Master's thesis

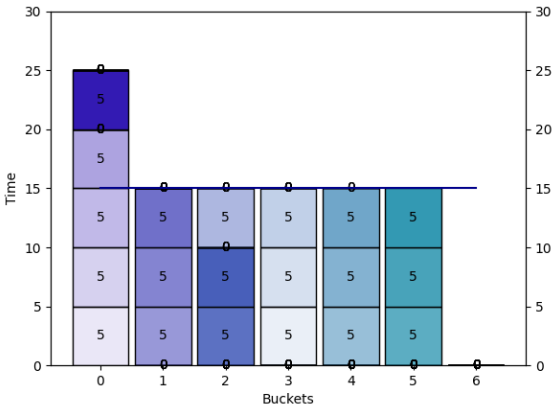
Potential KPIs: a local search technique to escape local optima.

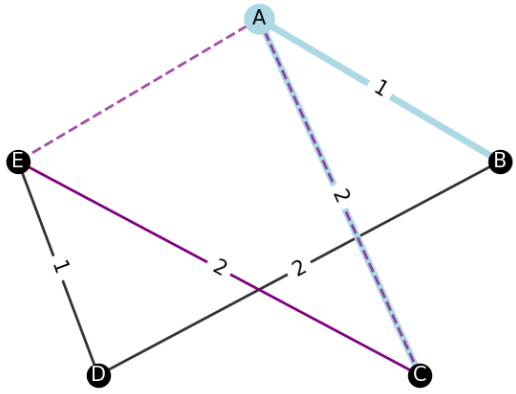
Author

Elise Deen (4896467)

Supervisors

Dr.ir. J.T. van Essen
Dr.ir. D. Huizing





Delft University of Technology
Faculty Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

September 2023 - June 2024

Master's thesis

Potential KPIs: a local search technique to escape local optima.

Elise Deen

September 2023 - June 2024

Abstract

Local search approaches are often used to find solutions for optimisation problems. However, these approaches easily get stuck in local optima, who are not yet globally optimal. More complex variants of those approaches cannot always escape those local optima easily. Thus, the need for another effective technique arises. This thesis proposes such a technique: potential KPIs. A potential KPI is an addition to the objective function such that the local search algorithm accepts solutions with 'potential', i.e., solutions that are closer to the global optimum in some sense.
This thesis focuses on two optimisation problems, the bucketised planning problem (BPP) and the travelling salesman problem, to investigate the performance of potential KPIs. Various potential KPI functions and employment methods are tested to determine the efficacy of potential KPIs and to study their behaviour.
The findings of this thesis indicate that adding a potential KPI to the objective function can significantly improve a basic local search algorithm. The performance of the method with potential KPI is highly dependent on the chosen potential KPI function. Moreover, defining such a function is challenging since some knowledge about the local and global optima is needed. While a random function can already improve the algorithm a lot, using a well-constructed function performs even better as potential KPI.
Furthermore, the performance of the method is dependent on how the chosen function is employed. The method should always end with the local search with the original objective function, but whether to start with or without potential KPI is dependent on the function. Nevertheless, starting without potential KPI seems to be slightly better due to the lower runtime and ability to be always better than the basic local search. The weight of the potential KPI in comparison with the original objective should be high and instance-dependent. Additionally, the initial solution also influences the found solution of the method, as a better initial solution, such as a greedy solution, leaves less room for improvement for the basic local search. Thus, the technique of a potential KPI can significantly improve the basic local search, but its effectiveness is dependent on the method of employment and constructed function. Moreover, a potential KPI can only be used when enough information about the optimisation problem is known.

Contents

Chapter 1

Introduction

Local search methods are frequently used to find solutions for optimisation problems. A local search algorithm explores the neighbourhood solutions of a current solution. The algorithm iteratively moves to those neighbourhood solutions when their objective function, or often called their key performance indicators (KPIs) in the business sector, exhibit a better value. Such a search algorithm can be used to get sufficient solutions to an optimisation problem in a relatively short time. These sufficient solutions are often locally optimal, instead of the desired global optimum. This happens quite frequently for the basic local search algorithm (see Figure 1.1), as this algorithm only searches for better solutions in the neighbourhood, which are not present in the neighbourhood of a local optimum. Figure 1.1 depicts the solution space with objective values. The objective value is represented by a colour gradient, with darker colours representing worse objective values and lighter colours representing better objective values. The yellow dots represent the solutions that a basic local search algorithm would visit. Note that the local search algorithm moves to neighbourhood solutions with better objective values. It can be indeed observed in the figure that the basic local search algorithm is drawn into a local optimum, while there is a solution in the space with a better objective value.

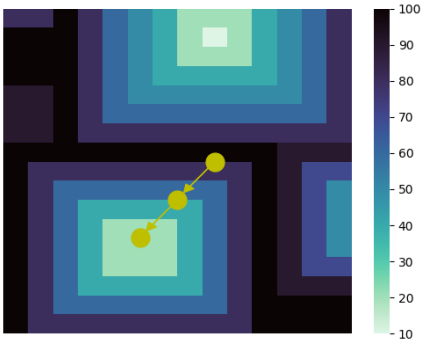
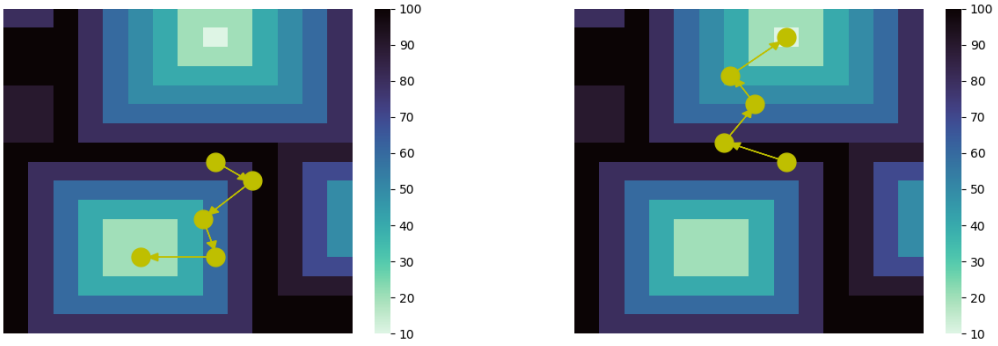


Figure 1.1: This is a visual representation of the solution space of an optimisation problem, with the objective values of the solutions. The yellow arrows and dots show the course of the basic local search algorithm. A lower objective value, indicated by a lighter colour, represents a better objective value.

There exist multiple variants/techniques that are able to escape these local optima [1], such as

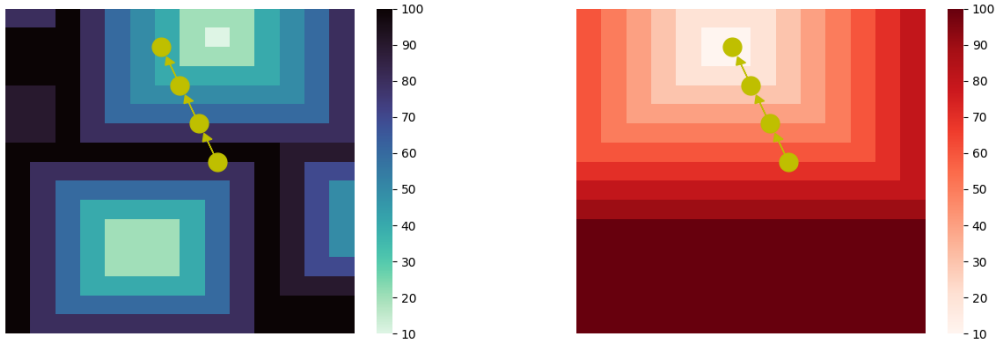
using more complex acceptance criteria (for example Simulated Annealing [2], Tabu Search [3]) or using a smart neighbourhood operator (by parallel scheduling [4] for example). However, these methods always try to find the best solution for the given objective function. While they do accept solutions that are worse than the current solution, they still prefer better solutions. This phenomenon can be observed in Figure 1.2. The solution space with objective values is depicted in the figure, with the yellow dots representing solutions that are visited by variants of the local search algorithm. The algorithm does occasionally accept worse solutions and is therefore able to escape the local optimum, as illustrated in Figure 1.2b. However, the algorithm is still pulled towards local optima. Therefore, dependent on the accepted solutions, the algorithm is still susceptible to becoming trapped in local optima which can be seen in Figure 1.2a.



(a) The algorithm gets stuck in a local optimum. (b) The algorithm escapes the local optimum.

Figure 1.2: This is a visual representation of the solution space of an optimisation problem, with the objective values of the solutions. The yellow arrows and dots show the course of a variant of the local search algorithm that is able to escape local optima. A lower objective value, indicated by a lighter colour, represents a better objective value.

Hence, the existing variants of the local search method have a preference for better solutions which can pull the algorithm into local optima. Moreover, this preference can result in significant computation time required to escape local optima, as it relies on accepting enough worse solutions. An approach that actively steers the algorithm away from these local optima, rather than merely accepting some worse solutions, might be more beneficial. A visual representation of such an approach can be seen in Figure 1.3a. In the figure, the direction, in which new solutions are explored, is different than it was for the basic local search algorithm. A method to change the directing of the algorithm is by using a different objective function as shown in Figure 1.3b. The new direction can help in escaping local optima and could potentially reduce the number of iterations and the runtime needed to escape. However, it is important to search in the right direction, since searching in the wrong direction can cause the algorithm to find only worse solutions than the local optima found with the basic local search algorithm.



(a) The original objective function as heat map. (b) The adapted objective function as heat map.

Figure 1.3: This is a visual representation of the solution space of an optimisation problem, with the objective values of the solutions. The yellow arrows and dots show the course of using a potential KPI in the local search algorithm. A lower objective value, indicated by a lighter colour, represents a better objective value.

A new concept for a local search technique that changes the direction of the search is proposed in this thesis. The concept is to adapt the objective function by adding a synthetic contribution to the objective function. This contribution favours solutions that have 'potential' to become better in next iterations of the algorithm. This contribution changes the objective values of solutions, inducing some solutions to be better (or worse) and directing the search towards different solutions, see Figure 1.3b.
The added component in the objective function is a function which we denote by *the potential KPI*, as the objective function is often called a KPI in the business sector and the addition is designed to favour solutions with 'potential'. The addition of a potential KPI can move the algorithm away from local optima and potentially more towards the global optimum, rather than merely accepting some worse solutions. With a well-defined function added to the objective function, the algorithm can escape local optima more easily and faster compared to most other existing methods, thereby reducing the overall runtime.
The main goal of this thesis is to determine whether the use of potential KPIs improves existing local search algorithms in terms of quality and runtime. The performance of an objective function with potential KPI is investigated for two different problems: the bucketised planning problem and the travelling salesman problem. It is also explored how to employ and find useful potential KPI functions. The research question investigated in this thesis is as follows:

How can potential KPIs be used in local search algorithms to find better solutions and escape local optima using the same local operators?

Other questions that are researched in this thesis and that may help answer the research question, are the following:

1. Does the concept of a potential KPI already exist in literature?
   (a) Which existing local search techniques/algorithms are able to escape local optima?
   (b) Which existing local search techniques/algorithms adapt the objective function in order to find better solutions in terms of the original objective function?

2. How can a potential KPI be defined?
   (a) When does the need arise to use a potential KPI? When are other meta-heuristics not good enough for solving the problem?
   (b) How can one find a potential KPI function and verify that it is a good addition to the objective function?
   (c) How to employ the potential KPI to get a good solution in terms of the original objective? (value of weight, method of employment etc.)
3. How does a local search algorithm with potential KPI perform in comparison with that algorithm without potential KPI?
   (a) What are the (dis)advantages of adding a potential KPI to the local search algorithm?
   (b) What are the limitations of potential KPIs? / When is it not possible to use a potential KPI?
   (c) How good is the solution found with a potential KPI with respect to the optimal solution and the solution found without a potential KPI in terms of the objective value?
   (d) How does the runtime change due to the addition of a potential KPI?

To answer these questions, first a literature review is performed, see Chapter 2. In this review, it is investigated whether the idea of a potential KPI already exists in literature, and moreover, what kind of other techniques and local search algorithms exist. After the literature review, the potential KPI is investigated further to see how such a function can be defined and how it can be employed in a local search algorithm. This is discussed in Chapter 3. Then, two example problems are used to investigate the performance of potential KPIs. The used problems are the bucketised planning problem, Chapter 4, and the travelling salesman problem, Chapter 5. In the discussion, Chapter 6, it is reflected whether the statements and observations found in Chapter 3 also apply in the investigated optimisation problems. Moreover, the two different optimisation problems are compared and it is examined whether some general statements and conclusions can be formed based on the results. This thesis ends with answering the research questions, see Chapter 7, and with some recommendations for future research in potential KPIs, see Chapter 8.

Chapter 2

Literature

The first research question is whether the technique of a potential KPI has already been investigated in literature. It is anticipated that this is not the case. Therefore, different local search methods are reviewed to see which techniques they use to escape local optima and to determine whether these techniques are similar to the concept of a potential KPI. The findings of this literature review are presented in this chapter. In this chapter, first the basic local search algorithm is explained, then different kinds of local search heuristics are stated with their technique to escape local optima. The last part of this chapter consists of some other techniques that adapt the objective function in order to find better solutions or simplify the problem.

2.1 Basic local search algorithm

The most basic form of a local search method is an algorithm that simply searches for a better solution in the neighbourhood of the current solution. A neighbourhood is the set of solutions that can be reached from the current solution by applying a predefined operation. This neighbourhood defining operation is problem-specific. Examples of useful operations are swapping some elements or adjusting variables by a small amount. With local search, the algorithm checks the solutions in the neighbourhood of the current solution and selects the solution that has the best objective value. It continues the search with this new solution if the objective value of this solution is better than the objective value of the current solution. The algorithm keeps searching for better solutions in the neighbourhood until no improvements are made. Note that the objective value indicates the quality or performance of the solution with respect to the optimisation goal.
There are three ways of selecting the best solution in the neighbourhood: the first solution found that has a better objective value than the current solution is selected (called *hill climbing* [5]); a random sample of neighbourhood solutions is checked and the best solution of this sample is selected (algorithm often used in practice); or all neighbours are checked before selecting the best solution. In this thesis, when the basic local search algorithm is mentioned, the latter approach is meant.

2.2 Escaping local optima

Once a basic local search ends in a local optimum, it cannot escape this optimum since only better solutions are considered in the algorithm and a local optimum is the best solution in its neighbourhood. Therefore, the algorithm often ends in a local optimum, which is not yet globally optimal, but there are multiple ways to improve this. Some general possibilities are:

- Plateau Search. Accept also solutions that are just as good as the current solution instead of only better solutions.
- Restart. Do a restart after no improvements are made in the algorithm for multiple iterations. There are two types of restarts, a cold restart, so start the algorithm completely over and compare the results; or a warm restart, so accept all (or part) of the neighbours and continue with those solutions. An example of such an algorithm is Iterative Local Search [6].
- Larger search radii. Use a bigger neighbourhood, so instead of looking only at solutions that differ on one aspect with the current solution, look at solutions that differ more. An example of such an algorithm is Variable Neighbourhood Search [7].
- Accepting worse solutions conditionally. A solution can be accepted by the algorithm if it meets a certain requirement, even if the solution is worse in terms of the objective value.

There are a lot more variations and improvements possible to help the local search algorithm in escaping a local optimum. These local search algorithms that are able to escape local optima, can be split into two main groups, population-based algorithms and trajectory-based algorithms. The second jumps from one solution to the next while the first group generates multiple solutions which together determines the solutions in the next iteration. The population-based algorithms are inspired by behaviours in nature and can be split in three main categories: evolutionary, based on evolution theory; swarm intelligent, based on the movements of groups of animals; and Physical Phenomena, based on concepts in physics. In the next section, these groups and some example algorithms are explained in more detail.

2.2.1 Population-based algorithms

Population-based algorithms are algorithms that keep track of a set/population of solutions instead of moving from one solution to another. Often, they have their foundation in the processes in nature. There are a lot of different algorithms that fall in this group and some of them are quite similar. Therefore, not all of them are explained here. In general, these type of algorithms can escape local optima because it uses multiple solutions to determine the solutions in the next iteration. By using multiple solutions in finding the optimum, it is clearer in which direction the optimum lies. There are three types of population-based algorithms, namely evolutionary, swarm intelligent and physical phenomena.

Evolutionary Algorithms

Evolutionary Algorithms (EA) [8] are algorithms that are based on the biological evolution. The idea of EA is that a population of solutions is created and some solutions in this population survive based on their fitness. Two solutions in the population create some offspring which determines the next iteration/generation of solutions. In general, two solutions are more likely to create offspring if their fitness is better. The fitness of a solution is based on the objective function in comparison to each other, so only the solutions with the best objective value survive. There are different types of determining the population by EA's:

- (1+1)-EA. From one point to another. Thus, there is one parent and one offspring and the new population is either the parent or the offspring. This version of evolutionary algorithm is similar to trajectory-based algorithms like simulated annealing (see next section).

