Insert value which is containing overlap in time if all
             voyages from vessels $V_i$ will be reassigned to another vessels
             in schedule $z'$;
8:           if $\mu < \theta$ then
9:                 $\vartheta = i$;
10:                $\mu = \theta$;
11:          end if
12:          $k = k + 1$;
13:    end while
14:    Array $R$ = an array containing all voyages from vessel $V_\vartheta$;
15:    Pool of visits $S = \emptyset$;
16:    $k = 0$;
17:    while $k < |R|$
18:          Remove voyage $R_k$ from vessel $V_\vartheta$;
19:          Insert voyage $R_k$ into the vessel $V_\varepsilon$;
20           Move each voyage during a day in vessel $V_\varepsilon$ for reducing overlap
             in time;
21:          Array: $P$ an array containing all voyages from vessel $V_\varepsilon$;
22:          Counter m = 0;
23:          while $\mathrm{m} < |P|$
24:                if voyage $P_m$ is overlapped in time then
25:                      Visit: $v$ = worst visit in voyage $P_m$;
26:                      Remove visit $v$ from voyage $P_m$;
27:                      Insert visit $v$ in Pool of visits $S$;
28:                else
29:                      $m = m + 1$;
30:                end if
31:          end while
32:          k = k + 1;
33:    end while;
34:    remove vessel $V_\vartheta$ from the solution $z'$;
35:    Call regret-2 insertion for Pool of visits $S$ and solution $z'$
       (Procedure 6.2.4.2)
36:    if $S = \emptyset$ then
37:          $z = z'$;
38:    end if
39: return $z$
```

6.2.6.3 Swap visits between voyages

Swap procedure is based on the one presented by Bräysy and Gendreau (2005). The main idea of the procedure is in the swap of customers between two any routes. Which customers should be selected for swapping is decided either heuristically (randomly) or by enumerating all possible combination and the selecting the most cost efficient. Figure 3

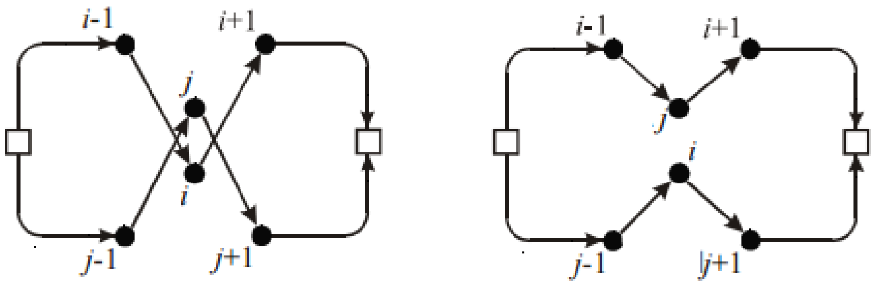


Figure 3 "Swap" improvement procedure (Bräysy and Gendreau 2005)

presents the example of swap procedures by (Bräysy and Gendreau 2005) which is executed under 6 edges. It consist of two figures, the left one demonstrates two routes before the application of the procedure, the right figure shows how routes were modified after "Swap". The edges $(i-1, i)$, $(i, i+1)$, $(j-1, j)$ and $(j, j+1)$ are replaced $(i-1, j)$, $(j, i+1)$, $(j-1, i)$ and $(i, j+1)$, i.e., two visits from different voyages are simultaneously inserted into the other voyages. Proposed in this master thesis procedure (Procedure 6.2.6.3) tries to swap two visits between voyages while cost decreasing is possible it possible.

6.2.6.4 Relocate visits between voyages

This procedure tries to reduce cost by insertion each visit from each voyage into another voyage. For example, in Figure 4 the right side shows the picture after execution of the procedure, the edges $(i-1, i)$, $(i, i+1)$ and $(j, j+1)$ are replaced by $(i-1, i+1)$, (j, i) and $(i, j+1)$, i.e., visit i from the origin voyage is inserted into the destination voyage. Procedure repeats while the total schedule cost decreases. Pseudocode of this porcedure is described in Procedure (6.2.6.4).

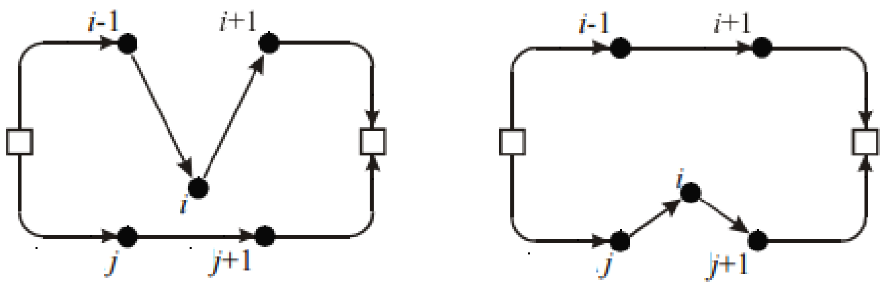


Figure 4 Relocate insertion procedure (Bräysy and Gendreau 2005)

Procedure 6.2.6.3. Swap visits between voyages

```
Function SwapVisits ($z \in \{solution\}$)
    $\varepsilon = \emptyset$;
    Array : $R$ = an array containing all voyages from z;
    do
        $\Delta = 0$;
        $\varepsilon = \emptyset$;
        $i = 0$;
        while $i < |R|$ do
            Array : $V$ = an array containing all visits from voyage $R_i$;
            $j = 0$;
            while $j < |V|$ do
                $k = i + 1$;
                while $k < |R|$ do
                    $l = 0$;
                    Array : $\Upsilon$ = all visits from voyage $R_k$;
                    while $l < |\Upsilon|$ do
                        if $V_j \neq \Upsilon_l$ then
                            evaluation $e \leftarrow$ get evaluation
                            by calling procedure 6.2.7.3
                            for $R_i, V_j, R_k, \Upsilon_l$;
                            If $e \neq \emptyset$ and $c(z) - c(e) > \Delta$ then
                                $\Delta = c(z) - c(e)$;
                                $\varepsilon = e$;
                            end if
                        end if
                        $l = l + 1$;
                    end while
                    $k = k + 1$;
                end while
                $j = j + 1$;
            end while
            $i = i + 1$;
        end while
        if $\varepsilon \neq \emptyset$ then
            $r1 \leftarrow$ route $r1$ in evaluation $\varepsilon$;
            $r2 \leftarrow$ route $r2$ in evaluation $\varepsilon$;
            $v1 \leftarrow$visit $v1$ from route $r1$ in evaluation $\varepsilon$;
            $v2 \leftarrow$visit $v2$, from route $r2$ in evaluation $\varepsilon$;
            remove visit $v1$ from route $r1$;
            remove visit $v2$ from route $r2$;
            insert visit $v1$ in route $r2$;
            insert visit $v2$ in route $r2$;
            Call procedure 6.2.7.1. for route $r1$;
            Call procedure 6.2.7.1. for route $r2$;
        end if
    while $\varepsilon \neq \emptyset$
return $z$;
```

Procedure 6.2.6.4. Relocate visits between voyages

```
Function RelocateVisits($z \in \{solution\}$)
    Evaluation: $\varepsilon = \emptyset$;
    Array : $R$ = an array containing all voyages from z;
    do
        Cost decrease value $\Delta = 0$;
        $\varepsilon = \emptyset$;
        Counter: $i = 0$;
        while $i < |R|$ do
            Array: $V$ = an array containing all visits from voyage $R_i$;
            Counter: $j = 0$;
            while $j < |V|$ do
                Counter: $k = 0$;
                while $k < |R|$ do
                    if $k \neq i$ then
                        Evaluation: $e$= Get evaluation for
                        insertion visit $V_j$ from route $R_i$ into
                        route $R_k$ (Procedure 6.2.7.2.);
                        if $e \neq \emptyset$ and $c(z) - c(e) > \Delta$ then
                            $\Delta = c(z) - c(e)$;
                            $\varepsilon = e$;
                        end if
                    end if
                    $k = k + 1$;
                end while
                $j = j + 1$;
            end while
            $i = i + 1$;
        end while
        if $\varepsilon \neq \emptyset$ then
            $sr \leftarrow$ source route in evaluation $\varepsilon$;
            $tr \leftarrow$ target route in evaluation $\varepsilon$;
            $\vartheta \leftarrow$ visit in evaluation $\varepsilon$ from route $sr$;
            remove visit $\vartheta$ from route $sr$;
            insert visit $\vartheta$ in route $tr$;
            Call procedure 6.2.7.1. for route $sr$;
            Call procedure 6.2.7.1. for route $tr$;
        end if
    while $\varepsilon \neq \emptyset$
return $z$;
```

6.2.7 Route optimization operator and evaluations.

In this section are detailed described route optimization procedure and “Insert visit Evaluation” and “Swap Visits Evaluation”.

6.2.7.1 Intra-voyage optimization procedure

The general idea of intra-voyage optimization procedure (Procedure 6.2.7.1) is determination an optimal sequence of installations in voyage r by solving TSPTW. This algorithm is frequently called from other procedures. In this regard, we made this procedure more simply with the aim to decrease computational time. The algorithm uses first-accept strategy while constructs an optimal visits sequence, i.e. procedure places each visit in first possible position in sequence if voyage duration is reduced.

Procedure(6.2.7.1) Intra-voyage optimization.

```
Function VoyageOptimization(voyage $r$)
do
    flag: $improve = false$;
    $V \leftarrow$ set of visits in voyage $r$;
    Counter: $i = 0$;
    do
        flag: $stepimprove = false$;
        $d \leftarrow$ voyage duration;
        set $v = V_i$;
        Counter: $j = 0$;
        do
            visit $v$ removed from position $i$;
            visit $v$ is placed on position $j$;
            $nd \leftarrow$ voyage duration;
            if $nd < d$ then
                $stepimprove = true$;
                if $j > i$ then
                    $i = i - 1$;
                end if
            else
                visit $v$ removed from position $j$;
                visit $v$ is placed on position $i$;
            end if
            $j = j + 1$;
        while $j < |V|$ and $stepimprove == false$ ;
        if $stepimprove == true$ then
            $improve = true$;
        end if
        $i = i + 1$;
    while $i < |V|$;
while $improve == true$;
return r;
```

6.2.7.2 Insert evaluation procedure.

This procedure (procedure 6.2.7.2.) is called from procedures 6.2.4.1 and 6.2.4.2. The aim of this procedure is compute the evaluation of insertion visit ϑ into the voyage tp. If it possible under several restrictions (which are described in pseudocode) a procedure inserts visit into the voyage, computes schedule cost, and removes visit from the voyage. In the end procedure returns evaluation ε of insertion, which is empty if insertion is impossible. Pseudocode of the procedure is provided below (Procedure 6.2.7.2).

```
Procedure(6.2.7.2) Inert visit evaluation
1: Function GetInsertEvaluation(solution $z$, voyage $t\rho$, visit $\vartheta$, voyage $s\rho$
(optionally))
2:    Evaluation: $\varepsilon = \emptyset$;
3:    Bool : $\omega = true$ if insert visit is possible under vessels capacity
      constraint, $false$ otherwise;
4:    Bool : $\gamma = true$ if insert visit is possible under maximum number of
      visits per voyage constraint, $false$ otherwise;
5:    Bool : $\theta = true$ if visit $\vartheta$ is not already on voyage $t\rho$, $false$
otherwise;
6:    Bool :$\lambda = true$ if insert visit is possible under spread of departures
      constraint, $false$ otherwise;
7:    if $\omega == true$ and $\gamma == true$ and $\theta == true$ and $\lambda == true$ then
8:          if $s\rho \,!= \emptyset$ then
9:                Remove visit $\vartheta$ from voyage $s\rho$;
10:               Call procedure 6.2.7.1. for voyage $s\rho$;
11:         end if
12:         Insert visit $\vartheta$ in voyage $t\rho$;
13:         Call procedure 6.2.7.1. for voyage $t\rho$;
14          Bool : $\sigma = true$ if voyage $t\rho$ is possible under maximum route
            duration constraint, $false$ otherwise;
15:         Bool :$\mu = true$ if voyage $t\rho$ is not overlapped in schedule during
            the planning horizon, $false$ otherwise;
16:         if $\mu == true$ and $\sigma == true$ then
17:               $\varepsilon \leftarrow$ voyages $t\rho$ and $s\rho$ ,visit $\vartheta$ , and $c(z)$;
18:         end if
19:         Remove visit $\vartheta$ from voyage $t\rho$;
20:         Insert visit $\vartheta$ in voyage $s\rho$;
21:         Call procedure 6.2.7.1. for voyage $t\rho$;
22:         Call procedure 6.2.7.1. for voyage $s\rho$;
23:   end if
24: return $\varepsilon$;
```

6.2.7.3 Swap evaluation procedure.

This procedure (procedure 6.2.7.2.) is called from procedure 6.2.6.3. The aim of this procedure is compute the evaluation of insertion visit $\vartheta 1$ into the voyage $r2$ and $\vartheta 2$ into the voyage $r1$. If it possible under several restrictions (which are described in pseudocode) a procedure Swaps visits between voyages, computes schedule cost, and returnt visit back into original voyages. In the end procedure returns evaluation ε of swap visits, which is empty if swap is impossible. Pseudocode of the procedure is provided below (Procedure 6.2.7.3).

```
Procedure(6.2.7.3) Swap visits evaluation
1: Function GetSwapEvaluation(solution $z$, voyage1 $r1$, visit1 $v1$, voyage2 $r2$, visit2 $v2$)
2:     $\varepsilon = \emptyset$;
3:     Bool : $\omega = true$ if swap visits is possible under vessels capacity
       constraint, $false$ otherwise;
4:     Bool : $\gamma = true$ if swap visits is possible under maximum number of
       visits per voyage constraint, $false$ otherwise;
5:     Bool : $\theta = true$ if visit $v1$ is not already on voyage $r2$ and visit $v2$
       is not on voyage $r1$, $false$ otherwise;
6:     Bool : $\lambda = true$ if swap visits is possible under spread of departures
       constraint, $false$ otherwise;
7:     if $\omega == true$ and $\gamma == true$ and $\theta == true$ and $\lambda == true$ then
8:         Remove visit $v1$ from voyage $r1$;
9:         Remove visit $v2$ from voyage $r2$;
10:        Insert visit $v2$ in voyage $r1$;
11:        Insert visit $v1$ in voyage $r2$;
12:        Call procedure 6.2.7.1. for voyage ($r1$);
13:        Call procedure 6.2.7.1. for voyage ($r2$);
14:        Bool : $\sigma = true$ if voyages $r1$ and $r2$ are possible under maximum
           route duration constraint, $false$ otherwise;
15:        Bool : $\mu = true$ if voyages $r1$ and $r2$ is not overlapped in
           schedule during the planning horizon, $false$ otherwise;
16:        if $\mu == true$ and $\sigma == true$ then
17:            $\varepsilon \leftarrow$ voyages $r1$ and $r2$, visit1 $v1$ and $v2$, and $c(z)$;
18:        end if
19:        Remove visit $v2$ from voyage $r1$;
20:        Remove visit $v1$ from voyage $r2$;
21:        Insert visit $v1$ in voyage $r1$;
22:        Insert visit $v2$ in voyage $r2$;
23:        Call procedure 6.2.7.1. for voyage ($r1$);
24:        Call procedure 6.2.7.1. for voyage ($r2$);
25:    end if
26 return $\varepsilon$;
```

6.2.8 Acceptance criteria

Since we need to diversify the search and strive to avoid local optimum, we need some mechanism enabling us to do so. The simplest way is to accept at the end of each iteration only those solutions, which are better than the current solution. This (as we convinced) leads to trapping into some local optimum neighbourhood. Therefore, we take the idea of solution acceptance from simulated annealing. Let s' be the solution obtained at the end of an iteration and s is the current solution. We assume to accept the solution s' with probability:

$$e^{-\frac{f(s')-f(s)}{T}} \tag{18}$$

Where T is the temperature and always positive (T>0). The temperature starts from T_{start} and decreases with each iteration according to the formula $T = T * c$, where $0 < c < 1$ is the cooling rate of the temperature. In our case, we set T_{start} equal to the cost of the initial solution. The values of c is made depended on the total number of iterations (η) the algorithm is to be run:

$$c = 1 - \frac{1}{\eta * \frac{1}{7}} \tag{19}$$

the term $\frac{1}{7}$ is defined empirically so that the probability of accepting the solution $s,'$ when $c(s') > c(s)$, is almost 0 by the last iteration.

7.0 Computational Experiments

In this section, we describe our computational experiments. In section 7.1 we start with finding of some tuning instances and proceed with the description of parameters tuning. In section 7.2 we present the results obtained by the algorithm. We compare heuristic performance to the two-phase exact approach and provide heuristic results for large instances.

7.1 Tuning instances

The set of tuning instances is represented by instances of medium size. The total number of installations supply from the base located in Mongstad. So we randomly deleted some number of installations and created 10 different instances. There are 3 instances with 8 installations, 3 instances with 10 installations, 3 instances with 12 installations and 1 instance with 13 installations. Those instances of the same size have different combinations of installation. The number of visits in this instances are varies from 27 to 48.

7.1.1 Parameters tuning

In this section we present user defined parameters which are subject to tuning and results.

All our parameters are subdivided in to three categories: parameters of destroy operators, parameters of repair operators and those used in acceptance criteria. We first review parameters of destroy operators. First we have do define which portion of the solution we have to destroy when making a move. As it was mentioned above, we remove random number of visits, which is limited by some minimum and maximum values (LV and MV). So, we have to define these minimum and maximum values experimentally. As regards destroy operators, only Shaw removal heuristic contains controlled parameters: α and β. Since we have already defined which regret heuristics to apply (regret-2, regret-3), there is no need in parameters tuning for insertion heuristics. As well we do not conduct experiments with the size of the segment for which weights and probabilities of *repair* and *destroy* operators are updated.

As regards the acceptance criteria we use parameter c defining cooling rate and we use 4 parameters for weight adjustment of destroy and repair operators: $\sigma_1, \sigma_2, \sigma_3$ and r see section 6.2.5

7.1.2 ALNS parameters tuning results

We developed some experimental values for each parameter (see Table 2). Fine-tuning of parameters is conducted on the second phase by allowing one of parameters to take predefine values, while keeping the rest of parameters fixed. Since we know the number of values each parameter can take, we can calculate the total number of combinations with different parameters values. This is done by multiplication of all the numbers of values of all parameters. In total we have 864 different combinations. We have 10 tuning instances for parameters tuning experiments. Each instance is supposed to be run 5 times for each combination of parameters values. In total we have to run the algorithm 4 320 times. For each combination of parameters values we define the average deviation from the best found solution for each instance and then the average for all instances (within a combination).Each instance is run for 2000 iterations that takes in average 7 minutes. The total time taken to conduct the whole experiment for all instances and parameters setting is 84 hours. The best setting of parameters is provided in Table 3. The procedure of parameters tuning was automated.

Parameters	**Possible values**			
$\boldsymbol{r}$	0.2	0.4	0.6	
$\boldsymbol{\sigma 1}$	20	25		
$\boldsymbol{\sigma 2}$	15	20		
$\boldsymbol{\sigma 3}$	10	15		
$\boldsymbol{\alpha}$	0.25	0.5	0.75	
$\boldsymbol{\beta}$	0.25	0.5	0.25	
$\boldsymbol{Visits}$ (%)	10-15	12-17	15-20	17-20

Table 2 - Experimental value for each parameter

N	α	β	$\sigma 1$	$\sigma 2$	$\sigma 3$	r	LV	MV	Gap(%)
468	0.5	0.5	25	20	10	0.6	17	20	0.02

Table 3 - Best founded parameters setting

The results of all experiments of all parameters settings a provided in Appendix A

7.2 Results

This section provides results of the computational experiments, which were conducted, with the aim of testing the performance of the heuristic. For this purpose, we developed a set of test instances.

7.2.1 Test instances

All the test instances were generated based on the instance provided by the Statoil ASA, which contains 26 installations (the main instance). All these instances are divided in to two groups. The first group involves instances of the small and medium size (3 – 13 installations per instance). The second group involves instances with 14-26 installations per instance. All these instances were generated by gradually deleting installations from the main instance (one by one). In total, we have 10 small and medium size instances and 13 large size instances.

Location of all installations supplied from the supply base located in Mongstad is provided in the **Error! Reference source not found.** below.

7.2.2 Input data

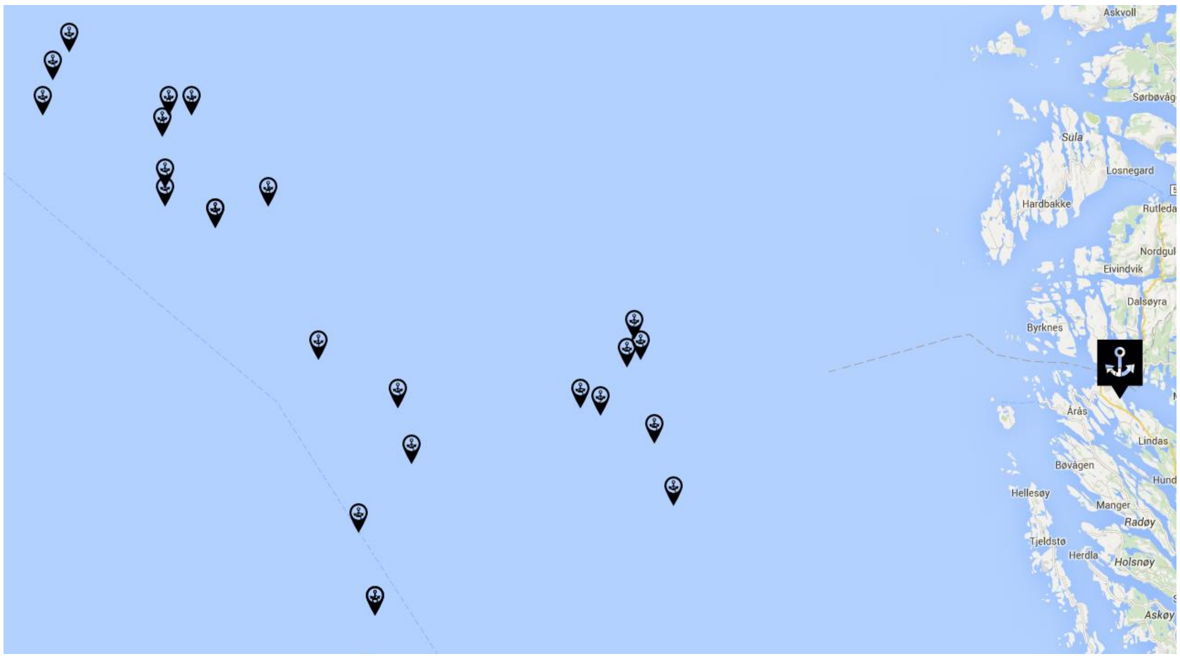


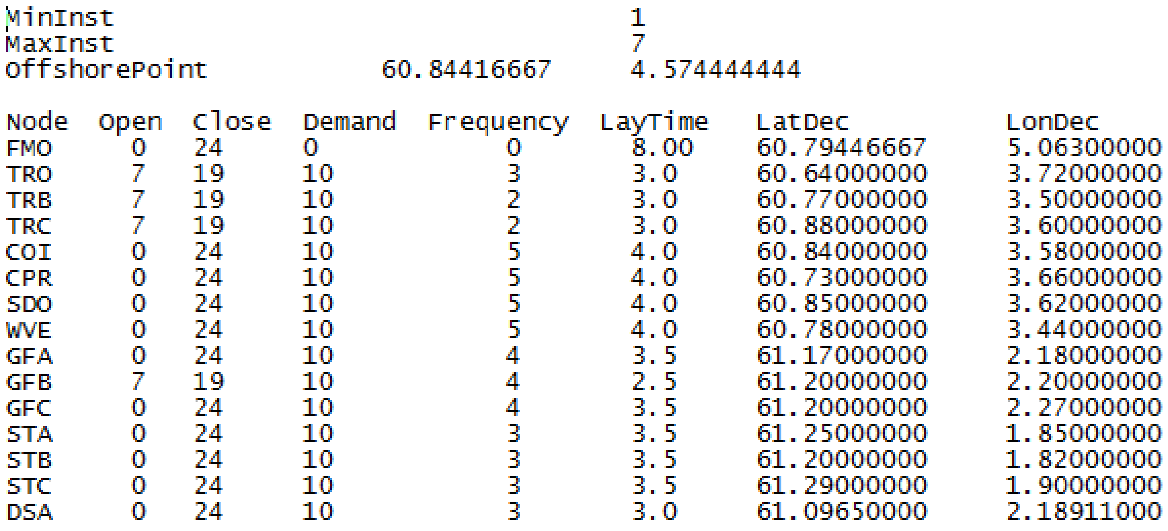
Figure 5 Location of offshore installations and Mongstad supply base.

In this section, we describe the input data to the algorithm on the example of the main instance (instance with 26 installations).

There are three input files used for the input data. The first file contains names of installations and supply base, open and closing times, demands, visits frequencies, service times (lay times) and as well coordinates of installations (LatDec and LonDec). In addition this file contains the values of minimum and maximum installations per voyage, coordinate of offshore point. There are three possible departures from the supply base within a day. Example of the input file is provided on the Figure 6.

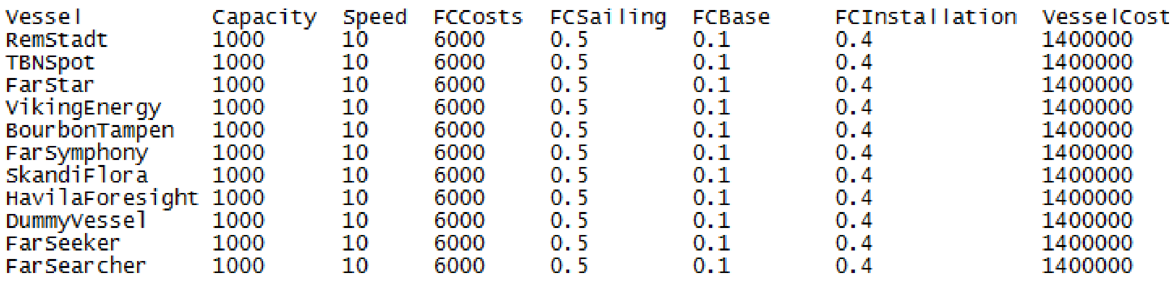
The second file contains information about the supply vessels fleet. For the list of supply vessels there is indicated its capacity, speed, fuel cost (NOK/ton), fuel consumption rates for sailing (ton/h), servicing and waiting at an installation (ton/h), at the supply base (ton/h). Example of this input file see on the Figure 7.

And the last input file contains feasible patterns of departures spread combinations for each visit frequency. See on the Figure 8.



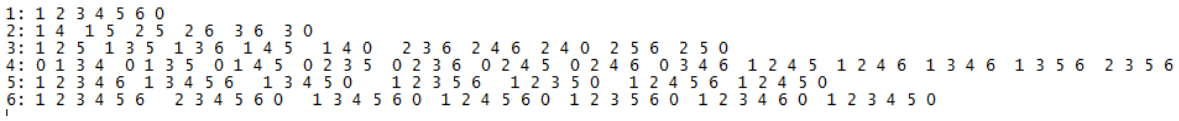
```
MinInst                                    1
MaxInst                                    7
OffshorePoint              60.84416667     4.574444444

Node   Open  Close  Demand  Frequency  LayTime  LatDec          LonDec
FMO     0    24     0          0        8.00    60.79446667     5.06300000
TRO     7    19     10         3        3.0     60.64000000     3.72000000
TRB     7    19     10         2        3.0     60.77000000     3.50000000
TRC     7    19     10         2        3.0     60.88000000     3.60000000
COI     0    24     10         5        4.0     60.84000000     3.58000000
CPR     0    24     10         5        4.0     60.73000000     3.66000000
SDO     0    24     10         5        4.0     60.85000000     3.62000000
WVE     0    24     10         5        4.0     60.78000000     3.44000000
GFA     0    24     10         4        3.5     61.17000000     2.18000000
GFB     7    19     10         4        2.5     61.20000000     2.20000000
GFC     0    24     10         4        3.5     61.20000000     2.27000000
STA     0    24     10         3        3.5     61.25000000     1.85000000
STB     0    24     10         3        3.5     61.20000000     1.82000000
STC     0    24     10         3        3.5     61.29000000     1.90000000
DSA     0    24     10         3        3.0     61.09650000     2.18911000
```

Figure 6 Input data example for offshore installations



```
Vessel          Capacity  Speed  FCCosts  FCSailing  FCBase   FCInstallation  VesselCost
RemStadt        1000      10     6000     0.5        0.1      0.4             1400000
TBNSpot         1000      10     6000     0.5        0.1      0.4             1400000
FarStar         1000      10     6000     0.5        0.1      0.4             1400000
VikingEnergy    1000      10     6000     0.5        0.1      0.4             1400000
BourbonTampen   1000      10     6000     0.5        0.1      0.4             1400000
FarSymphony     1000      10     6000     0.5        0.1      0.4             1400000
SkandiFlora     1000      10     6000     0.5        0.1      0.4             1400000
HavilaForesight 1000      10     6000     0.5        0.1      0.4             1400000
DummyVessel     1000      10     6000     0.5        0.1      0.4             1400000
FarSeeker       1000      10     6000     0.5        0.1      0.4             1400000
FarSearcher     1000      10     6000     0.5        0.1      0.4             1400000
```

Figure 7 Input data example for supply vessels.