- (μ, λ)-EA. The parents are replaced by the best of the offspring. So, the parents create some offspring by combining the solutions. For the next iteration, only the best solutions of the offspring are used to create a new generation. The number of parents is μ and the number of offspring solutions is λ.
- $(\mu+\lambda)$-EA. The new parents are the best of the parents and the offspring. Thus, it is possible that a parent survives and creates multiple generations of solutions. The number of parents is μ and the number of offspring solutions is λ.

Some examples of EA's are described below.

Genetic Algorithms (GA) [9] are EAs that are based on DNA strings, so the solutions are in the form of a string of (binary) numbers. These strings are recombined or mutated to get the next generation. An example of a GA for multi-objective problems is: *VEGA*. This algorithm selects offspring based on different objectives, one objective at a time.
Another example of a simple GA is *Random Local Search (RLS)* [10]. By RLS, the offspring of a parent is created by flipping one bit of the parent. This bit is chosen uniformly at random. Then the fitness determines whether to use this offspring as next parent.

Evolutionary Strategies (ES) [11] are similar algorithms to GAs. The differences between the algorithms are mainly due to the difference in the form of the solutions [12]. For GA, the solutions are strings of integers, often binary, while for ES, the solutions are represented as real-valued vectors. These real-valued vectors allows ES to utilise much more knowledge of the problem. Moreover, with ES, the new generation can be modified much more and in a more self-adaptive way.

Differential Evolution algorithm (DE) [13] is also a similar algorithm to GA. The algorithm starts with a population of random solutions, but iteratively replaces solutions in the population. The algorithm starts with one solution x and three other distinct solutions in the population a, b, c chosen at random. Now, a new solution is created by randomly changing a variable x_i of solution x to a combination of the values of these variable of the chosen solutions, namely $a_i + F(b_i - c_i)$ where F is a chosen constant and a_i, b_i and c_i the values of variable x_i for solutions a, b and c, respectively. Then, this new solution replaces x if it is better. This algorithm can escape local optima easier because it (randomly) decides for each part of the solution whether to change it or not, so this algorithm has a greater search radius. Moreover, the changes in solutions are based on other solutions in the population, and therefore, not completely random but based on 'good' solutions.

The (basic) EAs still end up in a local optimum quite often. However (for (μ, λ)-EA and $(\mu + \lambda)$-EA), it is less likely to do so since multiple solutions are considered and used to get the new solutions. Often, the best solutions are used to create new ones which helps finding the global optimum. There are more ways that EAs can escape local optima, for example by combining an EA with another technique. Hybrid algorithms that are a combination of an evolutionary algorithm with local search methods are also called *memetic algorithms* [14].

Swarm Intelligent

Swarm Intelligent algorithms (SI) [15] are methods that are based on the movements of swarms or groups of animals. Thus, there is a swarm of solutions and based on the movements of animals, the solutions are moved through the search space to find the optimum. This type of algorithm was

first coined by Gerardo Beni and Jing Wang in 1989 in the context of developing cellular robotic systems. Nowadays, there are multiple algorithms that fall in this category.

One example of a SI algorithm is *Particle Swarm Optimisation (PSO)* [16, 17]. This algorithm 'moves' solutions based on the best known position of the solution being moved and the best overall position of the swarm. Hence, a new solution is explored based on the best known position of a solution so far and on a combination of the current positions of the solutions in the swarm.

Ant Colony Optimisation (ACO) [18] is an algorithm based on the movements of ants. In the ACO algorithm, different ants search for a solution and record their moves and the quality of the found solutions. This information helps the ants in the next iteration in finding a new solution.

Bees Algorithms [19] are inspired on the movement of bees. First, a couple of scout bees explore the search space and land on random solutions. Then, the places with the best solutions are further investigated by foragers. If they find a better solution than the scout, this forager will be the new scout. If none of the foragers can find a better solution, the neighbourhood of this scout is shrunk.

Another example of a swarm intelligent algorithm is *Hunting Search* [20]. This method is inspired by the group hunting of some predators. The predators organise their position to surround the prey (the optimum) based on the position of the others, especially that of their leader. In this way, they slowly enclose the optimum or if the optimum 'escapes', they reform.

There are still a lot more algorithms in this category and still a lot of new algorithms are found based on the idea of groups of animals. Examples are *the Firefly Algorithm*, *the Bat Algorithm, the Stochastic Diffusion Search* and *the Chicken Swarm Optimisation.*
Swarm intelligent algorithms can escape local optima for similar reasons as the evolutionary algorithms, namely by considering multiple solutions at once and using multiple (good) solutions to get the next iteration of solutions. It is assumed that, since the movements work for the group of animals, the movements also work to find the global optimum for optimsation problems.

Physical Phenomena

Physical Phenomena algorithms (PP) are algorithms that are based on the laws of physics. This can be, for example, based on the movement of water or based on known chemical reactions. This category is less commonly known and used than swarm intelligent and evolutionary algorithms, but there are a lot of possible algorithms in this category. Examples of theses algorithms are the *Big Bang-Big Crunch (BBBC)* [21], *Harmony Search (HS)* [22], *Electromagnetic Field Optimisation (EFO)* [23], *Water Evaporation Optimisation (WEO)* [24], *Gases Brownian Motion Optimisation (GBMO)* [25] and *Artificial Chemical Reaction Optimisation Algorithm (ACROA)* [26].
These methods can escape local optima since they are population based, so the next solution is created based on multiple solutions instead of one. This allows the algorithm to explore more (different) solutions.

2.2.2 Trajectory-based algorithms

Trajectory-based algorithms are algorithms that move from one solution to the next and based on some criteria the new solution is either accepted or not. In contrast to the population-based algorithms, trajectory-based algorithms only keep track of one solution at a time and are often not

based on nature. The algorithms in this group employ a wide range of approaches.

A frequently used algorithm in this group is *Simulated Annealing (SA)* [2]. Simulated annealing is similar to (1+1)-GA. However, with GA only solutions with a better objective value survives. For SA, a new solution is either accepted because it has a better objective value or it is accepted with a certain decreasing probability. This makes it easier to escape local optima, since in some cases a solution with a worse objective value is accepted.

Another famous example is *Tabu Search (TS)*, created by Glover [3]. Tabu Search starts with a solution and continues with the best solution in the neighbourhood of the current solution. In the basic local search, the search only continues with the best solution in the neighbourhood when that solution is better than the current solution. Another difference is that solutions (or characteristics of solutions) on the tabu list are removed from the neighbourhood before looking for the best solution. The tabu list consists of solutions (or characteristics of solutions) that have been visited already. This is done to make sure no cycling occurs. There might be more reasons to put a solution on the tabu list. In some variants of TS, the solutions on the tabu list can be accepted if that solution seems promising, for example, if the objective of the solution is the best so far.
TS can escape local optima since some solutions are tabu and are therefore not visited. This makes it possible to move away from a local optimum. Moreover, solutions in the neighbourhood are accepted, even if they are worse than the current solution.

Iterated Local Search (ILS) [6], or *Chained Local Search (CLO) [27]*, is an algorithm that iteratively applies local search to perturbations of the current solution. Thus, first local search is applied to get a (local) optimum. For this step, all kinds of algorithms as described above can be used. Then again local search is applied but now with a perturbation of the found optimum as initial solution. The initial solution of the algorithm is very important for most local search algorithms and largely determines the final solution. Therefore, restarting the algorithm with a perturbation of the found solution allows the algorithm to explore a lot of different solutions and helps with escaping local optima. Multiple local optima are found and hopefully, one of the local optima is (close to) the global optimum.
An example of a more complicated version of ILS is *Variable Neighbourhood Search (VNS)* [7]. This algorithm uses, as perturbation, a change to a different neighbourhood. So, this algorithm jumps to different neighbourhoods after finding a local optimum. In this way, a large part of the solution space is explored. Moreover, the algorithm uses a restart and has a larger search radius. Hence, the search does not have to get stuck in a local optimum.

Guided Local Search (GLS) [28] is a method where the objective function is adapted as soon as a local optimum is reached, so this idea is really similar to the idea of a potential KPI. For GLS, first a local search algorithm is performed, which ends in a (local) optimum. Then, the objective function is adapted by penalising the most important features of this optimum. The most important features are determined with a utilisation function based on the costs associated with the features. Now, the local search algorithm is performed again but with the penalised features ensuring the outcome to be different. After doing this multiple times, the best (local) optimum can be chosen. The hard part in this algorithm is to define useful features and determine the value of the penalty. An example of useful features in the travelling salesman problem is “whether two cities are directly connected in the tour”. The associated costs are the distance between the two cities. The value of the penalty determines how intensely the search space is searched.
GLS can escape from local optima by penalising the found optima such that neighbours of the local

optimum are more attractive than the local optima itself. GLS can also be combined with a lot of other algorithms to make it even less likely to end in a local optimum. This is possible since GLS is simply an iterated local search algorithm with each time a different objective function.

2.3 Adapting the objective function

There are multiple methods that adapt the objective function to make the problem simpler or easier to solve. One of these methods is guided local search. As explained in Section 2.2.2, GLS adapts the objective function based on found local optima. Some other methods found in literature that adapt the objective function to find an optimal solution or simplify the problem are explained in the following subsections.

2.3.1 Equality constraints

The objective function can be adapted by converting equality constraints to components of the objective function. This is done to simplify the problem by reducing the number of constraints. There are two different methods to convert equality constraints to the objective function to solve the problem more easily.

The first method is with help of the *Lagrange multiplier* [29]. For an objective function $f(x)$ with constraint $g(x) = 0$, the new problem has no constraints but an objective function of

$$f(x) + \langle \lambda, g(x) \rangle$$

where λ is the Lagrange multiplier and $\langle \cdot \rangle$ the inner product.

Substitution is another method to adapt the objective function by converting the constraints. This is done by substituting the equality constraints in the objective function. For example, consider the problem with an objective equal to minimising xy and with one constraint $x + y = 4$. With substitution, x is replaced with $4 - y$ in the objective function. The new problem is to minimise $4y - y^2$ with no constraints. If there are no inequality constraints, the problem can simply be solved by finding the stationary points of the adapted objective function.

2.3.2 Soft constraints

In a given problem, there might be some constraints that may be violated. It might be beneficial to relax these constraints to get a larger solution space. Relaxing hard constraints into soft constraints [30] can be done by including the constraint in the objective function with a penalty. Thus, given the following problem:

$$\begin{aligned} \min \ & f(x) \\ \text{s.t.} \ & g(x) \leq 0 \end{aligned}$$

the objective function can be adapted such that the problem becomes:

$$\min \ f(x) + w \ c(g(x))$$

where c is a penalty function and w a weight. The weight can be used to determine importance of various soft constraints. Examples of penalty functions are the absolute value, least squares, and the maximum of a term and zero.

2.3.3 Pareto optimisation

Pareto optimisation is the study of multi-objective problems and finding a compromise between the different objectives. Often, minimising (maximising) one objective will cause the other objective to increase (decrease). This point, where one objective cannot be improved without increasing another objective, is called a Pareto optimal solution. It is important that a balance between the two objectives is found to get the correct Pareto optimum. This can be done with the help of weights. The more important an objective is, the higher weight it gets. It is, however, hard to determine which objective is more important and how much more. An example of an algorithm based on Pareto optimisation is a (1+1)-ES called *(1+1)-PAES*, see Knowles and Corne, [31].
Another method to find the Pareto optimum, is *the weighted sum method*. This method changes the weights of the objectives to find different solutions. A variant of this method is proposed by Kim and De Week, [32]. In this method, the weights are changed adaptively based on the results of the method so far, instead of using a weight selection determined beforehand. This way, the method focuses on unexplored areas.

2.4 Conclusion

There are a lot of different methods to escape a local optimum when using a local search approach. However, the concept of a potential KPI is not yet studied. There is an algorithm that is quite similar, namely Guided Local Search. This algorithm also changes the objective function to move away from local optima. GLS does this by penalising local optima while by potential KPI, solutions that have potential, i.e., are closer to the global optimum, are rewarded. Another difference between GLS and potential KPIs is that GLS searches first for an optimum and then adapts the objective function until some requirement is satisfied, while for potential KPIs, the objective function is only adapted before local search is applied (and maybe at some point back to the original objective function).
There are some algorithms that perform very well, but potential KPIs can still be a good alternative to escape local optima. Potential KPIs are namely build to move the algorithm towards the global optimum which could reduce the number of iterations needed to escape the local optima, in comparison with other more complex algorithms like SA. Whether using potential KPIs is indeed a good technique that helps the basic local search in escaping local optima should be determined by investigating the performance of potential KPIs across different scenarios.

Chapter 3

Potential KPI

A potential KPI is an addition to the objective function that has the goal to help local search algorithms escape from local optima. As explained in Section 2.1, the basic local search algorithm has a good chance to end in a local optimum because it searches for a better solution in just the neighbourhood of the current solution and existing approaches that can escape local optima that are not yet globally optimal, still prefer that local optima over another direction in the search space (see Figure 1.2). A potential KPI can be an extra technique that can help the process of escaping a local optimum and moving towards the global optimum. In this chapter, a definition for a potential KPI is given. Moreover, some initial observations and possibilities are stated and investigated. For example, the method to choose a useful function as potential KPI and the method of employment of that function are discussed.

3.1 Definition

We define a potential KPI to be a function that can be added to the objective function. So, if the objective function is equal to function f then the new objective function is defined as:

$$f^*(S) := f(S) + \nu g(S)$$

where function g is the potential KPI function, S a solution and ν the weight that can be used to determine the importance of the potential KPI. The potential KPI, g, is designed to reward solutions that have 'potential'. A solution that has potential is defined as follows:

Definition 3.1.1 (Potential solution)**.** A potential solution is a neighbourhood solution of the current solution found during a local search algorithm such that this neighbour is worse than the current solution in terms of the objective value, but is closer to the global optimum in terms of some characteristics.

The potential KPI, can be used as extra technique to find better solutions for specific optimisation problems. Good solutions for these problems can be found with any local search heuristic (and other (meta)heuristics), but it might be beneficial to use the adapted objective function f^* in the chosen heuristic to steer the algorithm towards a certain direction. Continuing the algorithm with the potential solution, as described above, steers the algorithm in the direction of the global optimum due to the similarities between the potential solution and the global optimum. Hence, by design of the potential KPI g, the heuristic should be able to escape local optima easier, and therefore, converge to a better solution. For example, the basic local search algorithm cannot escape a local optimum once it ends up in one. With the addition of a potential KPI, the algorithm should be

able to escape this local optimum, and therefore, end potentially in the global optimum, even with this simple algorithm. This makes the technique of potential KPIs attractive, as it has the ability to escape local optima even with simple algorithms. Moreover, a potential KPI has the ability to escape local optima fast since the algorithm is moved away from the local optimum by the potential KPI.
Often other (meta)heuristics like simulated annealing already have a mechanism to escape local optima, so when do you want to use a potential KPI? Using a potential KPI might be beneficial when it takes many iterations and when many worse solutions needs to accepted, in order to escape local optima. Existing methods are not effective in these cases, as they prefer better solutions and tend to return to local optima when many iterations are required to escape them. In Figures 1.2 and 1.3, an example of a solution space where it might be more beneficial to use a potential KPI is shown. In the example, other variants of the local search algorithm has the ability to escape a local optimum, but it still has a high chance to end in the local optimum. Changing the direction, i.e., adapting the objective function with a potential KPI, ensures that the algorithm escapes the local optimum.
A limitation of the method is that the method can only be used if there is some knowledge about the local optima of the problem, since this determines the potential KPI function. So, using a potential KPI might not be useful for all optimisation problems as determining a potential KPI function might be more difficult and time consuming than using more complex methods. For some optimisation problems, it might not even possible to define a useful potential KPI function since no function can be defined that steers the algorithm in the right direction instead of towards a local optimum.
In order to use a potential KPI, some decisions need to be made. Those decisions are 1) which function do you use that rewards solutions with potential; 2) how to employ the addition and for how long and 3) how important is the potential KPI, i.e. determine the weight ν. These decisions are highly dependent on the problem and on the heuristic/algorithm that is used to solve the problem. How to make these decisions is explained in the following sections.