```
1: 1 2 3 4 5 6 0
2: 1 4  1 5  2 5  2 6  3 6  3 0
3: 1 2 5  1 3 5  1 3 6  1 4 5  1 4 0  2 3 6  2 4 6  2 4 0  2 5 6  2 5 0
4: 0 1 3 4  0 1 3 5  0 1 4 5  0 2 3 5  0 2 3 6  0 2 4 5  0 2 4 6  0 3 4 6  1 2 4 5  1 2 4 6  1 3 4 6  1 3 5 6  2 3 5 6
5: 1 2 3 4 6  1 3 4 5 6  1 3 4 5 0  1 2 3 5 6  1 2 3 5 0  1 2 4 5 6  1 2 4 5 0
6: 1 2 3 4 5 6  2 3 4 5 6 0  1 3 4 5 6 0  1 2 4 5 6 0  1 2 3 5 6 0  1 2 3 4 6 0  1 2 3 4 5 0
```

Figure 8 Input data example for visit day's combinations.

7.2.3 Comparative analysis and results

In this section we provide the results of the conducted experiments and comments to them. All the tests were conducted with the use of the computer with following characteristics: 3.5. GHz Intel core i5 and 8 GB RAM. Mathematical model of the two phase approach was developed in AMPL (A Mathematical Programming Language) and run using solver CPLEX 12.6.0.0. Both route generation algorithm (for the two-phase approach) and

ALNS metaheuristic algorithm were programmed using C# programming language and .net 4.5 framework.

First we cover experiments for the small and medium size instances. In the **Error! Reference source not found.** provided results of the two phase approach and ALNS

Instance	Gap (%)	CPU (sec)		Number of Vessels	
	Two phase method vs ALNS	Two phase method	ALNS	Two phase method	ALNS
3-0-9	0	1	1	1	1
4-0-12	0	1	2	1	1
5-0-15	0	1	3	1	1
6-0-19	0	2	8	2	2
7-1-23	0	3	15	2	2
8-1-27	0	8	25	2	2
9-1-32	0	10	27	2	2
10-1-37	0.23	176	76	2	2
11-1-42	0.00	252	117	3	3
12-4-45	0.00	178	58	3	3
13-4-48	0.00	57	63	3	3
Average	**0.03**	**50.44**	**30.44**	**1.78**	**1.78**

Table 4 – Comparative analisis between two-phase approach and ALNS heuristic

heuristic.

The first column contains the names of all instances. Each name is compounded of several numbers. The first number defines the instance size interns of the number of installations. The second number define the number of installations with time windows, and the third number defines instance size interns of the total number of visits. The second column shows the gap in % between two phase method and ALNS heuristics objectives functions i.e. costs. As we see heuristic is able to provide optimal and near optimal solutions for small and medium size problems. There is no gap for instances with 3-9 and 11-13 installations, the heuristic manged to find optimal solutions. There is a minor gap for instance with 10 installations. The gap for this instance can be explained by a very narrow scope (neighbourhood) of solutions with two vessels. Such instances are so called “heavy instances”. The schedule for this instance is relatively tight and adding on more installation into schedule or increasing visit frequency of some existing installation in the schedule may lead to the fleet size increase. As we see from the results, the instance with 11 installations (and the rest larger instances) requires 3 vessels. Columns there and four (CPU sec) reflect the running time in seconds of the two-phase approach and ALNS heuristic. Two- phase approach performs faster (and this is obvious) on small size instances (3-9). Nevertheless, the situation changes for the medium size instances. As we see, the running time of the ALNS heuristic for medium size instances (10-13) is shorter compared to two-phase approach (in average twice). The last two columns provide the number of vessels used in the schedule provided by the two approaches. The number of vessels is equal for the same instance size. We may conclude that developed algorithm is able to provide optimal or near optimal solution for small and medium size instance within just a minute.

Now we discuss results of the experiments for large size instances. Unfortunately optimal solutions for large instances are unavailable since the problem complexity and as follows computational time grows exponentially with the problem size. Therefore, for the instances with 14-26 installations we conduct experiments for different number of iterations. The aim of such experiment is to define how the number of iterations influences the cost of the solution. And as well we find out the preferable number of iterations required to obtain the solution of a relatively good quality within a short time. For this we have to conduct trade-off analysis between objective values of solutions and running time of the algorithm.

We conducted experiments for 13 instances (14 -26 installations). For each instance the algorithm was run for different number of iterations, from 100 to 1000 with 100 interval. As well, we aimed to assess the stability of the results and for this purpose we run the algorithm **10 times** for each instance and number of iterations setup. The results of the experiments are summarized in the Appendix B. For each instance and for each setup of the number of iterations we defined the average cost, the average running time (for 10 runs) and the gap between the average cost and the cost of the best found solution for **all** setups of the number of iterations. In the table below (Table 5) we provide the excerpt from the Appendix B where for each instance showed only the cost of the best found solution, the number of vessels in the best found solution and the gap from this solution for each setup of the number of iterations. The table is performed in the form of the heat map. Small gaps are marked green and the colour gradually changes to bright red as the gap increases. As we see, the algorithm performs rather efficiently. In most cases (except instances with 17, 18 and 23 installations) the algorithm managed to find solutions which in average deviate from the best found less than 1%. Results with the gap less than 1% mean that for all 10 runs (for certain instance and number of iterations) the algorithm managed to find solutions with the same number of vessels as in the best found solution. A gap of 2.6-2.8 % means that in 1 of 10 runs the algorithm did not manage to drop the number of vessels to the minimum (under minimum we mean the number of vessels in the best found solution) and the number of vessels is by one vessel more than in the best found schedule. There were no solutions where the number of vessels is more by 2 than in the best found. Each gap increase by 2.6-2.8 % means reduction of the number of successful solutions (with minimal number of vessels) by one. Results with 26% gap mean that out of 10 runs there were not found any solutions with minimal number of vessels. As we see, the average gap reduces with increase of the number

Iterations	100	200	300	400	500	600	700	800	900	1000		Optimal number
Instance	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Optimal Costs	of vessels
14-4-51	0,26	0,15	0,28	0,08	0,10	0,03	0,08	0,07	0,02	0,02	5106776,97	3
15-4-54	0,55	0,64	0,32	0,35	0,29	0,19	0,21	0,29	0,21	0,18	5155404,36	3
16-4-58	11,29	2,98	0,44	2,86	0,19	0,15	0,10	0,16	0,08	0,14	5218878,34	3
17-5-61	21,84	24,39	8,54	3,44	11,09	5,67	0,62	3,18	0,37	3,06	5245374,24	3
18-5-64	26,40	26,36	26,31	18,46	23,67	18,44	21,02	26,24	23,57	20,95	5292785,34	3
19-6-66	0,52	0,37	0,31	0,29	0,33	0,33	0,25	0,32	0,27	0,22	6723965,22	4
20-6-72	0,38	0,30	0,21	0,15	0,17	0,16	0,15	0,15	0,14	0,10	6830353,28	4
21-6-77	0,55	0,41	0,43	0,37	0,34	0,36	0,30	0,30	0,27	0,29	6895713,99	4
22-6-81	2,75	0,63	4,56	2,41	0,48	0,32	0,34	0,34	0,26	0,29	6954724,85	4
23-7-84	14,64	8,64	8,52	8,52	6,50	6,51	6,50	2,41	2,48	0,41	6996699,15	4
24-7-87	0,59	0,51	0,38	0,34	0,36	0,35	0,31	0,26	0,24	0,27	8560334,23	5
25-8-88	0,41	0,32	0,26	0,27	0,21	0,21	0,22	0,15	0,19	0,16	8581096,14	5
26-8-91	0,59	0,51	0,48	0,44	0,33	0,34	0,36	0,29	0,26	0,31	8630445,26	5
Average	6,213276	5,092913	3,926709	2,922644	3,388671	2,544064	2,342772	2,627937	2,18184	2,031012		

Table 5 - A heat map of the gap from best-found solution with respect to the instance size and number of iterations

of iterations and the minimum gap is mostly achieved for 1000 iterations.

Special interest represent results for the instances with 17, 18 and 23 installations. The worst results are for the instance with 18 installations where the gap varies from 18 to 26 % that means that 7-10 runs out of 10 are unsuccessful. Those schedules where the algorithm managed to find solutions with minimal number of vessels turned out to be very tight (see example Figure 9) and very difficult for the algorithm to find. We refer the instances for which it is quite difficult to find a schedule with minimal number of vessels – «heavy instances». In most cases, a vessel should start loading/unloading operations just 10-15 minutes after it arrives to the base. The neighbourhood area of solutions with minimal number of vessels, for such instances, is very small and of course requires additional efforts for the algorithm to find a good solution. The gap reduction is achieved by increasing the number of iterations (as we see from the table). For instances with 17 and 23 installations, the gap reduced almost up to minimal when the number of iterations was set to 1000. Although for the instance with 18 installations such gap reduction is quite unclear (5%) that means we deal with a very tight schedule. One more interesting aspect, related to heavy instances, is that the solution for the next instance following the heavy instance (in terms of the number of installations) requires one vessel more. This fact further supports ours assumptions concerning so-called “heavy instances”. For heavy instance, increase of the number of installations at least by one (sometimes by two) or increase of the visit frequency leads to the increase of the fleet size. This is supported by experiments. For instances with 19 and 24 installation (which follow heavy instances with 18 and installations), the required fleet size 4 and 5 vessels respectively (that is one vessel more than for the instances with 18 and 23 installations).

NOK 5305405

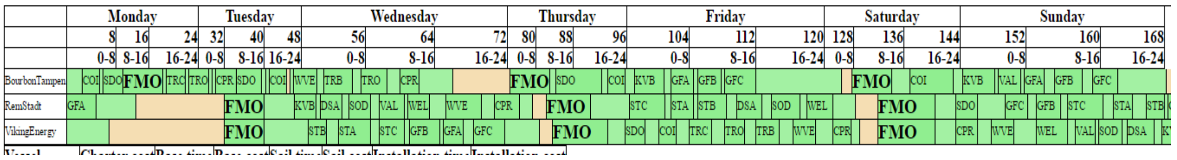


Figure 9 Example of tight schedule

As well we need to assess the running time of the algorithm. For this purpose, we analyse the larges instance (26 installations) and assess how changes the cost and computational time with increase of the number of iterations. On the Figure 10 Trade-off analyse for the instance with 26 installationsFigure 9 we see the results of the experiments conducted for different number of iterations. The *x-axis* corresponds to the number of iterations. The left *y*-axis correspond to the cost of the solution and the right y-axis corresponds to the running time of the algorithm. As we see, the highest cost (8837328) corresponds to the minimal number of iterations (30) and minimal computation time (32 sec). The lowest cost (8656772) is achieved for the maximal number of iterations (1000). However, such low cost is achieved at the expense of the computational time, which is 1048 sec. Both for minimal and maximal number of iteration the algorithm managed to find solution with minimal number of vessels. Therefore, selection of the number of iterations will affect only operational costs (sailing and servicing). We may observe a serious cost reduction from 30 to 40 iterations (by 129364 NOK). The difference between the cost of the solutions for 100 and 1000 is 36599. The dependence between the time and cost is linear and we state that **at least** 100 iterations (just 130sec) is preferably required to get a relatively good solution. Therefore, we assume that it is up to a researcher how many iteration to set. Taking into account the provided above analysis of the large size instances and presence of so-called “heavy instances”, requires running more iterations to insure sufficient fleet size reduction (since we do not know in advance which instance is “heavy”). From our point of view, taking into account quite high speed of the algorithm which is able to run 1000 iterations within 1000 sec (17 min), we recommend to run the algorithm for at least 1000 iterations to ensure good quality of a solution in case of heavy instance.

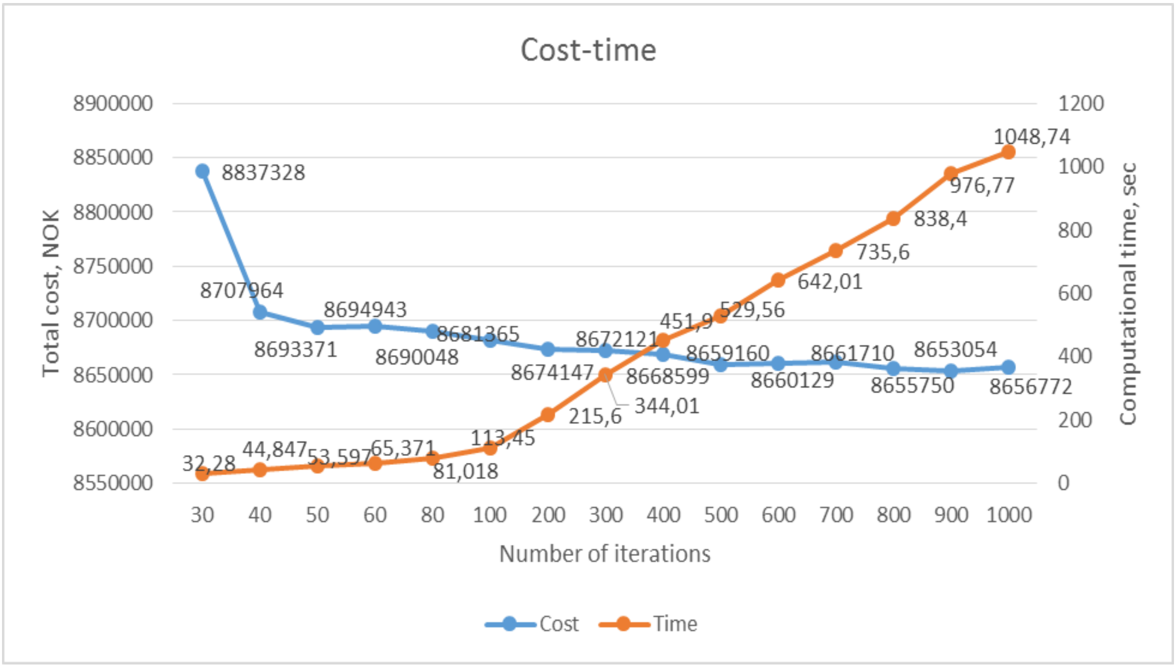


Figure 10 Trade-off analyse for the instance with 26 installations

In addition, Shyshou et al. (2012) kindly provided large size test instances (26-31 installations) which were used in their paper. We run ours algorithm for these instances and compared results. The results are summarized in the table 6. As we see, our algorithm provided better solutions for theses instance with the average gap 4.07 %. For instances with 27 and 28 installations the ALNS managed to find the solutions with fewer number of vessels (6 vessels compared to Shyshou et al. (2012) solutions which contain 7 vessels). The gaps in the objective function for these solutions are 11.38% and 10.07 % respectively. And as we see ours heuristic performs extremely faster compared to Shyshou et al. (2012). The average running time for these large instances is 427 seconds, while Shyshou et al. (2012) heuristic requires in average 13 795 sec, that is in average 32.3 times slower.

Instance	**Costs**		**Gap (%)**	**CPU (sec)**		**Number of Vessels**	
	LNS by Shyshou	**ALNS**	**LNS by vs ALNS**	**LNS by Shyshou**	**ALNS**	**LNS by Shyshou**	**ALNS**
26-94-5	5 603 570	5 553 327	0.9	12 712	95	6	6
27-98-5	6 458 500	5 723 607	11.38	10 403	263	**7**	6
28-102-5	6 553 680	5 953 712	10.07	22 584	1019	**7**	6
29-108-5	6 617 220	6 605 827	0.17	10 366	408	7	7
30-114-5	6 715 540	6 631 356	1.39	12 148	456	7	7
31-115-6	6 735 720	6 626 829	1.62	14 557	307	7	7
Average	**6 447 371**	**6 183 524**	**4.07**	**13 795**	**427**	**6.83**	**6.5**

Table 6-Comparison results of the Shushou LNS heuristic and represented ALNS heuristic for large size instances.

We may conclude that ours algorithm is able to find solutions of a better quality 30 times faster than that one developed by Shyshou et al. (2012).

8.0 Conclusions and further research

In the upstream offshore petroleum logistics, platform supply vessels (PSVs) are the main cost contributor. PSVs are used to deliver all the necessary material and equipment to offshore installations. Steady and uninterrupted supply is crucial for oil operators since the down time of an installation, in case some delay or disrupt, is enormous. The fleet of supply vessel sis associated with vessels charter and fuel costs. Therefore, there is trade-off between the service level and the cost of supply. To ensure high service level, sufficient number of vessels and theirs careful planning is required.

In this thesis, we to try to solve the problem of supply of the oil filed located in the North Sea and belonging to Statoil ASA, the largest oil operator in Norway. The oil field is supplied from the onshore supply based located in Mongstad and accounts for 26 installations. In the literature the problem is known as Periodic supply Vessel Planning Problem (PSVPP). The objective of PSVPP is to construct a weekly vessels schedule so that vessels charter cost and fuel cost is minimized. The problem is of a tactical level with planning horizon of one week. The problem involves three combinatorial optimization problem: packing (fleet size reduction), sequencing (routing) and scheduling (departures of vessels on voyages). Therefore, problems of large size is impossible to solve optimally within a reasonable time.

The objective of this thesis is to develop a decision support tool able to provide solutions of a good quality within a relatively short time. We studied existing literature dedicated to PSVPP and as well, some heuristic approaches to combinatorial problems. We selected Large Neighbourhood Heuristic (LNS) as a framework for ours algorithms and considered several known approaches to enhance its efficiency. As a starting point, we selected the LNS developed by Shyshou et al. (2012) for the PSVPP. We revised the heuristic, added several new procedures, improved existing and incorporated simulated annealing and adaptiveness framework. The resulted algorithm is referred to as Adaptive Large Neighbourhood Search (ALNS) heuristic.

Since we need to validate the resulted algorithm and check its performance, we developed two-phase exact approach based on set partitioning formulation of the PSVPP. The two-phase approach The ALNS was tested by comparing solutions obtained for small and medium size instances to those obtained by using the two-phase exact approach. The ALNS proved to be quite efficient both in terms of costs and computational times compared to exact approach. For most instances the heuristic managed to find optimal or near optimal solutions within rather short time and thus outperforming the exact approach. Since it is hardly possible to compare Heuristic and exact approach for large size instances, we compared ours ALNS to the LNS developed by (Shyshou et al. 2012)The results of the experiments state that ours algorithm is able to find better solutions 30 times faster than the LNS by Shyshou et al. Furthermore, for several instances ours algorithm manged to find solution with fewer number of vessels.

As well, we outline several directions for the future research. First, there is need to improve the efficiency of the heuristic to provide good solution for heavy instances i.e. instances for which resulted schedules are very tight. As experiment showed, the algorithm does not always mange to reduce the fleet size to the minimum for some instances (large size). Especial feature of such instances is that increase of the instance just by one installation (adding of a new one) or increase of the visit frequency of some existing inevitably lead to the fleet size increase. The resulting schedule with minimal number of vessels for such instances is very tight and as follows is very difficult to find. Therefore, some work should be conducted to improve the algorithm to search for good solution for such types of instances. (see (Ahuja et al. 2002))

As well, there is a need to incorporate some instrument allowing for generation of a robust solution to cope with weather uncertainty. Too tight schedules are inapplicable in practice and therefore some approach is needed to cope with uncertainty (see (Maisiuk and Gribkovskaia 2014); (Vlachos 2004))

Furthermore, in practice there are often cooperation between supply bases i.e. when a vessel starts a voyage at one bases and finishes at another. As well, there is a problem of distribution of installations between bases i.e. from which supply base it more efficient to serve some installations (especially those equally in between bases). For this reason, the algorithm should involve the possibility to construct schedules for several bases simultaneously. The problem then becomes multi base (see (Crevier, Cordeau, and Laporte 2007); (Cordeau, Gendreau, and Laporte 1997))

References

Ahuja, Ravindra K, Özlem Ergun, James B Orlin, and Abraham P Punnen. 2002. "A survey of very large-scale neighborhood search techniques." *Discrete Applied Mathematics* 123 (1):75-102.

Bräysy, Olli, and Michel Gendreau. 2005. "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms." *Transportation science* 39 (1):104-118.

Cordeau, Jean-François, Michel Gendreau, and Gilbert Laporte. 1997. "A tabu search heuristic for periodic and multi-depot vehicle routing problems." *Networks* 30 (2):105-119.

Crevier, Benoit, Jean-François Cordeau, and Gilbert Laporte. 2007. "The multi-depot vehicle routing problem with inter-depot routes." *European Journal of Operational Research* 176 (2):756-773.

Fagerholt, K., and H. Lindstad. 2000. "Optimal policies for maintaining a supply service in the Norwegian Sea." *Omega-International Journal of Management Science* 28 (3):269-275. doi: Doi 10.1016/S0305-0483(99)00054-7.

Gribkovskaia, I., G. Laporte, and A. Shlopak. 2008. "A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms." *Journal of the Operational Research Society* 59 (11):1449-1459. doi: 10.1057/palgrave.jors.2602469.

Halvorsen-Weare, Elin E, and Kjetil Fagerholt. 2011. "Robust supply vessel planning." In *Network optimization*, 559-573. Springer.

Halvorsen-Weare, Elin E, Kjetil Fagerholt, Lars Magne Nonås, and Bjørn Egil Asbjørnslett. 2012. "Optimal fleet composition and periodic routing of offshore supply vessels." *European Journal of Operational Research* 223 (2):508-517.

Korsvik, J. E., and K. Fagerholt. 2010. "A tabu search heuristic for ship routing and scheduling with flexible cargo quantities." *Journal of Heuristics* 16 (2):117-137.

Maisiuk, Yauhen, and Irina Gribkovskaia. 2014. "Fleet Sizing for Offshore Supply Vessels with Stochastic Sailing and Service Times." *Procedia Computer Science* 31:939-948.

Norlund, E. K., and I. Gribkovskaia. 2013. "Reducing emissions through speed optimization in supply vessel operations (vol 23, pg 105, 2013)." *Transportation Research Part D-Transport and Environment* 24:135-135. doi: 10.1016/j.trd.2013.09.001.

Pisinger, David, and Stefan Ropke. 2010. "Large neighborhood search." In *Handbook of metaheuristics*, 399-419. Springer.

Ropke, S., and D. Pisinger. 2006. "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows." *Transportation Science* 40 (4):455-472. doi: 10.1287/trsc.1050.0135.

Shaw, P. 1998. "Using constraint programming and local search methods to solve vehicle routing problems." *Principles and Practice of Constraint Programming - Cp98* 1520:417-431.

Shaw, Paul. 1997. "A new local search algorithm providing high quality solutions to vehicle routing problems." *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.*

Shyshou, A., I. Gribkovskaia, G. Laporte, and K. Fagerholt. 2012. "A Large Neighbourhood Search Heuristic for a Periodic Supply Vessel Planning Problem Arising in Offshore Oil and Gas Operations." *Infor* 50 (4):195-204. doi: 10.3138/infor.50.4.195.

Statoil, ASA. 2016. Statoil Annual Report on Form 20-F

Vlachos, DS. 2004. "Optimal ship routing based on wind and wave forecasts." *Applied Numerical Analysis & Computational Mathematics* 1 (2):547-551.

Appendix A.

Results obtained by ALNS heuristic for parameters tuning.

N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV
1	3.34	0.25	0.25	20	15	10	0.2	10	15
2	0.19	0.25	0.25	20	15	10	0.2	12	17
3	3.33	0.25	0.25	20	15	10	0.2	15	20
4	0.13	0.25	0.25	20	15	10	0.2	17	20