3.2 Choosing the function

The first decision that needs to be made before using a potential KPI, is to define a function g that rewards potential solutions. Defining a good function is very important, since the potential KPI function determines the direction of the search and will determine where the algorithm ends.
The determination of this function is dependent upon the specific optimisation problem, which can pose a challenge. The function needs to be designed such that potential solutions are rewarded. For example, for a minimisation problem the following inequalities should hold:

- $f(S_{\text{current}}) + \nu g(S_{\text{current}}) \geq f(S_{\text{potential}}) + \nu g(S_{\text{potential}})$
- $f(S_{\text{current}}) < f(S_{\text{potential}})$

with S_{current} a current solution and $S_{\text{potential}}$ a potential solution.
As concluded before, some information about the local optima and the global optimum needs to be known in order to design a useful potential KPI function. Function g is namely based on accepting potential solutions which have different characteristics for different optimisation problems. This need for knowledge makes it hard to give a general definition of a function that can be added to the objective function. However, it is possible to give a general strategy to find a potential KPI for a certain optimisation problem.

To determine the potential KPI function, first investigate the expected optimum. Determine especially certain characteristics or features of the optimum. For example, for a scheduling problem with a setup time per type of job, the optimal solution is likely to have the jobs of the same type scheduled consecutively since this reduces the setup times. This feature, jobs of same type scheduled consecutively, can be used to determine a potential KPI.
Besides the global optimum, you should also investigate the local optima and especially the features of these local optima. Now, a potential KPI is a function that rewards (some) features of the global optimum that are not features of the local optima. Choosing a function that rewards this feature, helps in escaping the local optimum, since solutions similar to the local optimum are worse due to the lack of this feature while solutions similar to the global optimum are rewarded.
Considering this strategy, it might be possible to have multiple potential KPIs for one problem. It is namely possible to have multiple local optima that have different features. Every potential KPI can help escape a 'wrong' feature.
Even with the strategy as described above, it can be very hard to determine a useful potential KPI. For very complex and hard to understand problems, it might not be clear what the optimum looks like. Moreover, it is hard to determine all local optima and it is uncertain if it is even possible to define a potential KPI for all problems. Sometimes, local optima are unknown or do not have a clear feature that is different than the features of the global optimum. Finally, it is also difficult to verify if the chosen function is a good potential KPI that allows the search to escape local optima. A way to check if the chosen potential KPI is useful, is by running the local search algorithm with potential KPI for an instance where the algorithm without potential KPI ends in a local optimum and see if it can escape this local optimum. Another option to verify the efficacy of a designed function is to comparing the function with some random noise function. This allows to check whether improved solutions are due to the right direction or due to the possibility of accepting worse solutions. Moreover, the function can be checked by testing the function on some simple benchmark data.

3.3 Mechanisms to employ

How to employ a potential KPI is another decision that needs to be made. Employing the function can be done in multiple ways. Note that the method of employment is especially important, since the potential KPI can steer the algorithm towards a solution that is not optimal for the original objective function but only for the adapted objective function.
In general, a method with potential KPI is divided into multiple phases. Each phase consists of running a local search algorithm for some iterations and the next phase begins with a solution found in the previous phase as initial solution. These phases can have different outlooks, like using the objective function with (a w-phase) or without potential KPI (a ω-phase). The different outlooks of the phases can be determined by answering 3 questions: 1) When does the algorithm uses a w-phase and when a ω-phase?; 2) How does the algorithm employ the potential KPI function, i.e. how to change the value of weight within a phase?; and 3) How long are the different phases of the algorithm?. Some possibilities for these decisions are discussed in this section.

When should the algorithm use the objective function with potential KPI and when the original objective function?

The question answered in this subsection is about when to use the objective function with potential KPI (w) and when the objective function without (ω). Thus, it is discussed which phases in the method have an objective function with potential KPI and without. Hence, it is determined how

the algorithm starts, how it ends and how many phases there are in between.
Starting with a phase with or without potential KPI are both good possibilities. If the algorithm starts with the original objective function, then the algorithm often first finds a local optimum and escapes from it in the later iterations, due to the design of the potential KPI. Moreover, starting without potential KPI has the advantage that the method is always at least as good as the basic local search since a solution with the basic local search is found first and the best solution is often used as final solution. However, starting with the potential KPI function added, might be a good option too, because this immediately steers the algorithm away from the local optimum. Some experiments on different optimisation problems are needed to determine which kind of start of the algorithm performs the best.
How to end the algorithm, can be determined without any experiments. Namely, to get the best solution, you should end the algorithm with a phase with only the original objective function. Indeed, the best solution needs to be the best one for the original objective function and an improvement in the objective function with potential KPI does not always mean an improvement for the original objective function. The improvements in objective value are probably roughly the same for both objective functions, but because the potential KPI rewards a specific feature, this feature might dominate the original objective function. This can be seen in Figure 1.3. The potential KPI steers the algorithm in the right direction, but does not steer the algorithm to the global optimum. However, if now the basic local search is run, the global optimum is found. Moreover, it is expected that once the search escapes a local optimum and is far enough, it cannot enter that local optimum again with the original objective function, because there is a lower point in the search space, hopefully the global optimum, which is now closer. Therefore, the method should end with just the original objective function if the phase with potential KPI has enough iterations.
It is possible to iterate phases with the original objective function and the objective function with potential KPI multiple times. This might help with escaping different local optima. However, using the same potential KPI, probably does not improve the found solution a lot, since it is designed for one local optima which the algorithm (hopefully) already escaped in the first iterations with potential KPI.
It is also possible to add multiple different potential KPI functions to escape local optima with different characteristics. This can be done either simultaneously or consecutively.

How does the algorithm employ the potential KPI function within a phase?

Changing the algorithm from objective function with potential KPI to without potential KPI (and the other way around), can be done in different ways. In general, there are two different methods:

1. *Hard cut.* This method switches between the two possibilities immediately after some time. So, for the complete phase, the potential KPI has a weight ν of zero and for the next (or previous) phase, that weight is equal to some higher value for the whole phase.

2. *Soft cut.* The weight ν of the potential KPI is gradually changed to the desired value by multiplying the weight with a value or adding a value. Hence, within a phase the value of weight is changed bit by bit until some value is reached.

Other possible methods to employ the weight are to keep ν at a value equal to a constant percentage of the original objective; or to change ν every time the need arises i.e. when the algorithm is stuck in a local optimum.

How long are the different phases of the algorithm?

There are roughly two possibilities for determining the length, i.e., the number of iterations, of one phase:

1. Run one phase until the local search method converges. Hence, only start the next phase after the previous one is stuck in an (local) optimum.

2. Run one phase for a fixed number of iterations and continue with the next phase after those iterations. This option might be a good and fast option since it has a fixed time. However, determining the number of iterations is hard. Probably, more than one iteration in a w-phase is needed since it should escape the local optimum and be far away enough. Therefore, it is important that a phase with the potential KPI has enough iterations such that it can completely escape the local optimum.

The number of iterations can be different per phase. The last phase, a ω-phase, should probably be run until it converges since otherwise there might still be a better solution in the neighbourhood.

Used method of employment

In this thesis, only the hard cut method is tested, since in preliminary tests, it was observed that the difference in performance between a soft cut and a hard cut are minimal. It is expected that the hard cut approach is slightly faster and easier to implement since it involves less changes. Hence, the hard cut approach is chosen to test in this thesis.
Moreover, in the experiments in this thesis, the phases are run until they converge. Only this is tested instead of a fixed number of iterations since it is hard to determine which number of iterations should be used and the results are highly dependent on this.
Starting with a phase with or without potential KPI are both tested in this thesis. For local search, the initial solution is very important and the results depend on this initial solution a lot. Therefore, also the start of the total algorithm with potential KPI is important and investigating this is valuable. Thus, two different methods are tested in this thesis, namely the wω-method, starting with a phase with potential KPI until the algorithm converges, then a phase without potential KPI until it converges; and the ωwω-method, starting with a phase without potential KPI until the algorithm converges, then a phase with potential KPI and ending the method with the basic local search.

3.4 Weight value

The importance of the potential KPI function in comparison with the original objective function can be modified with a weight ν. The weight should be high enough that the algorithm can actually escape a local optimum, but should not be so high that the algorithm forgets its original objective. Therefore, it might be useful to determine some bounds for the weights for which the algorithm performs the best. It is expected that the value of weight is dependent on the problem instance since also the height of the objective value is dependent on the instance.
The lower and upper bound of ν, i.e. the minimum and maximum value of weight such that the method performs well, is determined by investigating the algorithm and the potential solutions in more detail. The goal of adding a potential KPI is to accept potential solutions in the local search algorithm. These solutions have a better adapted objective value than the current solution while they do not have a better original objective value than the current solution. So, for potential

solutions of a minimisation problem, the difference in the potential KPI should be larger than or equal to the difference in the original objective between the current solution and the potential solution: $f(S_{\text{potential}}) - f(S_{\text{current}}) \leq \nu\,(g(S_{\text{current}}) - g(S_{\text{potential}}))$. With this inequality, the lower bound for the weight ν can be determined, namely

$$\nu \geq \frac{f(S_{\text{potential}}) - f(S_{\text{current}})}{g(S_{\text{current}}) - g(S_{\text{potential}})}.$$

For a maximisation problem, something similar can be done, but then the upper bound is found:

$$\nu \leq \frac{f(S_{\text{potential}}) - f(S_{\text{current}})}{g(S_{\text{current}}) - g(S_{\text{potential}})}.$$

As expected, these bounds are dependent on the neighbourhood (especially the potential solutions in the neighbourhood) and on the objective value, which is dependent on the instance: for example, the value of the processing times of jobs.
Determining the other bound for the weight ν, is harder. From preliminary tests on minimisation problems, it seems like there is some value for which the algorithm becomes less effective if the weight is higher. To investigate this bound, some experiments with different values of weight are performed.
Thus, for this thesis, some random, instance-independent values of weight are tested since determining the weight that gives the best performance is difficult. However, there is one bound for the weight which can be used to test an instance-dependent weight.

Chapter 4

Bucketised Planning Problem

The bucketised planning problem is an optimisation problem frequently used by OMP to plan certain tasks. Specifically, the goal is to determine the start times of the given tasks. The problem is as follows:
There is one machine on which products are processed, for example, a machine that cuts metal wires into shorter pieces. Different tasks can be performed on that machine, for example "cut 50 meter of aluminum wire". Each task corresponds to an order placed by a customer, so each task has a due date. The due date is a soft constraint: the objective is to minimise the total lateness. Each task also has a process time since the duration of tasks differs, for example to cut a 100 meter wire takes longer than to cut a 1 meter wire.
To plan all the tasks, the time horizon in which tasks are planned is divided into 'buckets'. A bucket can for example be equal to one week. In the problem, we keep track of the 'load' per bucket. The load of a bucket is how much processing time there is planned in this bucket. So, for example if a task with a process time of one hour is scheduled in the first week, then this task contributes a load of 1 hour to that bucket (week 1). It is also possible that a task is planned such that it ends in a different bucket than where it started. In that case, the task contributes some load to one bucket and some to the other. For example, assume a task with a load (or process time) of 100 is planned such that it falls for 70% in bucket 1 and for 30% in bucket 2, then this task contributes a load of 70 to bucket 1 and the rest, 30 to bucket 2. Another option is that the process time of a task is greater than the time in one bucket. In that case, the load of that task can span even more than two buckets.
Note that in this planning problem, it does not matter if tasks are processed at the same time. In a later scheduling problem, the exact order to execute all the tasks, is determined. Hence, it is possible to plan everything at the start such that all due dates are satisfied. However, in that case, all the planned tasks are probably more than what the machine can handle in a week. The time is divided in these buckets with load to account for this. Thus, in the bucketised planning problem, it is not checked whether tasks are scheduled at overlapping times, but only whether the total load does not exceed the capacity of a bucket for the time being. For example, the load cannot contain more than 168 hours if a bucket is defined as one week. The total capacity of a bucket is a soft constraint, so the objective is to minimise the amount of load that exceeds the capacity. This constraint is treated as a soft constraint because enforcing it as a hard constraint results in a scheduling problem with very high computation times when planning for a large time horizon. Therefore, solving the BPP first and then addressing the scheduling problem for shorter time periods is more practical.
In the bucketised planning problem, it is therefore determined for each task at what time the task

starts. This start time (a continuous variable) falls in a certain bucket and determines the load that it contributes to this bucket. The goal is to minimise the weighted sum of the following two KPIs:

- Overload: the sum of, per bucket, how much more load (in time) is planned than the actual capacity (for example 168 hours for a week).
- Lateness: the sum of, per task, how much later the task is finished than the due date of the task.

Hence, the objective function consists of two parts. The Overload objective is typically a lot more important than the Lateness objective: it can be explained to a customer that his order will be two weeks late. However, there are no more hours in a week, and hence, in reality an overload is not possible. Therefore, the weight for the Overload objective should be higher than the weight for the other objective.

4.1 Formal definition

The bucketised planning problem can be formulated more formally into an mixed integer linear program (MILP). This is done to give more insight in the problem and such that the optimal solution can be determined using an ILP solver. The MILP for the BPP is as follows:

$$\begin{array}{lll} \min & \lambda_1 \sum_{i \in I} R_i^1 & + \quad \lambda_2 \sum_{j \in J} R_i^2 \\ \text{s.t.} & & \end{array}$$

$$\begin{array}{rcll}
t_j - M(1 - X_{ij}) \leq & S_i & \leq t_{j+1} + M(1 - X_{ij}) & \forall i \in I, j \in J \quad (1) \\
t_j - M(1 - Y_{ij}) \leq & S_i + p_i & \leq t_{j+1} + M(1 - Y_{ij}) & \forall i \in I, j \in J \quad (2) \\
& \sum_{j \in J} X_{ij} & = 1 & \forall i \in I \quad (3) \\
& \sum_{j \in J} Y_{ij} & = 1 & \forall i \in I \quad (4) \\
X_{ij} + Y_{ij} - 1 \leq & Z_{ij}^{11} & \leq X_{ij} & \forall i \in I, j \in J \quad (5) \\
& Z_{ij}^{11} & \leq Y_{ij} & \forall i \in I, j \in J \quad (6) \\
X_{ij} - Y_{ij} \leq & Z_{ij}^{10} & \leq X_{ij} & \forall i \in I, j \in J \quad (7) \\
& Z_{ij}^{10} & \leq 1 - Y_{ij} & \forall i \in I, j \in J \quad (8) \\
Y_{ij} - X_{ij} \leq & Z_{ij}^{01} & \leq 1 - X_{ij} & \forall i \in I, j \in J \quad (9) \\
& Z_{ij}^{01} & \leq Y_{ij} & \forall i \in I, j \in J \quad (10) \\
\left(\sum_{j'=0}^{j-1} X_{ij'}\right) + \left(\sum_{j'>j} Y_{ij'}\right) - 1 \leq & Z_{ij}^{00} & \leq \left(\sum_{j'=0}^{j-1} X_{ij'}\right) & \forall i \in I, j \in J \quad (11) \\
& Z_{ij}^{00} & \leq \left(\sum_{j'>j} Y_{ij'}\right) & \forall i \in I, j \in J \quad (12) \\
t_{j+1} - S_i - (1 - Z_{ij}^{10})BT \leq & C_{ij}^{10} & \leq t_{j+1} - S_i + (1 - Z_{ij}^{10})BT & \forall i \in I, j \in J \quad (13) \\
0 \leq & C_{ij}^{10} & \leq BT Z_{ij}^{10} & \forall i \in I, j \in J \quad (14) \\
S_i + p_i - t_j - (1 - Z_{ij}^{01})BT \leq & C_{ij}^{01} & \leq S_i + p_i - t_j + (1 - Z_{ij}^{01})BT & \forall i \in I, j \in J \quad (15) \\
0 \leq & C_{ij}^{01} & \leq BT Z_{ij}^{01} & \forall i \in I, j \in J \quad (16) \\
0 \leq & S_i & \leq BT & \forall i \in I \quad (17) \\
0 \leq & R_i^1 & & \forall i \in I \quad (18) \\
S_i + p_i - d_i \leq & R_i^1 & & \forall i \in I \quad (19) \\
0 \leq & R_j^2 & & \forall j \in J \quad (20) \\
\sum_{i \in I} L_{ij} - T \leq & R_j^2 & & \forall j \in J \quad (21)
\end{array}$$

There are a lot of variables and constraints in the MILP formulation of the BPP as given above. These constraints and variables are now explained in more detail.
The **input** given by the problem instance is as follows:

- Some tasks $i \in I$ that needs to be processed with:
  - Due date d_i, the date that a task i should be finished.
  - Process time p_i, the time it takes to process task i.
- A time horizon in which the tasks are planned, divided in a set of buckets J consisting of B buckets of length T. For $j \in J$, the bucket j is defined as $[t_j, t_{j+1}]$.
- Some constants λ_1, λ_2 determining the ratio between the soft constraints, here typically $\lambda_1 < \lambda_2$.

The first basic **variables** needed to define the BPP problem are the following three:

$$S_i = \text{ Start time of task } i \in I \text{ (End time } E_i = S_i + p_i)$$

$$X_{ij} = \begin{cases} 1 & \text{if task } i \text{ starts in bucket } j \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ij} = \begin{cases} 1 & \text{if task } i \text{ ends in bucket } j \\ 0 & \text{otherwise} \end{cases}$$

Some **constraints** can be introduced to ensure that these variables have the same meaning as previously defined. Hence, $X_{ij} = 1$ if and only if $t_j \leq S_i \leq t_{j+1}$ and every task must start in exactly one bucket j. The same conditions hold for Y_{ij}. To formulate this by mathematically useful constraints, a large **constant** M is introduced. The constraints that follows from this, are inequalities (1)-(4).
Another **variable** needed for the problem is the load of a task $i \in I$ in bucket $j \in J$, L_{ij}. To define this variable, five different cases are considered:

1. Task i is completely processed in bucket j, so start time and end time are in bucket j, so $S_i \geq t_j$ and $E_i \leq t_{j+1}$. In this case, the load of task i on bucket j is the process time of task i, p_i. Moreover, $X_{ij} = 1$ and $Y_{ij} = 1$.
2. Task i starts in bucket j but ends in a later bucket, so $t_j \leq S_i < t_{j+1}$ and $E_i > t_{j+1}$. In this case, the load is the time from the start of task i to the end of the bucket, hence $t_{j+1} - S_i$. Furthermore, $X_{ij} = 1$ and $Y_{ij} = 0$.
3. Task i does not start in bucket j but does end in this bucket, so $S_i < t_j$ and $t_j \leq E_i \leq t_{j+1}$. For this case, the load is the time until the end time, so end time minus the start of the bucket $S_i + p_i - t_j$. Also, $X_{ij} = 0$ but $Y_{ij} = 1$.
4. The duration of task i is larger than the bucket size T but is partly processed in bucket j, so $S_i < t_j$ and $E_i > t_{j+1}$. The load of task i on bucket j is the total time of the bucket, T and $X_{ij} = 0$ and $Y_{ij} = 0$.
5. Task i is not planned to be processed in bucket j, so the load is 0 and both $X_{ij} = 0$ and $Y_j = 0$. This is the case if either $E_i < t_j$ or $S_i > t_{j+1}$.

From these cases, the load of task $i \in I$ on bucket $j \in J$ can be defined as follows:

$$
\begin{aligned}
L_{ij} =& X_{ij}Y_{ij}p_i && \text{Task } i \text{ is completely contained in bucket } j, \text{ hence the load is process time } p_i. \\
&+ X_{ij}(1 - Y_{ij})(t_{j+1} - S_i) && \text{Task } i \text{ starts in bucket } j, \text{ but does not end there, hence the load is start until end bucket } t_{j+1} - S_i. \\
&+ (1 - X_{ij})Y_{ij}(S_i + p_i - t_j) && \text{Task } i \text{ ends in bucket } j, \text{ but does not start there, hence the load is start bucket until end task } S_i + p_i - t_j. \\
&+ \left(\sum_{j^*=0}^{j-1} X_{ij^*}\right)\left(\sum_{j^*=j+1}^{J} Y_{ij^*}\right)T && \text{Task } i \text{ spans bucket } j \text{ completely, so } i \text{ does not start and end in } j \text{ and load is } T.
\end{aligned}
$$

However, this definition of the load is not linear. In order to solve the bucketised problem with a linear programming solver like Gurobi, the formulation should be made linear. This can be done by adding an extra variable equal to the product of two variables. So, the extra variables for all $i \in I$ and $j \in J$ are:

$$
\begin{aligned}
Z_{ij}^{11} &= \begin{cases} 1 & \text{if } X_{ij}Y_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \\
Z_{ij}^{10} &= \begin{cases} 1 & \text{if } X_{ij}(1 - Y_{ij}) = 1 \\ 0 & \text{otherwise} \end{cases} \\
Z_{ij}^{01} &= \begin{cases} 1 & \text{if } (1 - X_{ij})Y_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \\
Z_{ij}^{00} &= \begin{cases} 1 & \text{if } \left(\sum_{j'=0}^{j-1} X_{ij'}\right)\left(\sum_{j'>j} Y_{ij'}\right) = 1 \\ 0 & \text{otherwise} \end{cases} \\
C_{ij}^{10} &= Z_{ij}^{10}(t_{j+1} - S_i)_{\geq 0} \\
C_{ij}^{01} &= Z_{ij}^{01}(S_i + p_i - t_j)_{\geq 0}
\end{aligned}
$$

Note that Z^{11}, Z^{10}, Z^{01} and Z^{00} are binary variables and C^{10} and C^{01} continuous variables. C^{10} is equal to 0 if task i does not start in bucket j and equal to $t_{j+1} - S_i$ otherwise. Now the load per bucket $j \in J$ per task $i \in I$ is linear, namely:

$$L_{ij} = p_i Z_{ij}^{11} + C_{ij}^{10} + C_{ij}^{01} + T Z_{ij}^{00}.$$

There are a number of **constraints** that apply to these variables, ensuring that their meaning remains consistent. Note that the Z variables are introduced to replace the product of two binary variables. This gives the constraints (5)-(12). For example, for Z_{ij}^{11}, the constraint $Z_{ij}^{11} \geq X_{ij}+Y_{ij}-1$ should be satisfied since this ensures that $Z^{11} \geq 1$ only when both $X_{ij} = 1$ and $Y_{ij} = 1$. Moreover,

$Z^{11}_{ij} \leq X_{ij}$ and $Z^{11}_{ij} \leq Y_{ij}$ as $Z^{11} \leq 1$ only when both binary variables equal to 1. The same holds for the other Z variables, which leads to the constraints (5)-(12).
Variables C are introduced to replace the product of a binary and continuous variables. Thus, the corresponding constraints are slightly different than they were for the Z variables. The constraints for these products should assure that it is either the value of the continuous variable or 0. Hence, when the binary variable is 1, the C variable should be squeezed by the continuous value and by zero otherwise. Constraints (13)-(16) are constructed such that this holds.
Note that for these constraints, an upper bound is used. This upper bound is the end of the time horizon used, so that is the number of buckets B times the length of one bucket T, i.e. the end of the time horizon is BT. This is the upper bound because the difference between the start time S_i (or end time) of a task i and the start t_j (or end) of a bucket j can be at most the length of the complete time interval, BT. All the continuous variables should be bound by this upper bound and by the start of the interval ($= 0$). This results in constraints (14), (16) and (17).
Finally, we have the **objective function** of the MILP. Note that the goal of the problem is to determine the start times of when to process a task such that the lateness is minimised. For task $i \in I$, the lateness is $S_i + p_i - d_i$ if task i is indeed late, hence if the end time is later than the due time. Thus, the function to minimise for all tasks is $\max(S_i + p_i - d_i, 0)$. To linearise this, a new **variable** R^1_i is introduced with corresponding constraints (18) and (19).
Also, the overload in each bucket is minimised, so $\min \sum_{i \in I} L_{ij} - T$. For this part of the objective function, the same holds. Only buckets that are overloaded ($\sum_{i \in I} L_{ij} \geq T$) contribute to the objective function, so again a variable R^2_j and two constraints (20) and (21) are added to the problem. Note that no upper bound for R^2_j and R^2_j are given since the objective is to minimise. Hence, the lowest value possible is always used in the solution as long as the constants λ_1 and λ_2 are greater than zero.

4.2 Local search algorithm

For the bucketised planning problem, a simple local search algorithm is used. The algorithm starts with an initial solution for the problem and continues the search as follows:

1. Create all the neighbourhood solutions and calculate their objective value. Hence, for all tasks $i \in I$ and for all $t \in [-h, h]$ where h is a chosen step size:
   - Create a new solution by moving the start date of task i to the current start date plus t, $S_i + t$.
   - Check if all constraints are still satisfied, i.e. whether the start and end time of task i are in the given time horizon.
   - If the constraints are satisfied, then calculate the objective value of this new solution.
2. Select the neighbour with the best (lowest) objective value. If multiple solutions have the same, best objective value, then one of those solutions is selected uniformly at random.
3. If the objective value of the selected neighbour is better than or equal to the objective value of the current solution, choose this neighbour and repeat the previous steps with this solution as current solution.
4. When the objective value has not been improved for a number of iterations terminate the algorithm. This number of iterations is chosen to be equal to the number of tasks, such that there is a possibility that the complete plateau is searched.

The best solution, found with this algorithm and the original objective function, is simply the last solution found, since only better solutions are accepted and explored. However, when using this algorithm with a potential KPI added to the objective function, this does not have to be the case. It is possible that the algorithm visits solutions that are better than the last solution of the algorithm. Therefore, three possible algorithms occur:

1. Continue every phase with the solution that was last found with the algorithm and use this solution as final solution.
2. Continue every phase with the solution that was last found with the algorithm, but use the best solution that has been explored as final solution.
3. Continue every phase with the best solution found so far and use this solution as final solution.

Here, a phase is as described in Section 3.3, i.e. a complete run of the local search algorithm either with or without potential KPI. It is expected that the second variant of the algorithm works the best, since it remembers which visited solution was the best but does continue with the solution found with potential KPI which might be a worse solution in terms of the original objective value but has 'potential'.

4.3 Potential KPI

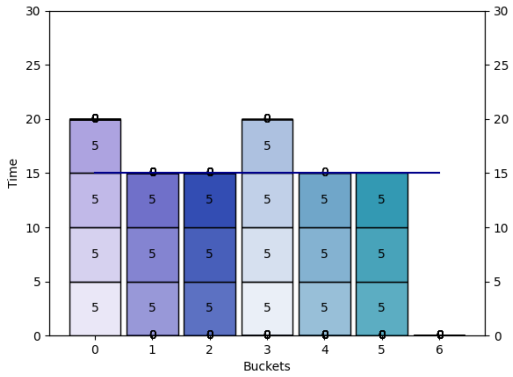


Figure 4.1: An example of a initial position for an instance of the bucketised planning problem where local search ends in a local minimum.

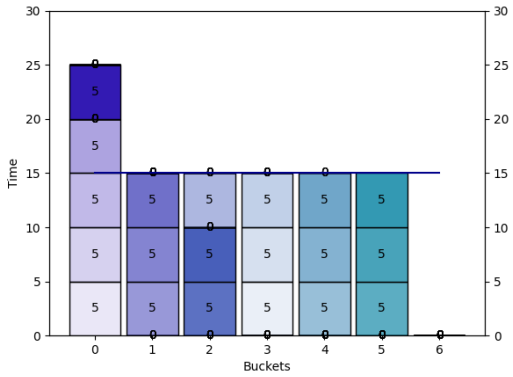


Figure 4.2: A local minimum of the bucketised planning problem which occurs when using the instance as shown in Figure 4.1 with a step size $h = 2$ buckets.

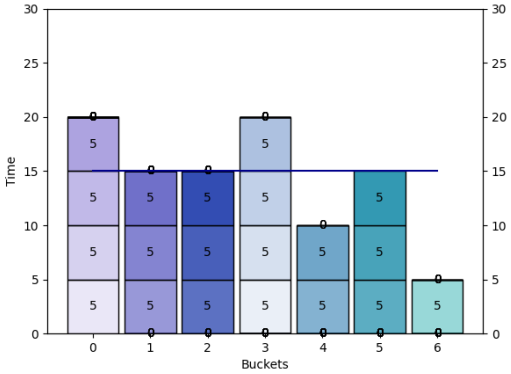


Figure 4.3: A potential solution of the bucketised planning problem for a position as shown in Figure 4.1.

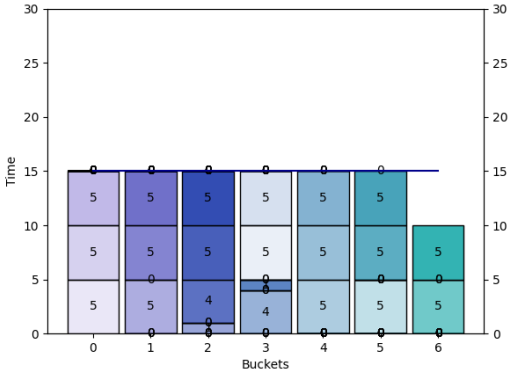


Figure 4.4: The optimum of the bucketised planning problem for the given instance as shown in Figure 4.1.

The basic local search algorithm without potential KPI as explained in Section 4.2 is likely to end in a local minimum that is not yet globally optimal (see Chapter 2). To create a potential KPI, it is important to know what such a local minimum for the BPP looks like, as explained in Section 3.2. An example of a local minimum, that is not the global minimum, is shown in Figure 4.2. In the example, the problem instance consists of a time frame of 7 buckets with a capacity of 15 and 20 tasks where all tasks have process time 5 and a due date of 0, so the start of the time frame. The initial solution for the local search algorithm is as shown in Figure 4.1 and the neighbours are created by moving the start time of a task maximal 2 buckets ($2 \cdot 15$ time units) later or earlier than its current start time. Running local search for this example with these settings results in the local minimum as shown in Figure 4.2. Indeed, moving a task to the right, later in time, increases the objective value since the overload stays the same but a task is finished later. Hence, the local search algorithm only moves tasks to the left and the final solution is indeed the local minimum as shown in Figure 4.2. This final solution is not the global minimum since there is a solution with a better objective shown in Figure 4.4. There is no overload in this solution which makes the objective value of this solution lower.
In this example, moving a task later in time does not improve the objective value. However, it might create potential to improve the objective value in later iterations. For example, moving a task from bucket 4 to bucket 6, as shown in Figure 4.3, creates room for a task of bucket 3 which eventually reduces the overload. With a potential KPI, these moves must be rewarded. Hence, the solution, shown in Figure 4.3, must have a lower value of the objective function plus potential KPI than the initial solution, as shown in Figure 4.1, has.

SquaredLoad

As previously stated, the optimal solution for a bucketised problem is one with the lowest possible overload. In many cases, this means no overload at all. A potential KPI can help by rewarding a more equal spread between buckets, because this means that overload is less likely to occur. There are probably multiple possible potential KPIs that ensure a more equal spread between buckets. One is to square the load per bucket, which we call 'SquaredLoad'. Hence, this potential KPI

function is defined as follows:

$$g(S) := \sum_{j \in J} \left(\sum_{i \in I} L_{ij}(S) \right)^2$$

where S is a solution for the problem. It can be verified that with this method, the algorithm can escape the local minimum in the example instance shown in Figure 4.2.