5	4.83	0.25	0.25	20	15	10	0.4	10	15
6	6.42	0.25	0.25	20	15	10	0.4	12	17
7	0.18	0.25	0.25	20	15	10	0.4	15	20
8	1.70	0.25	0.25	20	15	10	0.4	17	20
9	1.77	0.25	0.25	20	15	10	0.6	10	15
10	0.18	0.25	0.25	20	15	10	0.6	12	17
11	1.77	0.25	0.25	20	15	10	0.6	15	20
12	1.69	0.25	0.25	20	15	10	0.6	17	20
13	6.51	0.25	0.25	20	15	15	0.2	10	15
14	4.95	0.25	0.25	20	15	15	0.2	12	17
15	7.97	0.25	0.25	20	15	15	0.2	15	20
16	6.39	0.25	0.25	20	15	15	0.2	17	20
17	4.93	0.25	0.25	20	15	15	0.4	10	15
18	4.98	0.25	0.25	20	15	15	0.4	12	17
19	6.44	0.25	0.25	20	15	15	0.4	15	20
20	6.51	0.25	0.25	20	15	15	0.4	17	20
21	0.15	0.25	0.25	20	15	15	0.6	10	15
22	6.40	0.25	0.25	20	15	15	0.6	12	17
23	3.36	0.25	0.25	20	15	15	0.6	15	20
24	6.49	0.25	0.25	20	15	15	0.6	17	20
25	8.09	0.25	0.25	20	20	10	0.2	10	15
26	11.1	0.25	0.25	20	20	10	0.2	12	17
27	0.31	0.25	0.25	20	20	10	0.2	15	20
28	1.77	0.25	0.25	20	20	10	0.2	17	20
29	8.12	0.25	0.25	20	20	10	0.4	10	15
30	1.70	0.25	0.25	20	20	10	0.4	12	17
31	6.41	0.25	0.25	20	20	10	0.4	15	20
32	3.32	0.25	0.25	20	20	10	0.4	17	20
33	1.69	0.25	0.25	20	20	10	0.6	10	15
34	8.03	0.25	0.25	20	20	10	0.6	12	17
35	0.26	0.25	0.25	20	20	10	0.6	15	20
36	4.89	0.25	0.25	20	20	10	0.6	17	20
37	4.86	0.25	0.25	20	20	15	0.2	10	15
38	9.69	0.25	0.25	20	20	15	0.2	12	17
39	4.87	0.25	0.25	20	20	15	0.2	15	20
40	6.45	0.25	0.25	20	20	15	0.2	17	20
41	6.46	0.25	0.25	20	20	15	0.4	10	15
42	0.16	0.25	0.25	20	20	15	0.4	12	17
43	1.70	0.25	0.25	20	20	15	0.4	15	20
44	1.74	0.25	0.25	20	20	15	0.4	17	20
45	1.86	0.25	0.25	20	20	15	0.6	10	15
46	1.85	0.25	0.25	20	20	15	0.6	12	17
47	1.66	0.25	0.25	20	20	15	0.6	15	20
48	0.26	0.25	0.25	20	20	15	0.6	17	20
49	8.02	0.25	0.25	25	15	10	0.2	10	15
50	9.57	0.25	0.25	25	15	10	0.2	12	17
51	8.07	0.25	0.25	25	15	10	0.2	15	20
52	4.85	0.25	0.25	25	15	10	0.2	17	20
53	6.47	0.25	0.25	25	15	10	0.4	10	15
54	1.74	0.25	0.25	25	15	10	0.4	12	17
55	1.68	0.25	0.25	25	15	10	0.4	15	20
56	4.87	0.25	0.25	25	15	10	0.4	17	20
57	3.35	0.25	0.25	25	15	10	0.6	10	15
58	0.10	0.25	0.25	25	15	10	0.6	12	17
59	6.42	0.25	0.25	25	15	10	0.6	15	20
60	6.39	0.25	0.25	25	15	10	0.6	17	20
61	11.2	0.25	0.25	25	15	15	0.2	10	15
62	6.52	0.25	0.25	25	15	15	0.2	12	17
63	3.33	0.25	0.25	25	15	15	0.2	15	20
64	3.27	0.25	0.25	25	15	15	0.2	17	20
65	9.61	0.25	0.25	25	15	15	0.4	10	15
66	6.49	0.25	0.25	25	15	15	0.4	12	17
67	0.07	0.25	0.25	25	15	15	0.4	15	20
68	4.79	0.25	0.25	25	15	15	0.4	17	20
69	0.12	0.25	0.25	25	15	15	0.6	10	15
70	14.4	0.25	0.25	25	15	15	0.6	12	17
71	8.03	0.25	0.25	25	15	15	0.6	15	20
72	0.11	0.25	0.25	25	15	15	0.6	17	20
73	0.08	0.25	0.25	25	20	10	0.2	10	15
74	3.20	0.25	0.25	25	20	10	0.2	12	17
75	6.53	0.25	0.25	25	20	10	0.2	15	20
76	3.30	0.25	0.25	25	20	10	0.2	17	20
433	0.18	0.5	0.5	25	15	10	0.2	10	15
434	8.01	0.5	0.5	25	15	10	0.2	12	17
435	0.15	0.5	0.5	25	15	10	0.2	15	20
436	0.13	0.5	0.5	25	15	10	0.2	17	20
437	4.98	0.5	0.5	25	15	10	0.4	10	15
438	1.71	0.5	0.5	25	15	10	0.4	12	17
439	0.18	0.5	0.5	25	15	10	0.4	15	20
440	9.67	0.5	0.5	25	15	10	0.4	17	20
441	9.70	0.5	0.5	25	15	10	0.6	10	15
442	8.06	0.5	0.5	25	15	10	0.6	12	17
443	7.96	0.5	0.5	25	15	10	0.6	15	20
444	4.88	0.5	0.5	25	15	10	0.6	17	20
445	14.4	0.5	0.5	25	15	15	0.2	10	15
446	0.23	0.5	0.5	25	15	15	0.2	12	17
447	1.81	0.5	0.5	25	15	15	0.2	15	20
448	4.86	0.5	0.5	25	15	15	0.2	17	20
449	0.29	0.5	0.5	25	15	15	0.4	10	15
450	3.34	0.5	0.5	25	15	15	0.4	12	17
451	0.32	0.5	0.5	25	15	15	0.4	15	20
452	7.96	0.5	0.5	25	15	15	0.4	17	20
453	0.22	0.5	0.5	25	15	15	0.6	10	15
454	9.74	0.5	0.5	25	15	15	0.6	12	17
455	0.14	0.5	0.5	25	15	15	0.6	15	20
456	6.43	0.5	0.5	25	15	15	0.6	17	20
457	9.62	0.5	0.5	25	20	10	0.2	10	15
458	4.91	0.5	0.5	25	20	10	0.2	12	17
459	4.89	0.5	0.5	25	20	10	0.2	15	20
460	0.19	0.5	0.5	25	20	10	0.2	17	20
461	4.80	0.5	0.5	25	20	10	0.4	10	15
462	3.40	0.5	0.5	25	20	10	0.4	12	17
463	1.70	0.5	0.5	25	20	10	0.4	15	20
464	6.49	0.5	0.5	25	20	10	0.4	17	20
465	0.11	0.5	0.5	25	20	10	0.6	10	15
466	6.48	0.5	0.5	25	20	10	0.6	12	17
467	0.22	0.5	0.5	25	20	10	0.6	15	20
468	**0.02**	**0.5**	**0.5**	**25**	**20**	**10**	**0.6**	**17**	**20**
469	1.79	0.5	0.5	25	20	15	0.2	10	15
470	4.95	0.5	0.5	25	20	15	0.2	12	17
471	6.43	0.5	0.5	25	20	15	0.2	15	20
472	8.01	0.5	0.5	25	20	15	0.2	17	20
473	1.67	0.5	0.5	25	20	15	0.4	10	15
474	3.35	0.5	0.5	25	20	15	0.4	12	17
475	0.15	0.5	0.5	25	20	15	0.4	15	20
476	6.49	0.5	0.5	25	20	15	0.4	17	20
477	3.31	0.5	0.5	25	20	15	0.6	10	15
478	8.19	0.5	0.5	25	20	15	0.6	12	17
479	1.68	0.5	0.5	25	20	15	0.6	15	20
480	1.78	0.5	0.5	25	20	15	0.6	17	20
481	6.44	0.5	0.75	20	15	10	0.2	10	15
482	4.87	0.5	0.75	20	15	10	0.2	12	17
483	1.78	0.5	0.75	20	15	10	0.2	15	20
484	0.09	0.5	0.75	20	15	10	0.2	17	20
485	3.39	0.5	0.75	20	15	10	0.4	10	15
486	8.07	0.5	0.75	20	15	10	0.4	12	17
487	6.41	0.5	0.75	20	15	10	0.4	15	20
488	6.45	0.5	0.75	20	15	10	0.4	17	20
489	6.48	0.5	0.75	20	15	10	0.6	10	15
490	1.67	0.5	0.75	20	15	10	0.6	12	17
491	4.93	0.5	0.75	20	15	10	0.6	15	20
492	6.40	0.5	0.75	20	15	10	0.6	17	20
493	11.2	0.5	0.75	20	15	15	0.2	10	15
494	6.40	0.5	0.75	20	15	15	0.2	12	17
495	8.04	0.5	0.75	20	15	15	0.2	15	20
496	8.02	0.5	0.75	20	15	15	0.2	17	20
497	4.86	0.5	0.75	20	15	15	0.4	10	15
498	0.14	0.5	0.75	20	15	15	0.4	12	17
499	4.89	0.5	0.75	20	15	15	0.4	15	20
500	0.12	0.5	0.75	20	15	15	0.4	17	20
501	1.88	0.5	0.75	20	15	15	0.6	10	15
502	0.23	0.5	0.75	20	15	15	0.6	12	17
503	4.85	0.5	0.75	20	15	15	0.6	15	20
504	8.05	0.5	0.75	20	15	15	0.6	17	20
505	0.21	0.5	0.75	20	20	10	0.2	10	15
506	6.50	0.5	0.75	20	20	10	0.2	12	17
507	3.39	0.5	0.75	20	20	10	0.2	15	20
508	4.81	0.5	0.75	20	20	10	0.2	17	20

N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV	N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV
77	3.41	0.25	0.25	25	20	10	0.4	10	15	509	0.31	0.5	0.75	20	20	10	0.4	10	15
78	1.75	0.25	0.25	25	20	10	0.4	12	17	510	0.10	0.5	0.75	20	20	10	0.4	12	17
79	0.25	0.25	0.25	25	20	10	0.4	15	20	511	8.08	0.5	0.75	20	20	10	0.4	15	20
80	4.86	0.25	0.25	25	20	10	0.4	17	20	512	1.79	0.5	0.75	20	20	10	0.4	17	20
81	3.31	0.25	0.25	25	20	10	0.6	10	15	513	6.49	0.5	0.75	20	20	10	0.6	10	15
82	1.64	0.25	0.25	25	20	10	0.6	12	17	514	0.18	0.5	0.75	20	20	10	0.6	12	17
83	8.03	0.25	0.25	25	20	10	0.6	15	20	515	0.12	0.5	0.75	20	20	10	0.6	15	20
84	3.26	0.25	0.25	25	20	10	0.6	17	20	516	3.35	0.5	0.75	20	20	10	0.6	17	20
85	0.34	0.25	0.25	25	20	15	0.2	10	15	517	6.48	0.5	0.75	20	20	15	0.2	10	15
86	0.22	0.25	0.25	25	20	15	0.2	12	17	518	1.73	0.5	0.75	20	20	15	0.2	12	17
87	0.24	0.25	0.25	25	20	15	0.2	15	20	519	6.39	0.5	0.75	20	20	15	0.2	15	20
88	11.2	0.25	0.25	25	20	15	0.2	17	20	520	0.27	0.5	0.75	20	20	15	0.2	17	20
89	8.12	0.25	0.25	25	20	15	0.4	10	15	521	3.42	0.5	0.75	20	20	15	0.4	10	15
90	0.14	0.25	0.25	25	20	15	0.4	12	17	522	1.80	0.5	0.75	20	20	15	0.4	12	17
91	0.22	0.25	0.25	25	20	15	0.4	15	20	523	0.11	0.5	0.75	20	20	15	0.4	15	20
92	6.48	0.25	0.25	25	20	15	0.4	17	20	524	8.00	0.5	0.75	20	20	15	0.4	17	20
93	0.25	0.25	0.25	25	20	15	0.6	10	15	525	15.9	0.5	0.75	20	20	15	0.6	10	15
94	6.46	0.25	0.25	25	20	15	0.6	12	17	526	4.92	0.5	0.75	20	20	15	0.6	12	17
95	0.19	0.25	0.25	25	20	15	0.6	15	20	527	0.16	0.5	0.75	20	20	15	0.6	15	20
96	0.14	0.25	0.25	25	20	15	0.6	17	20	528	6.46	0.5	0.75	20	20	15	0.6	17	20
97	4.88	0.25	0.5	20	15	10	0.2	10	15	529	6.43	0.5	0.75	25	15	10	0.2	10	15
98	6.46	0.25	0.5	20	15	10	0.2	12	17	530	0.18	0.5	0.75	25	15	10	0.2	12	17
99	1.66	0.25	0.5	20	15	10	0.2	15	20	531	0.18	0.5	0.75	25	15	10	0.2	15	20
100	6.38	0.25	0.5	20	15	10	0.2	17	20	532	3.31	0.5	0.75	25	15	10	0.2	17	20
101	8.08	0.25	0.5	20	15	10	0.4	10	15	533	11.3	0.5	0.75	25	15	10	0.4	10	15
102	3.32	0.25	0.5	20	15	10	0.4	12	17	534	0.20	0.5	0.75	25	15	10	0.4	12	17
103	0.21	0.25	0.5	20	15	10	0.4	15	20	535	1.62	0.5	0.75	25	15	10	0.4	15	20
104	1.74	0.25	0.5	20	15	10	0.4	17	20	536	6.39	0.5	0.75	25	15	10	0.4	17	20
105	11.3	0.25	0.5	20	15	10	0.6	10	15	537	0.26	0.5	0.75	25	15	10	0.6	10	15
106	9.72	0.25	0.5	20	15	10	0.6	12	17	538	0.20	0.5	0.75	25	15	10	0.6	12	17
107	6.52	0.25	0.5	20	15	10	0.6	15	20	539	3.25	0.5	0.75	25	15	10	0.6	15	20
108	1.71	0.25	0.5	20	15	10	0.6	17	20	540	3.30	0.5	0.75	25	15	10	0.6	17	20
109	0.11	0.25	0.5	20	15	15	0.2	10	15	541	3.47	0.5	0.75	25	15	15	0.2	10	15
110	6.44	0.25	0.5	20	15	15	0.2	12	17	542	0.19	0.5	0.75	25	15	15	0.2	12	17
111	3.25	0.25	0.5	20	15	15	0.2	15	20	543	3.34	0.5	0.75	25	15	15	0.2	15	20
112	4.93	0.25	0.5	20	15	15	0.2	17	20	544	0.12	0.5	0.75	25	15	15	0.2	17	20
113	6.39	0.25	0.5	20	15	15	0.4	10	15	545	3.36	0.5	0.75	25	15	15	0.4	10	15
114	4.92	0.25	0.5	20	15	15	0.4	12	17	546	0.21	0.5	0.75	25	15	15	0.4	12	17
115	1.80	0.25	0.5	20	15	15	0.4	15	20	547	0.21	0.5	0.75	25	15	15	0.4	15	20
116	0.11	0.25	0.5	20	15	15	0.4	17	20	548	4.90	0.5	0.75	25	15	15	0.4	17	20
117	8.03	0.25	0.5	20	15	15	0.6	10	15	549	8.02	0.5	0.75	25	15	15	0.6	10	15
118	0.32	0.25	0.5	20	15	15	0.6	12	17	550	6.41	0.5	0.75	25	15	15	0.6	12	17
119	4.78	0.25	0.5	20	15	15	0.6	15	20	551	1.80	0.5	0.75	25	15	15	0.6	15	20
120	0.18	0.25	0.5	20	15	15	0.6	17	20	552	1.75	0.5	0.75	25	15	15	0.6	17	20
121	6.52	0.25	0.5	20	20	10	0.2	10	15	553	11.2	0.5	0.75	25	20	10	0.2	10	15
122	1.93	0.25	0.5	20	20	10	0.2	12	17	554	9.64	0.5	0.75	25	20	10	0.2	12	17
123	8.00	0.25	0.5	20	20	10	0.2	15	20	555	9.68	0.5	0.75	25	20	10	0.2	15	20
124	0.17	0.25	0.5	20	20	10	0.2	17	20	556	4.85	0.5	0.75	25	20	10	0.2	17	20
125	1.77	0.25	0.5	20	20	10	0.4	10	15	557	3.33	0.5	0.75	25	20	10	0.4	10	15