4.4 Experimental set-up

In this section, the experiments to test the performance of the SquaredLoad function as potential KPI for the bucketised planning problem are explained.
All the experiments are run on the DelftBlue supercomputer [33]. Furthermore, the experiments are run with the second variant of the algorithm explained in Section 4.2 since this algorithm is expected to perform best. The other variants are shortly investigated as well and the results of this are shown in Appendix A. Indeed, the second variant performs the best, although the first variant only performs slightly worse.
First, the SquaredLoad potential KPI is tested for some simple instances that are similar to the example in Section 4.3. The results of these tests are shown in Appendix A.
The first tests of the potential KPI for the bucketised planning problem are promising. However, to have some more conclusive results, more elaborate tests are needed. These tests are performed on a dataset consisting of 100 instances with buckets with a capacity of 15. The instances vary in number of tasks: there are 10 instances with 10 tasks, 45 with 20 tasks, 25 with 30, 15 with 40 and 5 instances with 50 tasks. These number of tasks are chosen, because this gives enough variation in size within the dataset but also keeps the total running time of the tests reasonable. For each task, the process time and due date is randomly set to a value between 0 and 15 and between 0 and 75, respectively. These values are chosen such that it gives plausible problem instances with enough variety within the 100 instances. To get instances that are more like the problems in reality, the number of buckets are chosen such that all the tasks can be processed within the capacity, so the number of buckets is given by $B = \lceil \frac{\sum_{i \in I} p_i}{T} \rceil$ where T is the length of one bucket. Moreover, the size used to determine the neighbourhood in the local search algorithm, i.e. the maximum amount of time a task can be moved forward or backwards to get a new neighbour, is equal to two times the capacity of the buckets ($= 2 \cdot 15 = 30$). This neighbourhood is the same as the one used in the example in Section 4.3. Finally, the constants in the objective function, λ_1 and λ_2, are set to 1 and 1000, respectively.
The SquaredLoad function is tested with two different initial solutions. The initial solution is very important in the local search algorithm, and therefore, it is interesting to see how the methods with potential KPI perform with different initial solutions. The first initial solution that is considered, is a random solution. This solution is constructed by choosing a start time uniformly at random for every task such that all start and end times are within the time horizon. Since this solution is random, it likely has a poor objective value and the difference between the solution found with the basic local search and the initial solution is potentially big. However, for the BPP, the random solution is not extremely bad, because the tasks are distributed across the buckets due to the uniform random selection of start times. Also, a second initial solution is used, a greedy solution. The greedy solution is constructed by filling the buckets with tasks until the capacity of the bucket is reached. With this greedy approach, the tasks with the earliest due date are planned first. If there are tasks left that do not fit in any buckets without causing overload, then these tasks are placed in the first bucket to minimise the possible lateness. This approach is used as greedy initial

solution since it is an intuitive method resulting in a good solution. The overload is namely small since buckets are filled up to their capacity and by starting with the earliest due dates, also the lateness is limited.
Besides the SquaredLoad function, a random function is tested as potential KPI. The experiments with a random function are performed to determine whether it is important to construct a function in a specific manner or that a random function is sufficient. If a random function is sufficient, then it is enough to accept some random solutions instead of steering the algorithm towards solutions with similar characteristics as the global optimum. The random function that is used for the experiments is $\sum_{i \in I} a_i S_i$ with a_i, $i \in I$ random integers between -5 and 5. Note that this function is linear, so has a fast runtime.
In the experiments, two different methods, starting with the adapted objective function, wω and starting with the original objective function, ωwω, are compared and tested. It is also investigated how much improvement there is when an additional run with the potential KPI is done, so methods wωwω and ωwωwω.
The four methods are investigated for different values of the weight ν of the potential KPI. The methods are first tested with values that are chosen arbitrarily between 1 and 100 and are the same for every instance. It is expected that the optimal weight, the value which gives the best results, is dependent on the value of the objective function and more specifically on the instance, as explained in Section 3.3. The optimal value for the weight is not known and is probably hard to determine. However, it is possible to determine a lower bound. A lower bound for the weight is found in Section 3.3 and is given by: $\nu \geq \frac{f(S_{\text{potential}}) - f(S_{\text{current}})}{g(S_{\text{current}}) - g(S_{\text{potential}})}$. It is hard to calculate the exact value of this bound for the bucketised planning problem with the SquaredLoad function as potential KPI since every potential solution is different and has a different objective value (in comparison with its current solution). Therefore, an estimate of the maximum of this bound is investigated. This estimate is used, since taking the maximum of the bound assures that the weight ν is high enough for all potential solutions. Hence, the estimate would probably be too high for most solutions and only gives a robust idea of where the real bound for the weight is. To determine this estimate for the maximum of the bound, first an estimate of the numerator is determined. This is done for a potential solution with characteristics as the one defined in Section 4.3:

$$\begin{aligned} f(S_{\text{potential}}) - f(S_{\text{current}}) &= \text{difference in OverLoad} + \text{difference in TooLate} \\ &= \text{difference in TooLate} \\ &\leq h = 2 \cdot \text{capacity of a bucket} \end{aligned}$$

since in one iteration only one task can be moved for at most the step size h which is equal to twice the capacity of the bucket for all the tested instances, so the moved task can be at most twice the capacity extra too late. Moreover, the overload is the same for the current and the potential solution as can be seen in Figure 4.3. Note that this is the case because it is assumed that the potential KPI looks as depicted in Figure 4.3, where creating overload to reduce lateness is never beneficial and reducing overload will already result in a lower objective value. Thus, for the tested instances, the numerator is equal to two times the bucket size which is equal for all instances, so the estimate for the numerator is $2 \cdot 15 = 30$ time units. Now, an estimate of the minimum of the denominator of the bound has to be found. Note that a potential solution is a neighbour of the current solution for which the value of the potential KPI is better than the potential KPI value of the current solution. Hence, $g(S_{\text{current}}) - g(S_{\text{potential}}) > 0$. Moreover, to create a potential solution, a task in the current solution is moved later in time such that some load of that task is moved from one bucket to a bucket with less load. Note that only the load of these two buckets changes in the potential solution. To determine the estimate for the weight bound, we define two buckets x and y

with in the current solution a load of L_x and L_y, respectively. We also define $a > 0$ to be the load that is moved from bucket x to bucket y while creating the potential solution. Thus, the load in the potential solution is equal to $L_x - a$ and $L_y + a$ respectively. Hence, the difference in potential KPI is

$$\begin{aligned} g(S_{\text{current}}) - g(S_{\text{potential}}) &= L_x^2 + L_y^2 - ((L_x - a)^2 + (L_y + a)^2) \\ &= L_x^2 + L_y^2 - L_x^2 + 2aL_x - a^2 - L_y^2 - 2aL_y - a^2 \\ &= 2aL_x - 2aL_y - 2a^2 \\ &= 2a(L_x - L_y - a). \end{aligned}$$

Since this difference should be greater than zero and $a > 0$, we know that $L_x > L_y + a$. Given that the problem is discrete, $a \geq 1$ and we have

$$g(S_{\text{current}}) - g(S_{\text{potential}}) \geq 2.$$

Thus, the estimate for the denominator of the weight bound is equal to 2. And hence, the estimate for the lower bound for the weight ν for the bucketised planning problem is equal to $\frac{2*\text{capacity of a bucket}}{2} = \frac{30}{2} = 15$ for all used instances. Hence, it can be said that the methods with potential KPI have a guaranteed impact from the potential KPI for weights higher than 15. The value of the weight bound is tested through a series of experiments in which multiple fractions of the bound are used as a weight in the tested methods. This process allows for the investigation of the discrepancy between the estimated and the actual bound.
To investigate the behaviour of the potential KPI and the impact of the different parameters described above, the results of the experiments are shown in boxplots. Most of the boxplots depict the objective values of the different instances such that a value of 1 means that it is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. So, the value depicted in the boxplot for a solution S found with one of the potential KPI methods, is constructed by using the following function:

$$h(S) = \begin{cases} \frac{f(S_{\text{initial}}) - f(S)}{f(S_{\text{initial}}) - f(S_{\text{without}})} - 1 & \text{if } f(S) \leq f(S_{\text{without}}) \\ \frac{f(S_{\text{without}}) - f(S)}{f(S_{\text{without}}) - f(S_{\text{optimum}})} & \text{if } f(S) \geq f(S_{\text{without}}) \end{cases}$$

where f is the objective function, S_{initial} the initial solution, S_{without} the solution found with the standard local search and S_{optimum} the optimal solution. The optimal solution is found by solving the ILP defined in Section 4.1 with Gurobi. This scaling is chosen, because the goal of the tests is to see how good the methods with potential KPI perform in comparison with the basic local search method, so without potential KPI. It is especially interesting to see this with respect to the optimal value, since ultimately that is the value to search for. Since the methods with potential KPI can also perform worse than the one without potential KPI, a reference value for -1 is needed. This value is set at the objective value of the initial solution since it is not possible to find a lower value. Thus, in the used boxplots, a box close to one means that the method works really well, since the values are close to the optimum; a box around zero means that this method is just as good as using no potential KPI; and a box lower than zero means that the method is worse than the simple local search without the use of a potential KPI. These boxplots are used to investigate the different values of weight and to compare the different methods. To investigate the performance in terms of runtime and number of iterations for the different methods, a histogram with the average value per method is used.

Statistical tests

After conducting the experiments described above, some results are compared and analysed using a statistical hypothesis test [34]. Such a statistical test is a method to determine whether the data, or results in our case, significantly support a particular hypothesis. The test is used to calculate a test statistic and a p-value. The p-value is the probability of rejecting the null hypothesis when that hypothesis is actually true.
To use a statistical test on our results, we first need to define a null hypothesis by identifying what we want to compare. The goal is to determine whether using a potential KPI in the local search algorithm is an effective technique and performs better than the basic local search method. Thus, we need to compare which method yields the best objective value. However, to compare two methods in a meaningful way, not just the objective values are used, but a scaled version that also takes the optimal value into account. The optimal value needs to be taken into account since the goal of the local search approach is to find a solution that is as close as possible to the global optimum. Thus, the tested value or independent variable is a measure of how close the found objective value is to the optimal value compared to the full spectrum, i.e. the difference between the objective value of the initial solution and the optimal value. Therefore, the tested value is defined as follows:

$$\frac{f(S) - f(S_{\text{optimum}})}{f(S_{\text{start}}) - f(S_{\text{optimum}})}.$$

As previously stated, we want to determine whether the objective value derived from one method differs significantly from the objective value obtained from the other method. This can be achieved by defining a null hypothesis as follows:

$$H_0 : \frac{f(S_{\text{Method 1}}) - f(S_{\text{optimum}})}{f(S_{\text{start}}) - f(S_{\text{optimum}})} = \frac{f(S_{\text{Method 2}}) - f(S_{\text{optimum}})}{f(S_{\text{start}}) - f(S_{\text{optimum}})}$$

where f is the objective function, $S_{\text{Method 1}}$ the solution found with 'Method 1' and $S_{\text{Method 2}}$ the solution found with 'Method 2'. Hence, method 1 and method 2 are compared to see if their scaled objective values are similar. Methods that are compared in this thesis, are the wω-method and the ωwω-method with the basic local search method. This is done for both the SquaredLoad as random potential KPI. Moreover, the random function is compared with the SquaredLoad function. Also, both methods with the SquaredLoad potential KPI function are compared with the methods with an extra phase, i.e. wωwω-method and wωwω-method.
If the null hypothesis is rejected, then the alternative hypothesis ($\frac{f(S_{\text{Method 1}})-f(S_{\text{optimum}})}{f(S_{\text{start}})-f(S_{\text{optimum}})} \neq \frac{f(S_{\text{Method 2}})-f(S_{\text{optimum}})}{f(S_{\text{start}})-f(S_{\text{optimum}})}$) is accepted, i.e. the two methods are significantly different and, based on the figures, it can be seen which method is better. The null hypothesis is rejected when a p-value smaller than $\alpha = 0.05$ is found.
There are a lot of statistical tests available to test the null hypothesis such as the student's t-test [35], the Chi-square test [36] and the Kruskal-Wallis test [37]. To determine the correct tests, the results can be seen as data points. The dependent variables in the data are the tasks with the given information (due date and process time) and the independent variable is the scaled objective value, as defined above. There are two independent groups of data, the objective values found with the first method and the objective values found with the second method. For our null hypothesis, we want to compare the median of both groups/methods.
There are some observations that holds true for these points, namely

- The dependent variable is ordinal. Hence, the information of the tasks is not necessarily continues but has a set order.

- The independent variables are two independent, categorical groups, namely a group with objective values found with one method and a group with objective values found with the other method.
- The data points are independent, so the objective values found with method 1 of different instances do not dependent on each other or on the objective values found with method 2.
- The data points are not normally distributed. Although the exact distribution of the objective values is uncertain, it is expected that they are not normally distributed since the objective values are always integers, due to the input (integer due dates and process times), making a normal distribution highly unlikely. To check this assumption, the Kolmogorov-Smirnov test [38] can be used to test for normality. However, this test can only determine if a distribution is likely to be normal, not whether it is not normal.
- The two groups of data points follow the same distribution since they are both found with a similar local search approach on the same input.

Since the distribution of the results is unknown, a non-parametric test needs to be used. Examples of non-parametric tests [39] are the 1-sample sign test, the Friedman test, the Kruskal-Wallis test, the Mann-Whitney U-test and the Spearman Rank Correlation. Based on the observations, the Mann-Whitney U test [40] seems to be the best, since this test compares the median of two groups of data given that its dependent variable is at least ordinal.
The Mann-Whitney U test uses the U statistic as test statistic to test the difference in medians of two groups. If $X_1, \dots X_{n_1}$ is an independent identically distributed sample from the first group and $Y_1, \dots Y_{n_2}$ an independent identically distributed sample from the second group, then the U statistic is defined as follows:

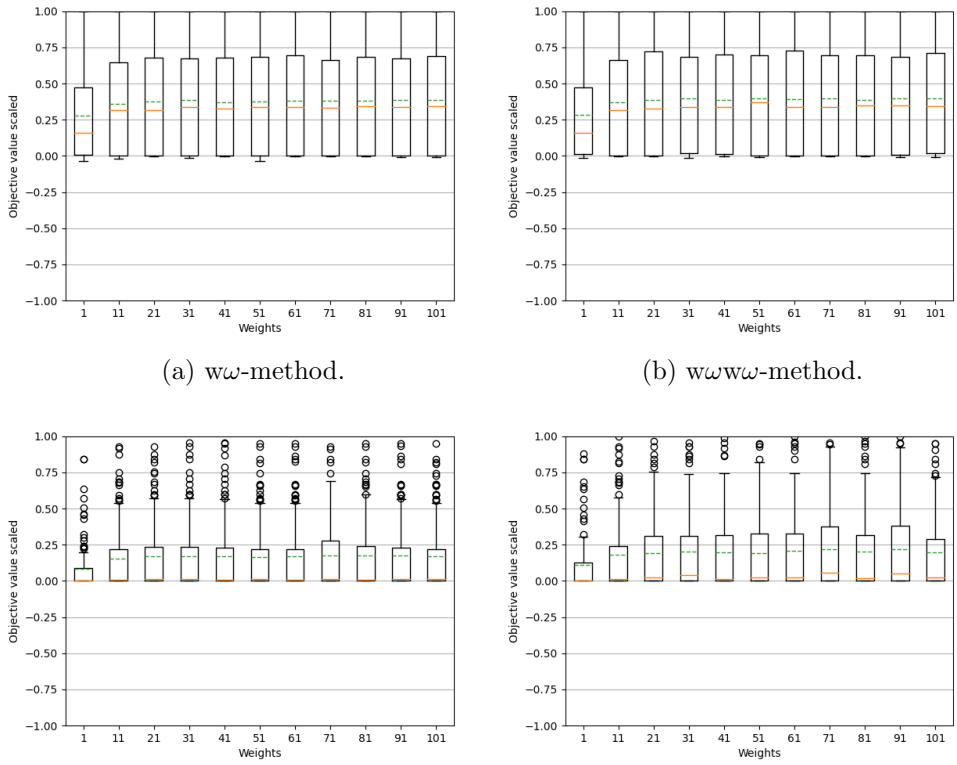
$$\begin{aligned} U &= \min\{U_1, U_2\} \\ &= \min\left\{n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2, n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1\right\} \end{aligned}$$

where R_1, R_2 are the sum of the ranks of group 1 and 2, respectively. The ranks of the data points X_i with $i \in \{1, \dots, n_1\}$ and Y_j with $j \in \{1, \dots, n_2\}$, are determined by combining both samples into one sample and assigning a numeric rank to each data point in the combined sample, starting from 1. If there are groups of tied values, then a rank equal to the midpoint of the unadjusted rankings are assigned to those values. For example, consider the samples $X = (3, 5, 8)$ and $Y = (5, 5)$. Then, the combined sample is equal to $(3, 5, 5, 5, 5, 8)$ and its ranks are $(1, 3.5, 3.5, 3.5, 3.5, 6)$, where the unadjusted ranks would be $(1, 2, 3, 4, 5, 6)$. To determine the values for R_1 and R_2, we then sum the ranks assigned to data points X_i with $i \in 1, ..., n_1$ and Y_j with $j \in 1, ..., n_2$, respectively. Hence, for the example, $R_1 = 1 + 3.5 + 6 = 10.5$ and $R_2 = 3.5 + 3.5 = 7$.
To test the results in this thesis, the Scipy.stats library of Python is used to calculated the U statistic and the corresponding p-value.

4.5 Results and discussion

In this section, the results of the performed experiments as explained in Section 4.4, are shown.

4.5.1 SquaredLoad



(a) wω-method.

(b) wωwω-method.

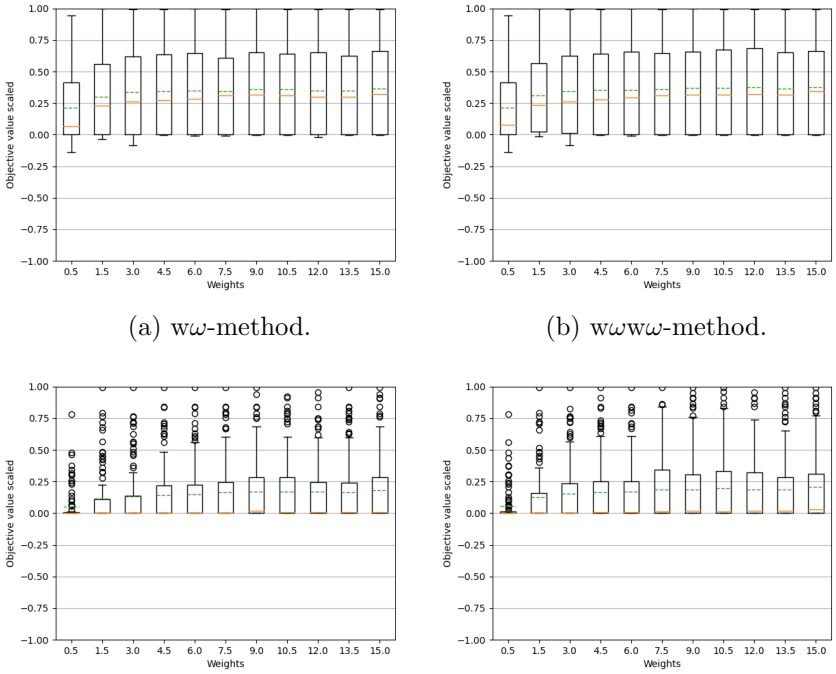
(c) ωwω-method.