126	1.70	0.25	0.5	20	20	10	0.4	12	17	558	0.18	0.5	0.75	25	20	10	0.4	12	17
127	1.70	0.25	0.5	20	20	10	0.4	15	20	559	6.46	0.5	0.75	25	20	10	0.4	15	20
128	1.73	0.25	0.5	20	20	10	0.4	17	20	560	9.61	0.5	0.75	25	20	10	0.4	17	20
129	4.88	0.25	0.5	20	20	10	0.6	10	15	561	3.37	0.5	0.75	25	20	10	0.6	10	15
130	4.88	0.25	0.5	20	20	10	0.6	12	17	562	3.35	0.5	0.75	25	20	10	0.6	12	17
131	6.41	0.25	0.5	20	20	10	0.6	15	20	563	0.08	0.5	0.75	25	20	10	0.6	15	20
132	4.93	0.25	0.5	20	20	10	0.6	17	20	564	0.13	0.5	0.75	25	20	10	0.6	17	20
133	9.57	0.25	0.5	20	20	15	0.2	10	15	565	1.75	0.5	0.75	25	20	15	0.2	10	15
134	4.77	0.25	0.5	20	20	15	0.2	12	17	566	4.76	0.5	0.75	25	20	15	0.2	12	17
135	3.26	0.25	0.5	20	20	15	0.2	15	20	567	0.20	0.5	0.75	25	20	15	0.2	15	20
136	0.14	0.25	0.5	20	20	15	0.2	17	20	568	0.19	0.5	0.75	25	20	15	0.2	17	20
137	4.93	0.25	0.5	20	20	15	0.4	10	15	569	3.31	0.5	0.75	25	20	15	0.4	10	15
138	9.63	0.25	0.5	20	20	15	0.4	12	17	570	3.35	0.5	0.75	25	20	15	0.4	12	17
139	6.46	0.25	0.5	20	20	15	0.4	15	20	571	1.76	0.5	0.75	25	20	15	0.4	15	20
140	6.46	0.25	0.5	20	20	15	0.4	17	20	572	8.09	0.5	0.75	25	20	15	0.4	17	20
141	3.45	0.25	0.5	20	20	15	0.6	10	15	573	6.48	0.5	0.75	25	20	15	0.6	10	15
142	6.53	0.25	0.5	20	20	15	0.6	12	17	574	0.27	0.5	0.75	25	20	15	0.6	12	17
143	1.68	0.25	0.5	20	20	15	0.6	15	20	575	1.69	0.5	0.75	25	20	15	0.6	15	20
144	1.64	0.25	0.5	20	20	15	0.6	17	20	576	3.29	0.5	0.75	25	20	15	0.6	17	20
145	1.75	0.25	0.5	25	15	10	0.2	10	15	513	6.49	0.5	0.75	20	20	10	0.6	10	15
146	8.02	0.25	0.5	25	15	10	0.2	12	17	577	1.70	0.75	0.25	20	15	10	0.2	10	15
147	0.22	0.25	0.5	25	15	10	0.2	15	20	578	1.77	0.75	0.25	20	15	10	0.2	12	17
148	1.73	0.25	0.5	25	15	10	0.2	17	20	579	0.15	0.75	0.25	20	15	10	0.2	15	20
149	1.72	0.25	0.5	25	15	10	0.4	10	15	580	1.75	0.75	0.25	20	15	10	0.2	17	20
150	8.05	0.25	0.5	25	15	10	0.4	12	17	581	9.61	0.75	0.25	20	15	10	0.4	10	15
151	6.37	0.25	0.5	25	15	10	0.4	15	20	582	6.45	0.75	0.25	20	15	10	0.4	12	17
152	0.11	0.25	0.5	25	15	10	0.4	17	20	583	3.37	0.75	0.25	20	15	10	0.4	15	20
153	8.02	0.25	0.5	25	15	10	0.6	10	15	584	6.39	0.75	0.25	20	15	10	0.4	17	20
154	1.68	0.25	0.5	25	15	10	0.6	12	17	585	8.13	0.75	0.25	20	15	10	0.6	10	15
155	6.40	0.25	0.5	25	15	10	0.6	15	20	586	3.41	0.75	0.25	20	15	10	0.6	12	17
156	6.47	0.25	0.5	25	15	10	0.6	17	20	587	1.75	0.75	0.25	20	15	10	0.6	15	20
157	0.12	0.25	0.5	25	15	15	0.2	10	15	588	1.66	0.75	0.25	20	15	10	0.6	17	20

N	Gap	α	β	$\sigma 1$	$\sigma 2$	$\sigma 3$	r	LV	MV	N	Gap	α	β	$\sigma 1$	$\sigma 2$	$\sigma 3$	r	LV	MV
158	1.75	0.25	0.5	25	15	15	0.2	12	17	589	1.86	0.75	0.25	20	15	15	0.2	10	15
159	11.3	0.25	0.5	25	15	15	0.2	15	20	590	4.92	0.75	0.25	20	15	15	0.2	12	17
160	3.33	0.25	0.5	25	15	15	0.2	17	20	591	6.47	0.75	0.25	20	15	15	0.2	15	20
161	8.13	0.25	0.5	25	15	15	0.4	10	15	592	8.08	0.75	0.25	20	15	15	0.2	17	20
162	0.12	0.25	0.5	25	15	15	0.4	12	17	593	5.04	0.75	0.25	20	15	15	0.4	10	15
163	1.78	0.25	0.5	25	15	15	0.4	15	20	594	8.07	0.75	0.25	20	15	15	0.4	12	17
164	4.91	0.25	0.5	25	15	15	0.4	17	20	595	0.11	0.75	0.25	20	15	15	0.4	15	20
165	4.96	0.25	0.5	25	15	15	0.6	10	15	596	1.68	0.75	0.25	20	15	15	0.4	17	20
166	6.52	0.25	0.5	25	15	15	0.6	12	17	597	4.97	0.75	0.25	20	15	15	0.6	10	15
167	1.74	0.25	0.5	25	15	15	0.6	15	20	598	6.40	0.75	0.25	20	15	15	0.6	12	17
168	1.77	0.25	0.5	25	15	15	0.6	17	20	599	6.44	0.75	0.25	20	15	15	0.6	15	20
169	4.96	0.25	0.5	25	20	10	0.2	10	15	600	6.49	0.75	0.25	20	15	15	0.6	17	20
170	4.93	0.25	0.5	25	20	10	0.2	12	17	601	4.88	0.75	0.25	20	20	10	0.2	10	15
171	8.00	0.25	0.5	25	20	10	0.2	15	20	602	6.41	0.75	0.25	20	20	10	0.2	12	17
172	0.20	0.25	0.5	25	20	10	0.2	17	20	603	0.08	0.75	0.25	20	20	10	0.2	15	20
173	8.05	0.25	0.5	25	20	10	0.4	10	15	604	0.10	0.75	0.25	20	20	10	0.2	17	20
174	6.42	0.25	0.5	25	20	10	0.4	12	17	605	8.06	0.75	0.25	20	20	10	0.4	10	15
175	8.09	0.25	0.5	25	20	10	0.4	15	20	606	1.68	0.75	0.25	20	20	10	0.4	12	17
176	1.71	0.25	0.5	25	20	10	0.4	17	20	607	0.08	0.75	0.25	20	20	10	0.4	15	20
177	4.97	0.25	0.5	25	20	10	0.6	10	15	608	1.79	0.75	0.25	20	20	10	0.4	17	20
178	1.69	0.25	0.5	25	20	10	0.6	12	17	609	3.43	0.75	0.25	20	20	10	0.6	10	15
179	0.20	0.25	0.5	25	20	10	0.6	15	20	610	4.80	0.75	0.25	20	20	10	0.6	12	17
180	6.40	0.25	0.5	25	20	10	0.6	17	20	611	1.69	0.75	0.25	20	20	10	0.6	15	20
181	11.1	0.25	0.5	25	20	15	0.2	10	15	612	6.43	0.75	0.25	20	20	10	0.6	17	20
182	0.14	0.25	0.5	25	20	15	0.2	12	17	613	8.03	0.75	0.25	20	20	15	0.2	10	15
183	0.16	0.25	0.5	25	20	15	0.2	15	20	614	1.68	0.75	0.25	20	20	15	0.2	12	17
184	3.24	0.25	0.5	25	20	15	0.2	17	20	615	1.78	0.75	0.25	20	20	15	0.2	15	20
185	3.31	0.25	0.5	25	20	15	0.4	10	15	616	1.77	0.75	0.25	20	20	15	0.2	17	20
186	3.16	0.25	0.5	25	20	15	0.4	12	17	617	0.20	0.75	0.25	20	20	15	0.4	10	15
187	0.11	0.25	0.5	25	20	15	0.4	15	20	618	0.19	0.75	0.25	20	20	15	0.4	12	17
188	1.85	0.25	0.5	25	20	15	0.4	17	20	619	0.06	0.75	0.25	20	20	15	0.4	15	20
189	1.69	0.25	0.5	25	20	15	0.6	10	15	620	4.97	0.75	0.25	20	20	15	0.4	17	20
190	4.95	0.25	0.5	25	20	15	0.6	12	17	621	7.99	0.75	0.25	20	20	15	0.6	10	15
191	0.15	0.25	0.5	25	20	15	0.6	15	20	622	4.84	0.75	0.25	20	20	15	0.6	12	17
192	9.60	0.25	0.5	25	20	15	0.6	17	20	623	0.11	0.75	0.25	20	20	15	0.6	15	20
193	6.57	0.25	0.75	20	15	10	0.2	10	15	624	6.41	0.75	0.25	20	20	15	0.6	17	20
194	5.00	0.25	0.75	20	15	10	0.2	12	17	625	1.65	0.75	0.25	25	15	10	0.2	10	15
195	8.04	0.25	0.75	20	15	10	0.2	15	20	626	3.32	0.75	0.25	25	15	10	0.2	12	17
196	0.08	0.25	0.75	20	15	10	0.2	17	20	627	0.16	0.75	0.25	25	15	10	0.2	15	20
197	4.90	0.25	0.75	20	15	10	0.4	10	15	628	0.35	0.75	0.25	25	15	10	0.2	17	20
198	0.19	0.25	0.75	20	15	10	0.4	12	17	629	0.20	0.75	0.25	25	15	10	0.4	10	15
199	0.09	0.25	0.75	20	15	10	0.4	15	20	630	0.15	0.75	0.25	25	15	10	0.4	12	17
200	3.29	0.25	0.75	20	15	10	0.4	17	20	631	8.16	0.75	0.25	25	15	10	0.4	15	20
201	3.26	0.25	0.75	20	15	10	0.6	10	15	632	3.36	0.75	0.25	25	15	10	0.4	17	20
202	1.81	0.25	0.75	20	15	10	0.6	12	17	633	0.12	0.75	0.25	25	15	10	0.6	10	15
203	6.46	0.25	0.75	20	15	10	0.6	15	20	634	1.78	0.75	0.25	25	15	10	0.6	12	17
204	8.02	0.25	0.75	20	15	10	0.6	17	20	635	4.88	0.75	0.25	25	15	10	0.6	15	20
205	1.66	0.25	0.75	20	15	15	0.2	10	15	636	7.98	0.75	0.25	25	15	10	0.6	17	20
206	6.48	0.25	0.75	20	15	15	0.2	12	17	637	6.48	0.75	0.25	25	15	15	0.2	10	15
207	1.72	0.25	0.75	20	15	15	0.2	15	20	638	0.13	0.75	0.25	25	15	15	0.2	12	17
208	3.29	0.25	0.75	20	15	15	0.2	17	20	639	1.71	0.75	0.25	25	15	15	0.2	15	20
209	8.01	0.25	0.75	20	15	15	0.4	10	15	640	1.72	0.75	0.25	25	15	15	0.2	17	20
210	6.47	0.25	0.75	20	15	15	0.4	12	17	641	0.13	0.75	0.25	25	15	15	0.4	10	15
211	1.78	0.25	0.75	20	15	15	0.4	15	20	642	0.16	0.75	0.25	25	15	15	0.4	12	17
212	8.04	0.25	0.75	20	15	15	0.4	17	20	643	0.08	0.75	0.25	25	15	15	0.4	15	20
213	8.02	0.25	0.75	20	15	15	0.6	10	15	644	1.73	0.75	0.25	25	15	15	0.4	17	20
214	0.24	0.25	0.75	20	15	15	0.6	12	17	645	0.21	0.75	0.25	25	15	15	0.6	10	15
215	6.49	0.25	0.75	20	15	15	0.6	15	20	646	0.26	0.75	0.25	25	15	15	0.6	12	17
216	1.76	0.25	0.75	20	15	15	0.6	17	20	647	0.12	0.75	0.25	25	15	15	0.6	15	20
217	3.29	0.25	0.75	20	20	10	0.2	10	15	648	8.05	0.75	0.25	25	15	15	0.6	17	20
218	6.42	0.25	0.75	20	20	10	0.2	12	17	649	0.21	0.75	0.25	25	20	10	0.2	10	15
219	3.31	0.25	0.75	20	20	10	0.2	15	20	650	8.03	0.75	0.25	25	20	10	0.2	12	17
220	0.32	0.25	0.75	20	20	10	0.2	17	20	651	9.69	0.75	0.25	25	20	10	0.2	15	20
221	6.45	0.25	0.75	20	20	10	0.4	10	15	652	4.82	0.75	0.25	25	20	10	0.2	17	20
222	8.03	0.25	0.75	20	20	10	0.4	12	17	653	4.88	0.75	0.25	25	20	10	0.4	10	15
223	1.69	0.25	0.75	20	20	10	0.4	15	20	654	0.12	0.75	0.25	25	20	10	0.4	12	17
224	8.00	0.25	0.75	20	20	10	0.4	17	20	655	3.29	0.75	0.25	25	20	10	0.4	15	20
225	1.71	0.25	0.75	20	20	10	0.6	10	15	656	6.42	0.75	0.25	25	20	10	0.4	17	20
226	0.20	0.25	0.75	20	20	10	0.6	12	17	657	6.47	0.75	0.25	25	20	10	0.6	10	15
227	0.13	0.25	0.75	20	20	10	0.6	15	20	658	0.33	0.75	0.25	25	20	10	0.6	12	17
228	4.85	0.25	0.75	20	20	10	0.6	17	20	659	15.9	0.75	0.25	25	20	10	0.6	15	20
229	8.00	0.25	0.75	20	20	15	0.2	10	15	660	8.01	0.75	0.25	25	20	10	0.6	17	20
230	3.34	0.25	0.75	20	20	15	0.2	12	17	661	3.54	0.75	0.25	25	20	15	0.2	10	15
231	6.49	0.25	0.75	20	20	15	0.2	15	20	662	1.65	0.75	0.25	25	20	15	0.2	12	17
232	3.32	0.25	0.75	20	20	15	0.2	17	20	663	7.95	0.75	0.25	25	20	15	0.2	15	20
233	3.49	0.25	0.75	20	20	15	0.4	10	15	664	0.18	0.75	0.25	25	20	15	0.2	17	20
234	8.04	0.25	0.75	20	20	15	0.4	12	17	665	6.52	0.75	0.25	25	20	15	0.4	10	15
235	0.23	0.25	0.75	20	20	15	0.4	15	20	666	8.09	0.75	0.25	25	20	15	0.4	12	17
236	1.72	0.25	0.75	20	20	15	0.4	17	20	667	0.07	0.75	0.25	25	20	15	0.4	15	20
237	1.82	0.25	0.75	20	20	15	0.6	10	15	668	0.09	0.75	0.25	25	20	15	0.4	17	20
238	1.82	0.25	0.75	20	20	15	0.6	12	17	669	1.80	0.75	0.25	25	20	15	0.6	10	15

N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV
239	1.75	0.25	0.75	20	20	15	0.6	15	20
240	0.09	0.25	0.75	20	20	15	0.6	17	20
241	3.37	0.25	0.75	25	15	10	0.2	10	15
242	8.03	0.25	0.75	25	15	10	0.2	12	17
243	1.71	0.25	0.75	25	15	10	0.2	15	20
244	0.17	0.25	0.75	25	15	10	0.2	17	20
245	1.85	0.25	0.75	25	15	10	0.4	10	15
246	0.14	0.25	0.75	25	15	10	0.4	12	17
247	0.21	0.25	0.75	25	15	10	0.4	15	20
248	6.64	0.25	0.75	25	15	10	0.4	17	20
249	9.64	0.25	0.75	25	15	10	0.6	10	15
250	0.18	0.25	0.75	25	15	10	0.6	12	17
251	3.35	0.25	0.75	25	15	10	0.6	15	20
252	3.37	0.25	0.75	25	15	10	0.6	17	20
253	8.11	0.25	0.75	25	15	15	0.2	10	15