(d) ωwωwω-method.

Figure 4.5: A boxplot of the objective value of the method with the SquaredLoad potential KPI per weight based on 100 instances of varying size with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The results of the SquaredLoad potential KPI with a random initial solution and general weights can be seen in Figure 4.5. In the figures, the objective value found with different methods are shown per weight in comparison with the optimum (a value of 1), the solution found with the basic local search (a value of 0) and the initial solution (a value of -1). Looking at the boxplots, it can be seen that all four methods perform almost always better than the basic local search algorithm, especially for weights higher than 1.
The wω-method seems to perform a lot better than the basic local search method. The median and mean are namely above 0.25 and the upper bound is even above 0.5 for all weights higher than 1. Moreover, the solution found with the wω-method, is sometimes equal to the optimal solution. However, the method also occasionally performs worse than the method without potential KPI as negative values occur. These negative values are close to zero, hence when the wω-method performs worse than the basic local search, then it only performs a little bit worse. This is probably the case because the initial solution is random. A random solution is often not a very good solution and the local search improves the solution already a lot. Also note that the scale in the boxplots is different

from -1 to 0, than from 0 to 1, which makes it difficult to determine how bad the solutions, worse than the ω-method, are in comparison with the 'good' solutions. Thus, the wω-method performs well in comparison with the basic local search algorithm with a random initial solution.
The ωwω-method, however, performs less than the wω-method. The ωwω-method still performs better than the basic local search method. The mean and upper bound are just below 0.25 and the median is just above zero. Also, the optimum is never reached. Note that since the method starts with a phase without potential KPI, the method never performs worse than the basic local search. Thus, the solution found with this method is closer to the optimum than the solution found with local search was, but the difference is small.
The SquaredLoad potential KPI is also tested when using two extra phases, wωwω and ωwωwω, see Figures 4.5b and 4.5d. It can be seen that the extra phases do improve the algorithm but only slightly.



(a) wω-method.

(b) wωwω-method.

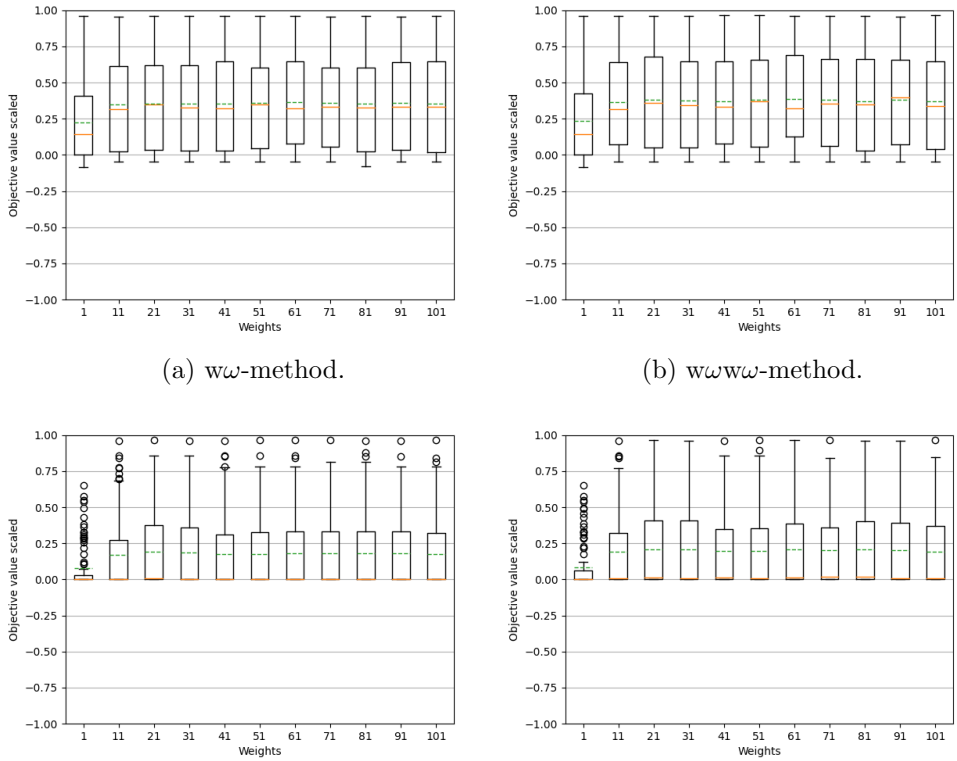
(c) ωwω-method.

(d) ωwωwω-method.

Figure 4.6: A boxplot of the objective value of the method with the SquaredLoad potential KPI per weight based on 100 random generated instances of varying sizes with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

In Section 4.4, a lower bound for the weight ν is estimated. The lower bound is equal to 15, i.e. for weights higher than 15, the performance of the methods with potential KPI should be the best and not improve anymore for higher weights. In Figure 4.5, it can be seen that indeed for weights higher than 15, the results are similar for all weights. Hence, it is also possible to use any weight higher than the weight bound. However, it is unknown if there is an upper bound and if there is

one, it is possible that an arbitrary high weight is higher than that upper bound and the results of the method with potential KPI are worse than it would be with a smaller weight. On the other hand, a weight of 11, lower than the bound of 15, seems to work just as good as the higher weights. This is expected, since the found bound is a very rough maximum. In practice, the difference in potential KPI between the current and potential solution is probably higher. A weight of 1 gives worse results, so the real bound of the weight is probably between 1 and 11.
In Figure 4.6, the results for values between 1 and 15 are plotted to investigate what the real bound might be. The results seems to be better for higher weights until weight 4.5, then, for weights higher than 4.5, the results are similar for all weights. Thus, the actual bound is probably around 4.5.



(a) wω-method.

(b) wωwω-method.

(c) ωwω-method.

(d) ωwωwω-method.

Figure 4.7: A boxplot of the objective value of the method with the SquaredLoad potential KPI per weight based on 100 instances of varying size for a greedy initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The results of the algorithm with a greedy initial solution are shown in Figure 4.7. In the figures, it can be seen that the methods perform similar as the method with a random initial solution. Indeed, for the algorithm with greedy initial solution, all four methods perform almost always better than the basic local search algorithm, especially for weights higher than 1.
The wω-method seems to perform better than the basic local search method. The median and mean are namely around 0.35 and the upper bound is even above 0.5 for all weights higher than 1. However, the solution found with the wω-method, is never equal to the optimal solution but

performs occasionally worse than the method without potential KPI as negative values occur. These negative values are more visible than they were for the results with the random initial solution, since the greedy solution is expected to be better than the random solution. However, it is hard to compare both, since the scale in the boxplots is different for both initial solutions. In conclusion, the wω-method performs well in comparison with the basic local search algorithm with a greedy initial solution.
The ωwω-method also performs better than the basic local search method, but worse than the wω-method. The mean and upper bound for the ωwω-method are around 0.25, the median is just above zero and the optimal value is never reached. Note that the method never performs worse than the basic local search because the method starts with a phase without potential KPI. Thus, the solution found with the ωwω-method and a greedy initial solution is closer to the optimum than the solution found without potential KPI was, but the difference is small.
The SquaredLoad potential KPI with greedy initial solution is also tested when using two extra phases, wωwω and ωwωwω, see Figures 4.7b and 4.7d. It can be seen that the extra phases do improve the algorithm but only slightly.

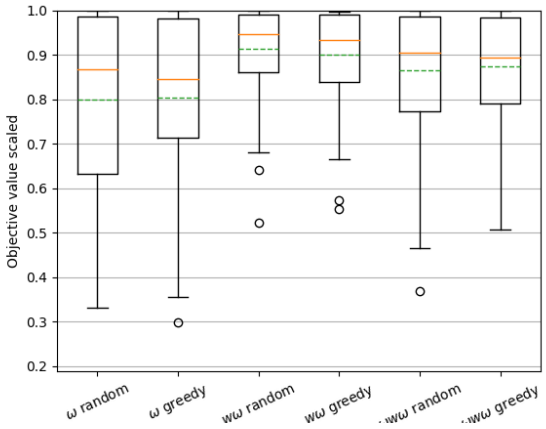
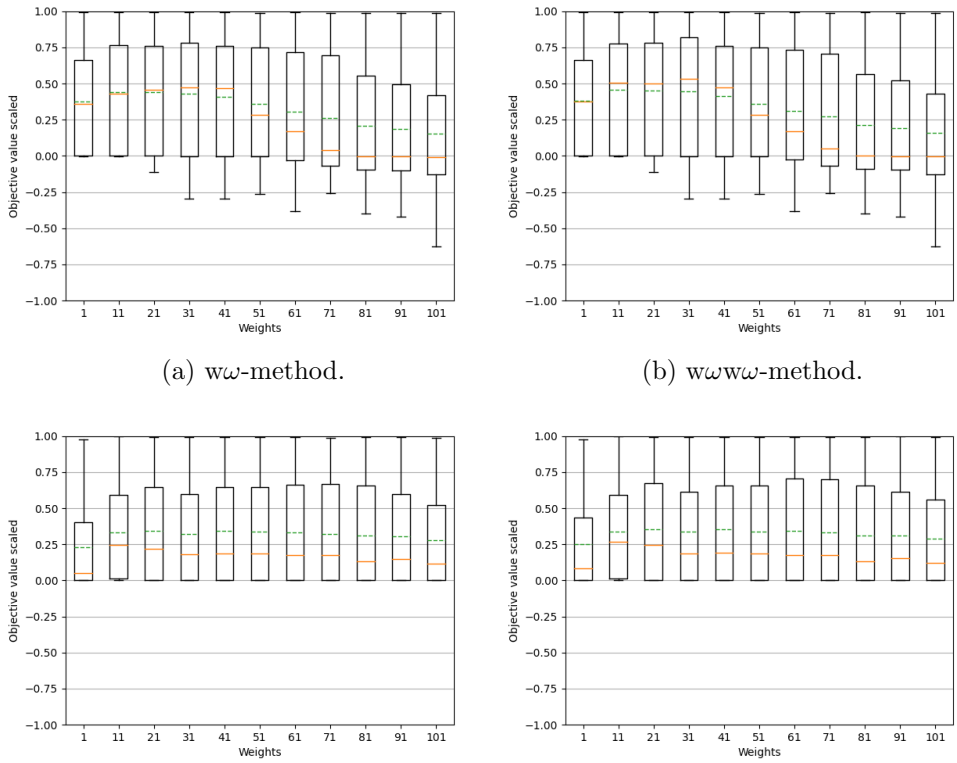


Figure 4.8: The results of the SquaredLoad function with a random and greedy initial solution where 0 represents the objective value of the random initial solution and 1 represents the optimal objective value. The figure is cut off by the average scaled objective value of the greedy initial solution.

In Figure 4.8, the results of the method with a random initial solution and the results of the method with a greedy initial solution can be compared. In the figure, zero namely represents the objective value of the random initial solution and one represents the optimal objective value. The figure displays those results with y-axes starting at the average scaled objective value of the greedy initial solution, which is just below 0.2. Hence, the greedy solution is on average a bit better than the random solution. Moreover, it has been verified that for 75 of the 100 instances, the greedy solution is better than the random solution. Thus, the greedy solution is not a lot better than the random solution. This observation can be explained by considering how the random solution is constructed: for every task, a start time is selected uniformly at random, leading to a distribution of tasks across different buckets, which prevents excessive overload. Furthermore, the greedy solution has some overload as well. Hence, the random solution is already a good initial solution for this problem and the greedy is not the best solution.
The performance of the methods with a greedy solution as initial solution is similar to the per-

formance of the methods with a random initial solution, as can be seen in Figure 4.8. For the wω-method, it even performs worse. This similarity is expected since the random solution is almost as good as the greedy solution.

4.5.2 Random function



(a) wω-method.

(b) wωwω-method.

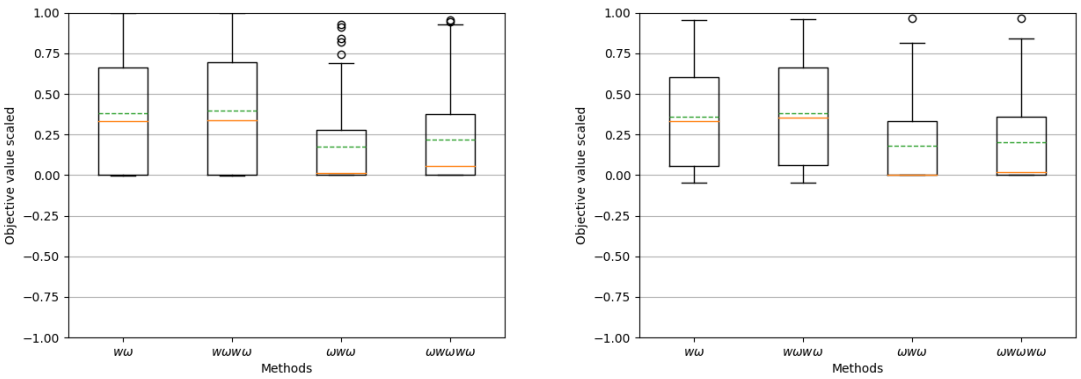
(c) ωwω-method.

(d) ωwωwω-method.

Figure 4.9: A boxplot of the objective value of the method with a random potential KPI per weight based on 100 instances of varying size with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The methods are also tested with a random function as potential KPI (see Section 4.4). The results of this experiment can be found in Figure 4.9. From the figure, it is clear that all four methods perform quite good and often better than the basic local search method. However, the results seems highly dependent on the value of weight, for higher weights the results are less good. For the wω-method, it seems that the results are less convincing for weights higher than 41 and for weights higher than 81, the median is even around zero, i.e. then the results of the method are just as good as the basic local search. For the ωwω-method, the results are already less good for weights higher than 21, although the scaled objective value stays clearly above zero. Hence, it can be useful to add some random factor to the objective function, but when that factor is too high, it is not useful anymore.

4.5.3 Comparing methods and functions



(a) Random initial solution

(b) Greedy initial solution

Figure 4.10: A boxplot of the objective value for different methods with the SquaredLoad potential KPI and a weight of 15, which is equal to the estimated bound. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

In Figures 4.10 and 4.11, additional results of the various methods are presented. From these figures, it is clear that all tested methods perform better than the basic local search algorithm in terms of objective value. However, the methods perform worse in terms of runtime. This increase is expected, since computing the objective value has become lengthier due to the SquaredLoad function. Unlike the original linear objective function, the SquaredLoad function is non-linear, thus requiring more time to compute. Moreover, for the ωwω, wωwω and ωwωwω-methods, the algorithm requires more iterations to converge than the ω-method. Despite the slight increase in runtime, utilising a potential KPI allows for a better solution attained with local search.
Comparing the wω-method and ωwω-method, it becomes evident that the wω-method outperforms the ωwω-method not only in terms of the objective value, as observed earlier, but also in terms of runtime. For a weight of 15, which equals the weight bound, the median of the boxplot of the wω-method is approximately 0.35 while for the ωwω-method, the median just exceeds 0. Furthermore, the number of iterations of the wω-method are on average around 80 iterations in comparison with approximately 110 iteration on average for the ωwω-method. Also, the runtime is a bit shorter for the wω-method than for the ωwω-method, although this difference is smaller since most iterations occur while running the method without potential KPI. Thus, the wω-method performs better than the ωwω-method for the bucketised planning problem for both a random initial solution and a greedy initial solution.
It has also been investigated whether adding two additional phases to the methods, wωwω and ωwωwω, will further enhance the performance of the algorithm. In Figure 4.10, it can be seen that adding two phases does yield a slight improvement in the objective value. However, both the runtime and number of iterations of the methods increase (see Figure 4.11). Considering the marginal improvement in the objective value, it is likely not worthwhile to include these extra phases, particularly given that the additional runtime can become quite substantial for larger instances due to the non-linearity of the SquaredLoad function.

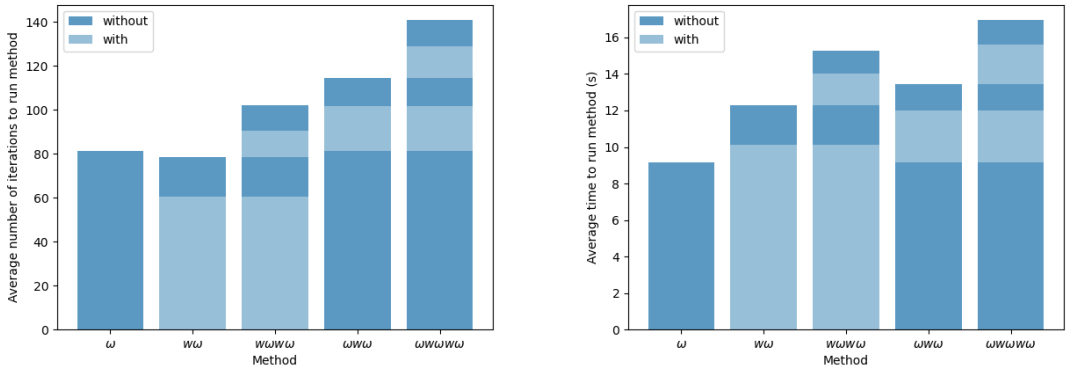


Figure 4.11: Histograms with the average runtime and number of iterations per method when the SquaredLoad potential KPI and a weight of 15 is used.

Statistical tests

In this section, the results of the statistical tests described in Section 4.4 are shown and discussed. In Tables 4.1 and 4.2, the different methods that are compared and their p-value and test statistic U are depicted. In table 4.1, the results of the methods with a random initial solution are shown and in Table 4.2, the results of the methods with a greedy initial solutions are shown. Note that the null hypothesis is that the objective values of both methods are similar in comparison with the optimal objective value and that this hypothesis is rejected when a p-value smaller than $\alpha = 0.05$ is found.

Table 4.1: Table with all the p-values and test statistics U of the statistical tests where method X is compared with method Y when using a random initial solution.

Method X	Method Y	U	p-value	Null hypothesis
SquaredLoad wω $\nu = 21$	No potential KPI ω	6419.0	0.000528	rejected
SquaredLoad ωwω $\nu = 21$	No potential KPI ω	5755.0	0.0652	
SquaredLoad wωwω $\nu = 21$	SquaredLoad wω $\nu = 21$	5104.5	0.799	
SquaredLoad ωwωwω $\nu = 21$	SquaredLoad ωwω $\nu = 21$	5139.0	0.735	
Random wω $\nu = 21$	No potential KPI ω	6383.0	0.000730	rejected
Random ωwω $\nu = 21$	No potential KPI ω	5965.0	0.0184	rejected
Random wω $\nu = 21$	SquaredLoad wω $\nu = 21$	5048.5	0.907	
Random ωwω $\nu = 21$	SquaredLoad ωwω $\nu = 21$	4729.0	0.509	

Multiple hypothesis tests have been performed to compare different methods. Firstly, the different methods and functions with their best weight are compared with the basic local search, i.e. no potential KPI. In table 4.1, it can be seen that for three of these comparisons, the null hypothesis is rejected. The first comparison is when comparing the wω-method with the SquaredLoad potential KPI and a weight of 21 with the basic local search method. Also, both potential KPI methods with the random function is significantly different than the basic local search method. If we compare this with the figures of previous sections, it can be concluded that these methods are significantly better than the basic local search. Hence, only the ωwω-method with the SquaredLoad potential KPI and a random initial solution is not significantly better than the basic local search method.
Secondly, it has been tested whether adding an extra phase with and without potential KPI is significantly better than not using the extra phases, i.e. the wω-method is compared with the ωwω-method and the ωwω-method with the ωwωwω-method. In table 4.1, it can be seen the differences between the methods are not significantly. This was expected based on the boxplots discussed earlier.
Finally, the random function is compared with the SquaredLoad function for both methods. Also, the differences between these methods are not significant as the p-value is higher than 0.05.

Table 4.2: Table with all the p-values and test statistics U of the statistical tests where method X is compared with method Y when using a greedy initial solution.

Method X	Method Y	U	p-value	Null hypothesis
SquaredLoad wω $\nu = 21$	No potential KPI ω	6274.5	0.00185	rejected
SquaredLoad ωwω $\nu = 21$	No potential KPI ω	5871.5	0.0333	rejected
SquaredLoad wωwω $\nu = 21$	SquaredLoad wω $\nu = 21$	5220.5	0.591	
SquaredLoad ωwωwω $\nu = 21$	SquaredLoad ωwω $\nu = 21$	5127.0	0.757	

In Table 4.2, more results of the hypothesis tests are shown. In this table, the methods using a greedy initial solution are compared. Note that the random function is not depicted in the table since the random function is not tested with a greedy initial solution. Hence, the two methods, wω and ωwω, are compared with the basic local search and it is tested whether running an extra phase significantly improves the results.
There are two p-values that are below $\alpha = 0.05$, namely when comparing the wω-method with the SquaredLoad potential KPI with the basic local search algorithm and when comparing the ωwω-method with the SquaredLoad potential KPI also with the basic local search method. Thus, both potential KPI methods perform significantly better than the basic local search method when using the SquaredLoad function and a greedy initial solution. Moreover, running an extra phase with the SquaredLoad potential KPI does not significantly improve the objective value.

4.6 Conclusion

From the results found in the previous section, some conclusions for the bucketised planning problem can be made. The experiments suggest that using SquaredLoad as a potential KPI in a simple local search algorithm, improves the algorithm and helps finding a solution with a better objective value. However, a random function also already performs significantly better than the basic local search algorithm when using a good value of weight. Thus, both the SquaredLoad function and a random function can be useful as potential KPI functions.
In general, the objective values found by the methods with the SquaredLoad as potential KPI are often closer to the optimal value than the method without potential KPI. Sometimes, the methods with potential KPI even finds a solution equal to the optimal solution. Moreover, the wω-method performs significantly better than the basic local search method according to the Mann-Whitney U test. However, the wω-method performs sometimes also worse than the ω-method, but then the difference between the two methods is so small that it is barely noticeable in the figures (see for example Figure 4.10). Hence, using SquaredLoad as potential KPI practically always improves the basic local search algorithm.
The runtime however, does increase due to the additional iterations with potential KPI. This increase is mainly because SquaredLoad is a non-linear function and therefore has an even higher

impact on large instances. However, the improvement in objective value is worth the extra time in most cases.
There are multiple parameters that can be altered and that influence the performance of the algorithm with a potential KPI. The impact of some of these parameters are tested by using and comparing different values of the parameters with the experiments described in Section 4.4.
First, the effect of the value of weight ν on the performance of the methods with SquaredLoad is tested. In Section 4.4, an estimate for a lower bound for this weight is determined. This bound is equal to 15 for the used instances, but in the experiments, the actual bound seems to be around 4.5 as weights lower than 4.5 give worse results and higher weights give similar results. However, it does not really matter what the exact value of the bound for the weight is. It is known that a weight higher than the calculated estimate would definitely give good results, and from the experiments, it looks like there is no upper bound, or at least not one that is close to the lower bound. Thus, the value of the calculated bound can be used as weight for the bucketised planning problem to obtain good results. However, for the random function, it seems like there is an upper bound for the value of weight. The results are clearly worse for higher weights.
Second, the influence of the initial solution was tested. Two different initial solutions are used for the SquaredLoad function, a random solution and a greedy solution. The algorithm with potential KPI performs similar for both initial solutions as can be seen in Figure 4.8. This is expected since the greedy solution is only slightly better than the random solution.
Lastly, the various methods to employ the potential KPI are compared in terms of objective value and runtime. The two methods tested, are the wω-method, starting with potential KPI, and the ωwω-method, starting without potential KPI. The experimental results suggests that the wω-method is the better method to use for the bucketised planning problem with the SquaredLoad function as potential KPI. The median and upper quartile for the wω-method are namely a lot closer to one than for ωwω, as can be seen in Figure 4.10. The runtime for the wω-method is also a bit lower. Moreover, the difference with the basic local search algorithm is significant for the wω-method while this is not the case for the ωwω-method. It is also tested whether adding two phases, i.e. adding wω, gives even better results. It turns out that the results of the algorithm with extra phases are a bit better than the results without those phases (see Figure 4.10), although the difference is not significantly according to the Mann-Whitney U test. However, the runtime also increases a lot, see Figure 4.11. Hence, it is not worthwhile to run the algorithm an extra phase with potential KPI, since the improvement in objective value is too small in comparison with the extra runtime.
In conclusion, it is almost always beneficial to use a potential KPI in a simple local search algorithm for the bucketised planning problem. The runtime does increase a bit, but the objective value relative to the optimum is better compared to the objective value found without potential KPI. Choosing the right values for the parameters can improve the algorithm even more. The effect of the weight is not very large, but a weight higher than the calculated bound gives the best results. This bound is 15 for the SquaredLoad function. The best method to employ the SquaredLoad potential KPI is to first run a local search algorithm with potential KPI, and then run the same algorithm without potential KPI with as initial solution the solution found with the potential KPI. Hence, according to the experimental results, the wω-method with the SquaredLoad function and a weight of 15, is the best improvement to the local search algorithm for the considered instances of the bucketised planning problem.

Chapter 5

Travelling Salesman Problem

The travelling salesman problem (TSP) is a commonly known optimisation problem. In this problem, a salesman wants to travel to all cities exactly once, starting and ending in the same city, and he wants to do this as fast as possible, so with the least distance. More formally, the problem is as follows: given a set of points with a certain distance away from each other, find a route of minimum length that visits each point exactly once. There are symmetric and asymmetric instances for this problem. For a symmetric instance, the costs from city 1 to city 2 are the same as the costs from city 2 to city 1, while for an asymmetric instance this is not the case. In Figure 5.1, an example of a symmetric instance for the travelling salesman problem is shown. The blue line depicts a path that visits all nodes. Hence the path in blue is a solution for TSP. The costs for this path is equal to the costs of the edges in the path, i.e. $1 + 2 + 1 + 2 + 2 = 8$ However, the path is not optimal, since a better path exists where the costs of the path is only 5 $(1 + 1 + 1 + 1 + 1)$.

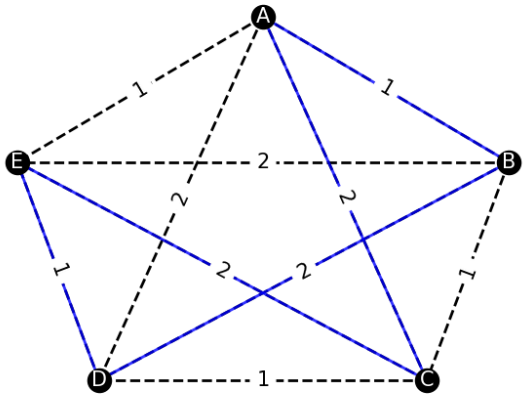


Figure 5.1: An example of a symmetric instance for the TSP with in blue a possible solution, i.e., a path that visits each node.

The travelling salesman problem is known to be NP-hard [41], but there are multiple heuristics known that can still find a good solution for this problem in reasonable time. The most common local search heuristics for TSP are 2-opt, i.e. swap two edges; and 3-opt, i.e. swap 3 edges. In this section, it is investigated whether a potential KPI can improve a local search heuristic for this problem.

5.1 Formal definition

The travelling salesman problem can be formulated as an integer linear program (ILP). This is useful to get more insight in the problem. Moreover, an ILP can be solved to optimality with an ILP solver like Gurobi. The ILP for the TSP is as follows:

$$\begin{array}{lll} \min & \sum_{i\in I}\sum_{j\in I} c_{ij}X_{ij} & \\ \text{s.t.} & & \\ & U_i - U_j + 1 \leq (|I|-1)(1-X_{ij}) & \forall i,j \neq 1 \in I,\, i \neq j \quad (1) \\ & \sum_{i\in I, i\neq j} X_{ij} = 1 & \forall j \in I \quad (2) \\ & \sum_{j\in I, j\neq i} X_{ij} = 1 & \forall i \in I \quad (3) \\ 1 \leq & U_i \leq |I| & \forall i \neq 1 \in I \quad (4) \\ & X_{ij} \in \{0,1\} & \forall i,j \in I \quad (5) \end{array}$$

In this ILP formulation, the **input** consists of a number of cities/nodes given by the set I with

- Costs (distance) $c_{ij} > 0$ it takes to go from city $i \in I$ to city $j \in I$

A well known example of a set of nodes, are points in the Euclidean space where the costs are the Euclidean distances between the points.
There are multiple **variables** in the formulation needed to define the problem. One variable determines which cities are connected in the path,

$$X_{ij} = \begin{cases} 1 & \text{path goes directly from city } i \in I \text{ to city } j \in I \\ 0 & \text{otherwise.} \end{cases}$$

Another variable for this problem is a dummy variable U_i that is needed to keep track of the order in which the cities in the path are visited. Variable U_i is equal to the position node $i \in I$ has in the path, i.e. equal to the number of cities that has been visited before city $i \in I$ in the path plus one, where one city $s \in I$ is chosen as starting point ($U_s = 1$). So, if city $i \in I$ is earlier in the path than city $j \in I$, then $U_i < U_j$. To preserve the correct order, we have $U_j = U_i + 1$ if $X_{ij} = 1$. This requirement can be formulated in a **constraint** as (1)
In the TSP, a path is made that visits every city $i \in I$ exactly once. To ensure this, there are two **constraints**, namely constraints (2) and (3).
Then, there are also some constraints that are needed to bound the problem. Those constraints are constraints (4) and (5).
Note that the objective of the travelling salesman problem is to minimise the costs of the total path. Hence, the objective function is as formulated in the ILP.

5.2 Local search algorithm

For the travelling salesman problem, a simple local search algorithm is used. The algorithm starts from an initial solution and continues as follows:

1. Look at the objective value of all neighbours created with 2-opt.
2. Select the neighbour with the best (lowest) objective value. When multiple neighbours have the same best objective value, one of these neighbours is selected uniformly at random.

3. If this neighbour has a better or equal objective value as the current solution, repeat the steps with this neighbour.

4. When no improvement is found for a number of iterations equal to the number of nodes, end the algorithm.

The algorithm ends when no improvement is detected for a number of iterations equal to the number of nodes because this number of iterations gives the possibility to search the complete plateau for better solutions.
To create the neighbours of a current solution, a 2-opt move is used. With this move, the neighbour is created as follows: from the current solution, remove two edges and connect the two distinct paths again such that one path is made and the path is different than before. Hence, given two points $i \in I$ and $j \in I$, the neighbours can be created by following the following three steps:

1. Take the same path from a start node to node i as the current solution.

2. Take the path between nodes j and $i + 1$ and add them in reverse order to the new path.

3. Take the same path in the same order between node $j + 1$ and the start node as the current solution to complete the path.

Note that $i + 1$ ($j + 1$) is the point directly after i (j) in the current path.
In this basic algorithm, the final solution is equal to the last solution that is found. This solution is always equal to the best solution that has been visited, since a neighbour is only explored if its objective value is better or equal. However, this does not have to be the case when using a potential KPI. It is possible to visit better solutions, but move to a neighbour with a worse objective value as the objective function plus potential KPI can be better. This gives three possible algorithms:

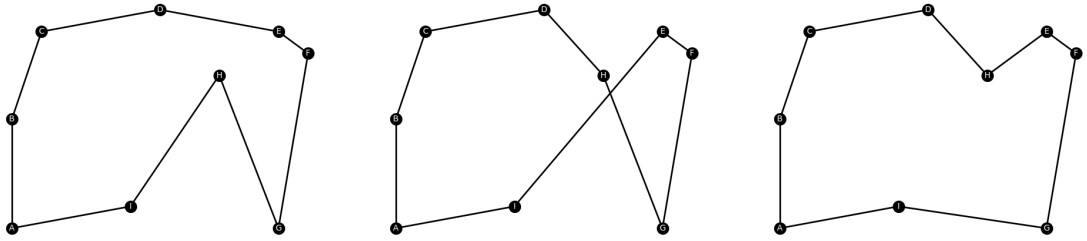
1. Continue every phase with the solution that was last found with the algorithm and use this solution as final solution.

2. Continue every phase with the solution that was last found with the algorithm, but use the best solution that has been explored as final solution.

3. Continue every phase with the best solution found so far and use this solution as final solution.

The phases are explained in more detail in Section 3.3. It is expected that the last variant of the algorithm does not work very well, since the goal of the potential KPI is to move away from the local minimum, the best solution so far, to hopefully find new better solutions. Moreover, the second variant is probably a bit better than the first since it takes the best solution which is always at least similar to the last solution. Hence, the second variant is used to test the performance of a potential KPI on the TSP.

5.3 Potential KPI



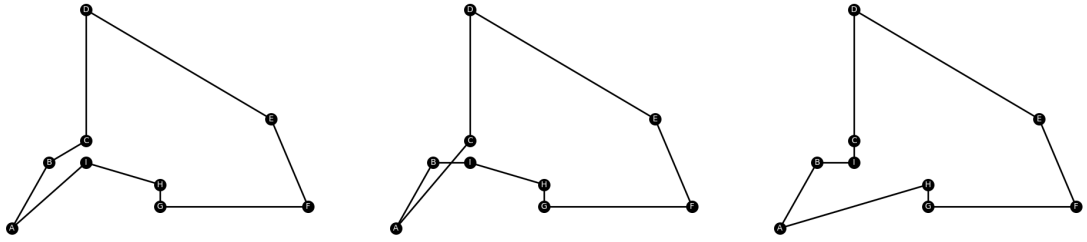
(a) Local minimum found with 2-opt.