254	0.21	0.25	0.75	25	15	15	0.2	12	17
255	1.79	0.25	0.75	25	15	15	0.2	15	20
256	0.15	0.25	0.75	25	15	15	0.2	17	20
257	0.13	0.25	0.75	25	15	15	0.4	10	15
258	3.36	0.25	0.75	25	15	15	0.4	12	17
259	0.23	0.25	0.75	25	15	15	0.4	15	20
260	6.40	0.25	0.75	25	15	15	0.4	17	20
261	4.88	0.25	0.75	25	15	15	0.6	10	15
262	0.12	0.25	0.75	25	15	15	0.6	12	17
263	1.76	0.25	0.75	25	15	15	0.6	15	20
264	4.84	0.25	0.75	25	15	15	0.6	17	20
265	1.77	0.25	0.75	25	20	10	0.2	10	15
266	6.45	0.25	0.75	25	20	10	0.2	12	17
267	1.84	0.25	0.75	25	20	10	0.2	15	20
268	1.75	0.25	0.75	25	20	10	0.2	17	20
269	8.02	0.25	0.75	25	20	10	0.4	10	15
270	6.47	0.25	0.75	25	20	10	0.4	12	17
271	4.84	0.25	0.75	25	20	10	0.4	15	20
272	6.40	0.25	0.75	25	20	10	0.4	17	20
273	9.66	0.25	0.75	25	20	10	0.6	10	15
274	7.99	0.25	0.75	25	20	10	0.6	12	17
275	4.85	0.25	0.75	25	20	10	0.6	15	20
276	4.84	0.25	0.75	25	20	10	0.6	17	20
277	6.47	0.25	0.75	25	20	15	0.2	10	15
278	3.29	0.25	0.75	25	20	15	0.2	12	17
279	0.19	0.25	0.75	25	20	15	0.2	15	20
280	3.33	0.25	0.75	25	20	15	0.2	17	20
281	6.55	0.25	0.75	25	20	15	0.4	10	15
282	0.26	0.25	0.75	25	20	15	0.4	12	17
283	9.54	0.25	0.75	25	20	15	0.4	15	20
284	4.91	0.25	0.75	25	20	15	0.4	17	20
285	0.21	0.25	0.75	25	20	15	0.6	10	15
286	6.43	0.25	0.75	25	20	15	0.6	12	17
287	15.9	0.25	0.75	25	20	15	0.6	15	20
288	6.47	0.25	0.75	25	20	15	0.6	17	20
289	8.06	0.5	0.25	20	15	10	0.2	10	15
290	4.93	0.5	0.25	20	15	10	0.2	12	17
291	1.73	0.5	0.25	20	15	10	0.2	15	20
292	8.00	0.5	0.25	20	15	10	0.2	17	20
293	5.00	0.5	0.25	20	15	10	0.4	10	15
294	0.18	0.5	0.25	20	15	10	0.4	12	17
295	6.40	0.5	0.25	20	15	10	0.4	15	20
296	1.68	0.5	0.25	20	15	10	0.4	17	20
297	0.18	0.5	0.25	20	15	10	0.6	10	15
298	6.32	0.5	0.25	20	15	10	0.6	12	17
299	6.43	0.5	0.25	20	15	10	0.6	15	20
300	0.29	0.5	0.25	20	15	10	0.6	17	20
301	7.97	0.5	0.25	20	15	15	0.2	10	15
302	0.10	0.5	0.25	20	15	15	0.2	12	17
303	4.88	0.5	0.25	20	15	15	0.2	15	20
304	4.77	0.5	0.25	20	15	15	0.2	17	20
305	3.34	0.5	0.25	20	15	15	0.4	10	15
306	4.95	0.5	0.25	20	15	15	0.4	12	17
307	0.05	0.5	0.25	20	15	15	0.4	15	20
308	1.64	0.5	0.25	20	15	15	0.4	17	20
309	9.60	0.5	0.25	20	15	15	0.6	10	15
310	8.00	0.5	0.25	20	15	15	0.6	12	17
311	1.87	0.5	0.25	20	15	15	0.6	15	20
312	3.32	0.5	0.25	20	15	15	0.6	17	20
313	1.81	0.5	0.25	20	20	10	0.2	10	15
314	4.89	0.5	0.25	20	20	10	0.2	12	17
315	6.55	0.5	0.25	20	20	10	0.2	15	20
316	12.8	0.5	0.25	20	20	10	0.2	17	20
317	6.60	0.5	0.25	20	20	10	0.4	10	15
318	1.76	0.5	0.25	20	20	10	0.4	12	17
319	0.22	0.5	0.25	20	20	10	0.4	15	20
670	0.09	0.75	0.25	25	20	15	0.6	12	17
671	0.19	0.75	0.25	25	20	15	0.6	15	20
672	0.14	0.75	0.25	25	20	15	0.6	17	20
673	1.67	0.75	0.5	20	15	10	0.2	10	15
674	1.78	0.75	0.5	20	15	10	0.2	12	17
675	0.24	0.75	0.5	20	15	10	0.2	15	20
676	0.11	0.75	0.5	20	15	10	0.2	17	20
677	6.46	0.75	0.5	20	15	10	0.4	10	15
678	7.98	0.75	0.5	20	15	10	0.4	12	17
679	8.08	0.75	0.5	20	15	10	0.4	15	20
680	0.09	0.75	0.5	20	15	10	0.4	17	20
681	6.45	0.75	0.5	20	15	10	0.6	10	15
682	1.86	0.75	0.5	20	15	10	0.6	12	17
683	1.71	0.75	0.5	20	15	10	0.6	15	20
684	0.18	0.75	0.5	20	15	10	0.6	17	20
685	8.10	0.75	0.5	20	15	15	0.2	10	15
686	1.78	0.75	0.5	20	15	15	0.2	12	17
687	11.2	0.75	0.5	20	15	15	0.2	15	20
688	0.09	0.75	0.5	20	15	15	0.2	17	20
689	4.87	0.75	0.5	20	15	15	0.4	10	15
690	1.73	0.75	0.5	20	15	15	0.4	12	17
691	0.16	0.75	0.5	20	15	15	0.4	15	20
692	6.46	0.75	0.5	20	15	15	0.4	17	20
693	1.82	0.75	0.5	20	15	15	0.6	10	15
694	0.17	0.75	0.5	20	15	15	0.6	12	17
695	11.2	0.75	0.5	20	15	15	0.6	15	20
696	0.07	0.75	0.5	20	15	15	0.6	17	20
697	6.47	0.75	0.5	20	20	10	0.2	10	15
698	1.72	0.75	0.5	20	20	10	0.2	12	17
699	3.21	0.75	0.5	20	20	10	0.2	15	20
700	8.36	0.75	0.5	20	20	10	0.2	17	20
701	3.27	0.75	0.5	20	20	10	0.4	10	15
702	6.53	0.75	0.5	20	20	10	0.4	12	17
703	0.25	0.75	0.5	20	20	10	0.4	15	20
704	0.14	0.75	0.5	20	20	10	0.4	17	20
705	6.46	0.75	0.5	20	20	10	0.6	10	15
706	0.16	0.75	0.5	20	20	10	0.6	12	17
707	1.76	0.75	0.5	20	20	10	0.6	15	20
708	1.74	0.75	0.5	20	20	10	0.6	17	20
709	7.97	0.75	0.5	20	20	15	0.2	10	15
710	1.78	0.75	0.5	20	20	15	0.2	12	17
711	1.76	0.75	0.5	20	20	15	0.2	15	20
712	1.67	0.75	0.5	20	20	15	0.2	17	20
713	0.22	0.75	0.5	20	20	15	0.4	10	15
714	6.54	0.75	0.5	20	20	15	0.4	12	17
715	3.33	0.75	0.5	20	20	15	0.4	15	20
716	0.22	0.75	0.5	20	20	15	0.4	17	20
717	1.76	0.75	0.5	20	20	15	0.6	10	15
718	4.92	0.75	0.5	20	20	15	0.6	12	17
719	3.31	0.75	0.5	20	20	15	0.6	15	20
720	8.10	0.75	0.5	20	20	15	0.6	17	20
721	4.88	0.75	0.5	25	15	10	0.2	10	15
722	0.12	0.75	0.5	25	15	10	0.2	12	17
723	4.91	0.75	0.5	25	15	10	0.2	15	20
724	6.45	0.75	0.5	25	15	10	0.2	17	20
725	1.85	0.75	0.5	25	15	10	0.4	10	15
726	1.85	0.75	0.5	25	15	10	0.4	12	17
727	1.71	0.75	0.5	25	15	10	0.4	15	20
728	0.27	0.75	0.5	25	15	10	0.4	17	20
729	6.42	0.75	0.5	25	15	10	0.6	10	15
730	9.59	0.75	0.5	25	15	10	0.6	12	17
731	8.00	0.75	0.5	25	15	10	0.6	15	20
732	3.30	0.75	0.5	25	15	10	0.6	17	20
733	0.14	0.75	0.5	25	15	15	0.2	10	15
734	1.73	0.75	0.5	25	15	15	0.2	12	17
735	1.76	0.75	0.5	25	15	15	0.2	15	20
736	3.30	0.75	0.5	25	15	15	0.2	17	20
737	0.08	0.75	0.5	25	15	15	0.4	10	15
738	0.36	0.75	0.5	25	15	15	0.4	12	17
739	3.31	0.75	0.5	25	15	15	0.4	15	20
740	1.71	0.75	0.5	25	15	15	0.4	17	20
741	1.77	0.75	0.5	25	15	15	0.6	10	15
742	6.43	0.75	0.5	25	15	15	0.6	12	17
743	6.42	0.75	0.5	25	15	15	0.6	15	20
744	3.24	0.75	0.5	25	15	15	0.6	17	20
745	6.46	0.75	0.5	25	20	10	0.2	10	15
746	4.88	0.75	0.5	25	20	10	0.2	12	17
747	3.33	0.75	0.5	25	20	10	0.2	15	20
748	1.71	0.75	0.5	25	20	10	0.2	17	20
749	4.91	0.75	0.5	25	20	10	0.4	10	15
750	3.20	0.75	0.5	25	20	10	0.4	12	17

N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV
320	0.10	0.5	0.25	20	20	10	0.4	17	20
321	0.38	0.5	0.25	20	20	10	0.6	10	15
322	3.33	0.5	0.25	20	20	10	0.6	12	17
323	1.79	0.5	0.25	20	20	10	0.6	15	20
324	0.19	0.5	0.25	20	20	10	0.6	17	20
325	0.19	0.5	0.25	20	20	15	0.2	10	15
326	6.45	0.5	0.25	20	20	15	0.2	12	17
327	4.85	0.5	0.25	20	20	15	0.2	15	20
328	1.76	0.5	0.25	20	20	15	0.2	17	20
329	4.88	0.5	0.25	20	20	15	0.4	10	15
330	9.57	0.5	0.25	20	20	15	0.4	12	17
331	6.50	0.5	0.25	20	20	15	0.4	15	20
332	0.19	0.5	0.25	20	20	15	0.4	17	20
333	1.78	0.5	0.25	20	20	15	0.6	10	15
334	8.07	0.5	0.25	20	20	15	0.6	12	17
335	9.65	0.5	0.25	20	20	15	0.6	15	20
336	0.09	0.5	0.25	20	20	15	0.6	17	20
337	8.05	0.5	0.25	25	15	10	0.2	10	15
338	1.73	0.5	0.25	25	15	10	0.2	12	17
339	6.46	0.5	0.25	25	15	10	0.2	15	20
340	6.52	0.5	0.25	25	15	10	0.2	17	20
341	4.84	0.5	0.25	25	15	10	0.4	10	15
342	0.17	0.5	0.25	25	15	10	0.4	12	17
343	1.79	0.5	0.25	25	15	10	0.4	15	20
344	1.66	0.5	0.25	25	15	10	0.4	17	20
345	1.72	0.5	0.25	25	15	10	0.6	10	15
346	4.82	0.5	0.25	25	15	10	0.6	12	17
347	4.85	0.5	0.25	25	15	10	0.6	15	20
348	7.96	0.5	0.25	25	15	10	0.6	17	20
349	1.71	0.5	0.25	25	15	15	0.2	10	15
350	0.25	0.5	0.25	25	15	15	0.2	12	17
351	1.81	0.5	0.25	25	15	15	0.2	15	20
352	3.31	0.5	0.25	25	15	15	0.2	17	20
353	3.45	0.5	0.25	25	15	15	0.4	10	15
354	0.13	0.5	0.25	25	15	15	0.4	12	17
355	1.75	0.5	0.25	25	15	15	0.4	15	20
356	1.65	0.5	0.25	25	15	15	0.4	17	20
357	0.08	0.5	0.25	25	15	15	0.6	10	15
358	7.98	0.5	0.25	25	15	15	0.6	12	17
359	0.12	0.5	0.25	25	15	15	0.6	15	20
360	6.47	0.5	0.25	25	15	15	0.6	17	20
361	0.12	0.5	0.25	25	20	10	0.2	10	15
362	1.82	0.5	0.25	25	20	10	0.2	12	17
363	0.10	0.5	0.25	25	20	10	0.2	15	20
364	0.10	0.5	0.25	25	20	10	0.2	17	20
365	3.33	0.5	0.25	25	20	10	0.4	10	15
366	7.95	0.5	0.25	25	20	10	0.4	12	17
367	1.72	0.5	0.25	25	20	10	0.4	15	20
368	9.61	0.5	0.25	25	20	10	0.4	17	20
369	3.24	0.5	0.25	25	20	10	0.6	10	15
370	3.33	0.5	0.25	25	20	10	0.6	12	17
371	8.01	0.5	0.25	25	20	10	0.6	15	20
372	8.14	0.5	0.25	25	20	10	0.6	17	20
373	3.49	0.5	0.25	25	20	15	0.2	10	15
374	0.23	0.5	0.25	25	20	15	0.2	12	17
375	0.26	0.5	0.25	25	20	15	0.2	15	20
376	3.43	0.5	0.25	25	20	15	0.2	17	20
377	0.14	0.5	0.25	25	20	15	0.4	10	15
378	0.30	0.5	0.25	25	20	15	0.4	12	17
379	1.76	0.5	0.25	25	20	15	0.4	15	20
380	1.72	0.5	0.25	25	20	15	0.4	17	20
381	0.14	0.5	0.25	25	20	15	0.6	10	15
382	6.56	0.5	0.25	25	20	15	0.6	12	17
383	0.24	0.5	0.25	25	20	15	0.6	15	20
384	0.18	0.5	0.25	25	20	15	0.6	17	20
385	3.45	0.5	0.5	20	15	10	0.2	10	15
386	3.27	0.5	0.5	20	15	10	0.2	12	17
387	8.06	0.5	0.5	20	15	10	0.2	15	20
388	0.08	0.5	0.5	20	15	10	0.2	17	20
389	4.92	0.5	0.5	20	15	10	0.4	10	15
390	4.93	0.5	0.5	20	15	10	0.4	12	17
391	0.18	0.5	0.5	20	15	10	0.4	15	20
392	1.85	0.5	0.5	20	15	10	0.4	17	20
393	0.24	0.5	0.5	20	15	10	0.6	10	15
394	1.68	0.5	0.5	20	15	10	0.6	12	17
395	0.13	0.5	0.5	20	15	10	0.6	15	20
396	1.67	0.5	0.5	20	15	10	0.6	17	20
397	1.83	0.5	0.5	20	15	15	0.2	10	15
398	1.78	0.5	0.5	20	15	15	0.2	12	17
399	0.22	0.5	0.5	20	15	15	0.2	15	20
400	7.97	0.5	0.5	20	15	15	0.2	17	20
751	0.16	0.75	0.5	25	20	10	0.4	15	20
752	1.72	0.75	0.5	25	20	10	0.4	17	20
753	1.99	0.75	0.5	25	20	10	0.6	10	15
754	6.42	0.75	0.5	25	20	10	0.6	12	17
755	3.28	0.75	0.5	25	20	10	0.6	15	20
756	3.30	0.75	0.5	25	20	10	0.6	17	20
757	6.45	0.75	0.5	25	20	15	0.2	10	15
758	1.72	0.75	0.5	25	20	15	0.2	12	17
759	1.72	0.75	0.5	25	20	15	0.2	15	20
760	12.8	0.75	0.5	25	20	15	0.2	17	20
761	3.39	0.75	0.5	25	20	15	0.4	10	15
762	0.29	0.75	0.5	25	20	15	0.4	12	17
763	7.99	0.75	0.5	25	20	15	0.4	15	20
764	6.43	0.75	0.5	25	20	15	0.4	17	20
765	9.74	0.75	0.5	25	20	15	0.6	10	15
766	4.89	0.75	0.5	25	20	15	0.6	12	17
767	1.74	0.75	0.5	25	20	15	0.6	15	20
768	3.26	0.75	0.5	25	20	15	0.6	17	20
769	4.86	0.75	0.75	20	15	10	0.2	10	15
770	0.24	0.75	0.75	20	15	10	0.2	12	17
771	1.76	0.75	0.75	20	15	10	0.2	15	20
772	4.86	0.75	0.75	20	15	10	0.2	17	20
773	12.8	0.75	0.75	20	15	10	0.4	10	15
774	0.23	0.75	0.75	20	15	10	0.4	12	17
775	1.75	0.75	0.75	20	15	10	0.4	15	20
776	0.11	0.75	0.75	20	15	10	0.4	17	20
777	1.73	0.75	0.75	20	15	10	0.6	10	15
778	0.15	0.75	0.75	20	15	10	0.6	12	17
779	3.36	0.75	0.75	20	15	10	0.6	15	20
780	0.19	0.75	0.75	20	15	10	0.6	17	20
781	0.24	0.75	0.75	20	15	15	0.2	10	15
782	0.25	0.75	0.75	20	15	15	0.2	12	17
783	0.33	0.75	0.75	20	15	15	0.2	15	20
784	0.11	0.75	0.75	20	15	15	0.2	17	20
785	8.09	0.75	0.75	20	15	15	0.4	10	15
786	7.97	0.75	0.75	20	15	15	0.4	12	17
787	6.34	0.75	0.75	20	15	15	0.4	15	20
788	1.73	0.75	0.75	20	15	15	0.4	17	20
789	4.81	0.75	0.75	20	15	15	0.6	10	15
790	8.00	0.75	0.75	20	15	15	0.6	12	17
791	0.14	0.75	0.75	20	15	15	0.6	15	20
792	3.30	0.75	0.75	20	15	15	0.6	17	20
793	7.97	0.75	0.75	20	20	10	0.2	10	15
794	0.20	0.75	0.75	20	20	10	0.2	12	17
795	3.24	0.75	0.75	20	20	10	0.2	15	20
796	8.04	0.75	0.75	20	20	10	0.2	17	20
797	6.57	0.75	0.75	20	20	10	0.4	10	15
798	0.18	0.75	0.75	20	20	10	0.4	12	17
799	1.80	0.75	0.75	20	20	10	0.4	15	20
800	0.22	0.75	0.75	20	20	10	0.4	17	20
801	8.11	0.75	0.75	20	20	10	0.6	10	15
802	6.51	0.75	0.75	20	20	10	0.6	12	17
803	1.71	0.75	0.75	20	20	10	0.6	15	20
804	0.23	0.75	0.75	20	20	10	0.6	17	20
805	12.8	0.75	0.75	20	20	15	0.2	10	15
806	7.98	0.75	0.75	20	20	15	0.2	12	17
807	0.23	0.75	0.75	20	20	15	0.2	15	20
808	4.89	0.75	0.75	20	20	15	0.2	17	20
809	0.27	0.75	0.75	20	20	15	0.4	10	15
810	0.19	0.75	0.75	20	20	15	0.4	12	17
811	6.39	0.75	0.75	20	20	15	0.4	15	20
812	4.97	0.75	0.75	20	20	15	0.4	17	20
813	9.63	0.75	0.75	20	20	15	0.6	10	15
814	6.38	0.75	0.75	20	20	15	0.6	12	17
815	1.79	0.75	0.75	20	20	15	0.6	15	20
816	1.73	0.75	0.75	20	20	15	0.6	17	20
817	1.78	0.75	0.75	25	15	10	0.2	10	15
818	3.28	0.75	0.75	25	15	10	0.2	12	17
819	1.65	0.75	0.75	25	15	10	0.2	15	20
820	3.34	0.75	0.75	25	15	10	0.2	17	20
821	6.54	0.75	0.75	25	15	10	0.4	10	15
822	4.86	0.75	0.75	25	15	10	0.4	12	17
823	0.22	0.75	0.75	25	15	10	0.4	15	20
824	0.10	0.75	0.75	25	15	10	0.4	17	20
825	1.77	0.75	0.75	25	15	10	0.6	10	15
826	4.83	0.75	0.75	25	15	10	0.6	12	17
827	0.15	0.75	0.75	25	15	10	0.6	15	20
828	3.25	0.75	0.75	25	15	10	0.6	17	20
829	1.78	0.75	0.75	25	15	15	0.2	10	15
830	8.01	0.75	0.75	25	15	15	0.2	12	17
831	6.39	0.75	0.75	25	15	15	0.2	15	20

N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV	N	Gap	α	β	$\sigma1$	$\sigma2$	$\sigma3$	r	LV	MV
401	3.33	0.5	0.5	20	15	15	0.4	10	15	832	1.74	0.75	0.75	25	15	15	0.2	17	20
402	1.71	0.5	0.5	20	15	15	0.4	12	17	833	3.45	0.75	0.75	25	15	15	0.4	10	15
403	1.71	0.5	0.5	20	15	15	0.4	15	20	834	6.46	0.75	0.75	25	15	15	0.4	12	17
404	1.70	0.5	0.5	20	15	15	0.4	17	20	835	0.09	0.75	0.75	25	15	15	0.4	15	20
405	1.72	0.5	0.5	20	15	15	0.6	10	15	836	3.36	0.75	0.75	25	15	15	0.4	17	20
406	8.05	0.5	0.5	20	15	15	0.6	12	17	837	11.1	0.75	0.75	25	15	15	0.6	10	15
407	4.89	0.5	0.5	20	15	15	0.6	15	20	838	6.49	0.75	0.75	25	15	15	0.6	12	17
408	0.15	0.5	0.5	20	15	15	0.6	17	20	839	0.16	0.75	0.75	25	15	15	0.6	15	20
409	8.10	0.5	0.5	20	20	10	0.2	10	15	840	0.16	0.75	0.75	25	15	15	0.6	17	20
410	8.02	0.5	0.5	20	20	10	0.2	12	17	841	6.49	0.75	0.75	25	20	10	0.2	10	15
411	6.59	0.5	0.5	20	20	10	0.2	15	20	842	1.79	0.75	0.75	25	20	10	0.2	12	17
412	1.73	0.5	0.5	20	20	10	0.2	17	20	843	0.08	0.75	0.75	25	20	10	0.2	15	20
413	6.51	0.5	0.5	20	20	10	0.4	10	15	844	6.49	0.75	0.75	25	20	10	0.2	17	20
414	0.14	0.5	0.5	20	20	10	0.4	12	17	845	0.24	0.75	0.75	25	20	10	0.4	10	15
415	3.28	0.5	0.5	20	20	10	0.4	15	20	846	6.41	0.75	0.75	25	20	10	0.4	12	17
416	0.13	0.5	0.5	20	20	10	0.4	17	20	847	7.94	0.75	0.75	25	20	10	0.4	15	20
417	6.50	0.5	0.5	20	20	10	0.6	10	15	848	1.65	0.75	0.75	25	20	10	0.4	17	20
418	3.29	0.5	0.5	20	20	10	0.6	12	17	849	0.06	0.75	0.75	25	20	10	0.6	10	15
419	1.68	0.5	0.5	20	20	10	0.6	15	20	850	8.02	0.75	0.75	25	20	10	0.6	12	17
420	0.15	0.5	0.5	20	20	10	0.6	17	20	851	6.48	0.75	0.75	25	20	10	0.6	15	20
421	3.29	0.5	0.5	20	20	15	0.2	10	15	852	1.83	0.75	0.75	25	20	10	0.6	17	20
422	0.19	0.5	0.5	20	20	15	0.2	12	17	853	8.11	0.75	0.75	25	20	15	0.2	10	15
423	6.38	0.5	0.5	20	20	15	0.2	15	20	854	0.14	0.75	0.75	25	20	15	0.2	12	17
424	3.27	0.5	0.5	20	20	15	0.2	17	20	855	4.87	0.75	0.75	25	20	15	0.2	15	20
425	4.84	0.5	0.5	20	20	15	0.4	10	15	856	0.22	0.75	0.75	25	20	15	0.2	17	20
426	4.93	0.5	0.5	20	20	15	0.4	12	17	857	6.43	0.75	0.75	25	20	15	0.4	10	15
427	4.91	0.5	0.5	20	20	15	0.4	15	20	858	8.04	0.75	0.75	25	20	15	0.4	12	17
428	0.15	0.5	0.5	20	20	15	0.4	17	20	859	7.96	0.75	0.75	25	20	15	0.4	15	20
429	1.85	0.5	0.5	20	20	15	0.6	10	15	860	1.79	0.75	0.75	25	20	15	0.4	17	20
430	3.37	0.5	0.5	20	20	15	0.6	12	17	861	4.99	0.75	0.75	25	20	15	0.6	10	15
431	9.61	0.5	0.5	20	20	15	0.6	15	20	862	11.3	0.75	0.75	25	20	15	0.6	12	17
432	0.23	0.5	0.5	20	20	15	0.6	17	20	863	6.38	0.75	0.75	25	20	15	0.6	15	20

Table 7 Experiments results for param tuning ALNS heuristic

Appendix B

Experiments resultы for 14-26 instances with different number of iterations.

Iterations	100			200			300		
Instances	Objective	Time (sec)	Gap (%)	Objective	Time (sec)	Gap (%)	Objective	Time (sec)	Gap (%)
14-4-51	5120045,70	20,10	0,26	5114275,66	48,14	0,15	5120888,20	68,49	0,28
15-4-54	5183711,90	27,14	0,55	5188312,92	52,38	0,64	5171981,32	71,42	0,32
16-4-58	5808079,19	28,83	11,29	5374565,78	62,02	2,98	5242023,75	87,49	0,44
17-5-61	6390910,44	34,04	21,84	6524785,79	67,44	24,39	5693463,52	97,61	8,54
18-5-64	6706411,05	34,14	26,40	6703979,57	69,41	26,36	6701520,58	104,11	26,31
19-6-66	6758649,82	41,19	0,52	6749164,66	88,70	0,37	6744809,29	127,41	0,31
20-6-72	6856607,12	53,22	0,38	6850643,64	95,74	0,30	6844471,32	151,03	0,21
21-6-77	6933699,66	61,12	0,55	6924068,60	127,87	0,41	6925405,46	191,06	0,43
22-6-81	7145779,42	72,85	2,75	6998202,81	142,89	0,63	7272093,82	324,74	4,56
23-7-84	8021064,67	69,41	14,64	7601507,23	134,77	8,64	7592628,92	200,37	8,52
24-7-87	8610735,01	75,18	0,59	8603847,58	147,45	0,51	8593260,29	335,11	0,38
25-8-88	8616684,28	89,76	0,41	8608975,97	195,90	0,32	8603167,21	269,92	0,26
26-8-91	8681364,59	113,45	0,59	8674147,44	215,60	0,51	8672120,89	344,01	0,48
Average		55,42	6,21		111,41	5,09		182,52	3,93

Table 8 Results for 14-26 instances with 100 – 300 iterations

Iterations	400			500			600		
Instances	Objective	Time (sec)	Gap (%)	Objective	Time (sec)	Gap (%)	Objective	Time (sec)	Gap (%)
14-4-51	5111061,19	85,71	0,08	5111986,05	114,37	0,10	5108317,91	138,86	0,03
15-4-54	5173647,48	97,33	0,35	5170304,50	121,42	0,29	5165258,30	158,47	0,19
16-4-58	5368316,28	116,00	2,86	5228608,48	152,56	0,19	5226722,86	166,67	0,15
17-5-61	5425634,29	125,44	3,44	5826886,88	159,85	11,09	5542871,29	176,53	5,67
18-5-64	6284991,83	139,60	18,46	6561463,30	170,12	23,67	6284024,53	217,77	18,44
19-6-66	6743266,85	173,51	0,29	6746366,06	213,80	0,33	6746049,23	262,93	0,33
20-6-72	6840364,42	211,97	0,15	6841720,04	233,82	0,17	6841079,89	308,52	0,16
21-6-77	6921331,50	254,31	0,37	6919380,53	319,02	0,34	6920275,52	378,46	0,36
22-6-81	7122582,72	266,83	2,41	6987914,55	335,45	0,48	6977291,86	414,08	0,32
23-7-84	7592994,62	258,61	8,52	7451395,00	312,04	6,50	7452245,15	388,72	6,51
24-7-87	8589650,41	275,35	0,34	8591389,44	346,17	0,36	8590316,95	427,31	0,35
25-8-88	8604433,28	375,63	0,27	8598830,52	444,88	0,21	8599322,49	541,06	0,21
26-8-91	8668598,88	451,90	0,44	8659159,62	529,56	0,33	8660128,94	642,01	0,34
Average		217,86	2,92		265,62	3,39		324,72	2,54

Table 9 Results for 14 – 26 instances with 400 - 600 iterations

Iterations	700			800			900		
Instances	Objective	Time	Gap (%)	Objective	Time	Gap (%)	Objective	Time	Gap (%)
14-4-51	5110778,66	157,13	0,08	5110174,53	174,68	0,07	5107761,19	210,08	0,02
15-4-54	5166278,29	169,42	0,21	5170159,41	197,29	0,29	5166012,80	214,98	0,21
16-4-58	5224024,51	196,65	0,10	5227304,36	234,04	0,16	5223226,20	262,86	0,08
17-5-61	5278032,73	229,54	0,62	5412024,79	265,59	3,18	5264862,58	290,77	0,37
18-5-64	6420629,24	251,80	21,02	6697754,55	266,98	26,24	6556189,14	387,34	23,57
19-6-66	6740552,52	296,47	0,25	6745441,16	324,75	0,32	6742276,49	377,22	0,27
20-6-72	6840895,64	338,43	0,15	6840504,59	395,97	0,15	6839892,79	426,69	0,14
21-6-77	6916667,10	434,65	0,30	6916586,31	518,65	0,30	6914560,12	583,15	0,27
22-6-81	6978388,58	454,28	0,34	6978568,37	552,16	0,34	6972994,07	578,53	0,26
23-7-84	7451225,10	482,23	6,50	7165405,24	523,13	2,41	7170114,22	689,71	2,48
24-7-87	8586662,51	468,79	0,31	8582785,70	569,79	0,26	8580845,18	597,02	0,24
25-8-88	8600078,04	594,93	0,22	8594321,01	733,81	0,15	8597167,67	821,67	0,19
26-8-91	8661710,23	735,60	0,36	8655750,36	838,40	0,29	8653054,32	976,77	0,26
Average		369,99	2,34		430,40	2,63		493,60	2,18

Table 10 Results for 14-26 instances with 700 – 900 iterations

Iterations	1000		
Instances	Objective	Time	Gap (%)
14-4-51	5107670,54	236,30	0,02
15-4-54	5164885,08	250,38	0,18
16-4-58	5225937,11	286,61	0,14
17-5-61	5406131,95	334,48	3,06
18-5-64	6417286,32	361,27	20,95
19-6-66	6738833,94	420,36	0,22
20-6-72	6837138,95	494,35	0,10
21-6-77	6915522,13	622,09	0,29
22-6-81	6974638,86	672,21	0,29
23-7-84	7025614,49	587,68	0,41
24-7-87	8583470,44	693,69	0,27
25-8-88	8595034,23	875,18	0,16
26-8-91	8656771,75	1048,74	0,31
Average		529,49	2,03

Table 11 Results for 14-26 instances with 1000 iterations