(b) Potential solution.

(c) Global minimum and solution found with 3-opt.

Figure 5.2: Example of a symmetric TSP instance with Euclidean distances as costs where the 2-opt algorithm ends in a local minimum while the 3-opt algorithm does results in the global optimum.

To find a function that can help in escaping a local minimum, it is first investigated what a local minimum looks like relative to the global optimum. The local minimum might be different for different heuristics. For example, a local minimum for 2-opt, that is not a local minimum for 3-opt, is shown in Figure 5.2a where Figure 5.2c depicts the optimal solution found with 3-opt. Another example of such a local minimum for 2-opt is shown in Figure 5.3. For this instance, the global minimum is as shown in Figure 5.3c. Both local minima share a common characteristic: there exists a point that would be better positioned later in the path, since that results in a reduced distance. Hence, the distance between this point and one side of the path is smaller than that between the point and the side of the path it is currently connected to. For the example in Figure 5.2, the node, that is wrongly positioned, is node H, for the example in Figure 5.3, this node is node I. Since 2-opt only unfolds crossing roads and cannot resolve this issue, the algorithm inevitably ends in this local minimum. Thus, for the travelling salesman problem with 2-opt, potential solutions are solutions that are more likely to insert these points into the nearest part of the path in future iterations. An example of such potential solutions for the instances depicted in Figure 5.2a and 5.3a are shown in Figures 5.2b and 5.3b, respectively. A useful potential KPI function should be defined such that these potential solutions are rewarded. There are three possible functions that are considered as potential KPIs in this thesis.



(a) Local minimum of 2-opt.

(b) Potential solution.

(c) Global minimum.

Figure 5.3: Example of a symmetric TSP instance with the Euclidean distance as cost function. For this example, the 2-opt algorithm results in a local minimum instead of the optimum.

ClosestEdge

For the optimal path, nodes should be positioned in a segment of the path that is nearest, as illustrated in the examples in Figures 5.2 and 5.3. A path is therefore better, if for every node a part of the path is close to that node. Hence, the potential KPI should favour those solutions. This can be achieved by a function that examines the minimum costs between nodes and their closest edge in the current path. This function is called ClosestEdge (CloEd) and can be written as:

$$g(S) = \sum_{i \in I} \min_{\substack{k,l \neq i \in I; \\ X_{kl}(S)=1}} \frac{c_{ki} + c_{il}}{2} \tag{5.1}$$

where S is the current solution.
In Figure 5.1, another example of an instance for TSP is shown. For this example, the value of ClosestEdge function for one node is illustrated in Figure 5.4. Considering node A, there are three edge not adjacent to node A, namely edge BD, DE and EC. The distance or cost of node A to those edges can be calculated by the average of the costs of node A to the nodes of the edge. This distance is denoted by the lightblue dotted lines in the figure. Now, the value of the CloEd function for node A is equal to the minimum of these distances. In this example, all dotted lines have the same costs, so the CloEd value for node A is equal to those costs, which is $\frac{2+1}{2} = 1.5$. Note that this calculation can be done for all the nodes of the instance. The ClosestEdge function is equal to the sum of the minimal distances of all nodes.

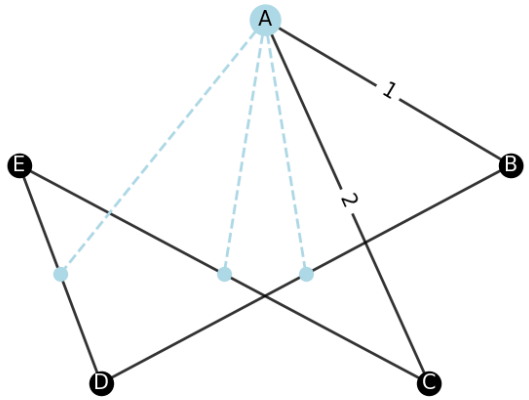


Figure 5.4: The ClosestEdge function for the instance and solution described in Figure 5.1. The black line depicts the path/solution. ClosestEdge is calculated for node A. Hence, the value of CloEd for this node is equal to the minimal costs of the lightblue dotted lines, i.e., the minimal distance from node A to the midpoint of a non-adjacent edge.

It has been verified that indeed, the potential solutions, as shown in Figures 5.2b and 5.3b, have lower values for the ClosestEdge potential KPI then the local minima shown in Figures 5.2a and 5.3a, respectively. Hence, the potential solutions are now accepted by the 2-opt local search algorithm.
However, there are also some disadvantages of the function. One disadvantage is that CloEd tends to stimulate a high number of crossings, as the crossing edge is often close to a node. However, after executing the algorithm with a potential KPI, a subsequent run of the algorithm without the potential KPI resolves these crossings to minimise the costs of the path. Additionally, a function

that permits numerous crossings steers the algorithm toward different solutions than it would do without the potential KPI. Consequently, this approach explores a wider array of solutions, increasing the likelihood of finding a good solution.
Another disadvantage of this function is its quadratic nature, which results in longer computation times for calculating the value of the potential KPI and may significantly slow down the entire algorithm.

ConnectingNodes vs ClosestNodes

Another possible potential KPI is a function which is called ConnectingNodes vs ClosestNodes (CoNoCloNo). This function compares the current costs between connecting nodes with the possible minimal costs between nodes, so this function can be written as:

$$g(S) = \sum_{i \in I} \frac{\sum_{j \in I, j \neq i} c_{ji} X_{ji}(S) + c_{ij} X_{ij}(S)}{\min_{k \neq l \neq i \in I} c_{ki} + c_{il}} \tag{5.2}$$

where S is the current solution.
This function examines the desired costs for a node and determines how high the current costs are in comparison with these desired costs. The desired costs represent the minimal costs achievable. Hence, the function computes the sum of the ratio of each node's current costs to its two minimal costs.
In Figure 5.5, an example instance is shown with the ConnectingNodes vs. ClosestNodes function for node A illustrated. The purple dotted lines are the desired distances for node A. Hence, nodes E and B are closest to node A. The lightblue lines are the distances to the connecting nodes in the current path. Now, to calculate CoNoCloNo for node A, the costs of the lightblue lines, 2 and 1, are divided by the costs of the purple lines, 1 and 1. Hence, CoNoCloNo for node A is equal to $\frac{2+1}{1+1} = \frac{3}{2}$.

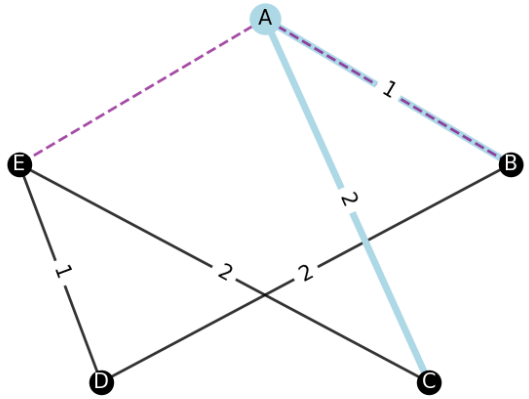


Figure 5.5: The ConnectingNodes vs. ClosestNodes function for the instance and solution described in Figure 5.1. The black line depicts the path/solution. CoNoCloNo is calculated for node A. Hence, the value of CoNoCloNo for this node is equal to the costs of the lightblue lines divided by the costs of the purple dotted lines.

Note that the goal of this function is to prioritise solutions where nodes are positioned close to their optimal locations, thereby guiding the algorithm toward the optimal solution. However, this goal

is almost similar to the original objective and therefore the function closely resembles the original objective function. Hence, using this function may not result in significantly different solutions compared to those produced by the algorithm without the potential KPI.
An advantage of this function is its linearity. Hence, calculating the objective value plus potential KPI in the local search algorithm is quite fast.

ConnectingNodes vs ClosestEdge

The third and final function considered as potential KPI for TSP is a combination between the two previously mentioned functions and is called ConnectingNodes vs ClosestEdge (CoNoCloEd). The function is defined as follows:

$$g(S) = \sum_{i \in I} \frac{\sum_{j \in I, j \neq i} c_{ji} X_{ji}(S) + c_{ij} X_{ij}(S)}{\min\limits_{\substack{k,l \in I; \\ X_{kl}(S)=1}} c_{ki} + c_{il}}. \tag{5.3}$$

In Figure 5.6, the function is illustrated for one node. Similar to the CoNoCloNo function, the connecting nodes in the current path are compared with the desired nodes. In the figure, the edges to the connecting nodes are depicted in lightblue and the desired edges are purple dotted lines. For this function, the desired nodes, are also the nodes closest by, but only two nodes that are adjacent in the current path are considered as possible desired nodes. Hence, for node A in Figure 5.6, nodes B and D, E and C and nodes D and E are considered as possible pairs of desired nodes since there is an edge between them in the current path. Thus, the ConnectingNodes vs. ClosestEdge for node A is equal to the costs of A to its connecting nodes, costs of AB (=1) and AC (=2), divided by the minimal costs of A to two nodes that are adjacent in the current path, costs of AC (=2) and AE (=1).

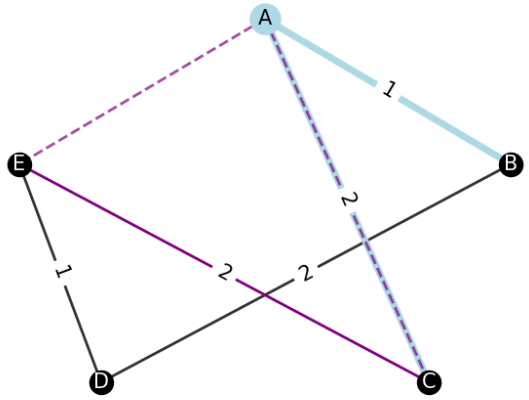


Figure 5.6: The ConnectingNodes vs. ClosestEdge function for the instance and solution described in Figure 5.1. The black line depicts the path/solution. CoNoCloEd is calculated for node A. Hence, the value of CoNoCloEd for this node is equal to the costs of the lightblue lines divided by the costs of the purple dotted lines.

The CoNoCloEd function evaluates how high the current distance of a node is to its neighbours in comparison with the minimum distance achievable after inserting the node into another part of the path. Hence, this function could be a promising combination of the two previously described functions as it considers both the desired costs and possible solutions based on the current path.

However, the function is still similar to the original objective function and utilising CoNoCloEd may not results in exploring different solutions than the algorithm did without potential KPI. Another disadvantage of this function is its non-linearity, which significantly increases the runtime of the algorithm.

These three potential KPI functions are designed to escape local minima with characteristics as described by the examples in Figures 5.2a and 5.3. It has been verified that these functions indeed enable the algorithm to escape the specific instances depicted in the examples. However, it is uncertain whether they will consistently escape local minima, since they are similar to the original objective function and/or do not take the optimum into account. Therefore, their performance is investigated later in this chapter. It is anticipated that there may be additional functions for TSP that could serve as potential KPI, but these have not been investigated.

5.4 Experimental set-up

In this section, the experiments to test the performance of the three potential KPI functions for the travelling salesman problem, constructed in Section 5.3, are explained. All the experiments are run on the DelftBlue supercomputer [33]. Furthermore, all the instances, used for the experiments, are symmetric TSP instances, i.e. the costs in the instances are independent of the direction of the route. Only symmetric instances are used since this simplifies the experiments and still gives enough insight in the performance of potential KPIs. Moreover, the second variant of the simple local search, as explained in Section 5.2, is used for all experiments. The other two variants are investigated as well, but give worse results and are not investigated further. The results of these variants can be found in Appendix B.
First, the potential KPIs are assessed using some simple instances similar to the examples used to construct the functions (see Figures 5.2 and 5.3). The results of these tests are shown in Appendix B. While these initial tests for the travelling salesman problem show promise, the differences in results between the methods are small. The objective value of the basic local search algorithm, so without potential KPI, is on average only 1.5% higher than the optimum objective value. Hence, the basic local search heuristic performs already quite well for the created instances, leading to only small improvements with potential KPI. To obtain more conclusive results, more elaborate tests are necessary where the difference between the optimum and the solution found with the 2-opt local search algorithm is more pronounced. Moreover, the data used to generate these initial tests consists of only small instances with minimal differences between the instances. Therefore, some more experiments were performed with more varied and elaborated data. This is done by using the benchmark dataset by Reinelt [42]. This dataset consists of 112 instances with varying number of nodes and varying costs between nodes. The costs encompass various types of distances, such as the Euclidean distance, the Manhattan distance, or explicit costs in the form of a matrix. For the experiments in this thesis, only the smallest 33 instances are used, ranging from 14 to 127 nodes. These instances are specifically chosen because the larger instances would require too much time to investigate the performance of potential KPIs.
To investigate the performance of a potential KPI for the travelling salesman problem, the three constructed functions, ClosestEdge (CloEd), ConnectingNodes vs ClosestNodes (CoNoCloNo) and ConnectingNodes vs ClosestEdge (CoNoCloEd), are tested with different parameters. The functions were individually tested in a similar way as the bucketised planning problem, see Section 4.4. Following these tests, the three functions are compared to each other. To compare the different functions and to see if the chosen functions are indeed good choices as potential KPI, also a random

linear function, $\sum_{i\in I}\sum_{j\in I} a_{ij}c_{ij}X_{ij}$ with a_{ij}, $i,j \in I$ equal to random integers between -5 and 5, is tested. The concept of a potential KPI is to steer the local search algorithm away from the local minima that are not yet globally minimal and towards the right direction in the search space, i.e. solutions closer to the global minimum. Using the random function, it can be investigated whether the selected potential KPI functions indeed steer the algorithm towards the intended direction, or if steering the algorithm away from local minima towards a random direction results in similar solutions.
Similar to SquaredLoad for the bucketised planning problem, the different functions are tested for different initial solutions, different ways of employing the potential KPI function and different values of weight. The two used initial solutions are a random solution and a greedy solution. The random solution is constructed by uniformly at random selecting the next node in the path until all nodes are in the path exactly once. The greedy solution is constructed by starting from the edge with minimum costs and then continuing the path with the closest node until all nodes have been visited. This approach to construct the greedy solution is used instead of more famous approaches like the heuristic by Christofides [43], because this approach is simple and can be performed fast. Moreover, the goal of testing the algorithm with a potential KPI and different initial solutions is to see what the effect of the initial solution is on the performance of the algorithm, especially in comparison with the algorithm without potential KPI. For this goal, a very good greedy solution is not needed and would only complicate the total algorithm. With the simple greedy initial solution as defined above, it is possible to investigate the effect of using a better initial solution in comparison with using a random initial solution. It is expected that the random initial solution is not a good solution and that therefore the basic local search without potential KPI improves the solution already a lot. The greedy initial solution is a better solution, and after the local search, the found objective value would probably be close to the optimum value.
For the travelling salesman problem, two different methods are tested, starting with potential KPI, wω and starting without potential KPI, ωwω (see Section 3.3). These methods are tested for each potential KPI function that is introduced in Section 5.3. Since running the algorithm an extra time with potential KPI did not improve the algorithm a lot for the bucketised planning problem, it is not investigated what the impact of an extra run with wω is for TSP.
For each of the methods and functions, the value of the weight ν is investigated. First, some general weights between 1 and 100 are used to investigated the performance of the potential KPIs. However, it is expected that the value of weight resulting in the best objective value, is dependent on the instance. In Section 3.3, a lower bound for an effective weight is found: $\nu \geq \frac{f(S_{\text{potential}})-f(S_{\text{current}})}{g(S_{\text{initial}})-g(S_{\text{potential}})}$. An estimate of the maximum of this bound is used as weight since the bound should hold for all possible potential solutions. For each of the constructed functions, an estimate of the bound is determined.

ClosestEdge

To estimate the maximum lower bound of the weight, it is assumed that the potential solutions resemble the one depicted in Figure 5.3b in Section 5.3. Note that in general, a potential solution is a neighbour of the current solution where the objective value $f(S) = \sum_{i\in I}\sum_{j\in I, j\neq i} c_{ij}X_{ij}$ of the potential solution is worse, but the value of the potential KPI, i.e. CloEd, for the potential solution is better than the current solution. So, $f(S_{\text{potential}}) - f(S_{\text{current}}) > 0$ and $g(S_{\text{potential}}) - g(S_{\text{current}}) < 0$. Furthermore, to create a potential solution, two edges are replaced with two different edges, where at least one edge is replaced by a longer edge, since the total costs of the path, f increases. First, the maximum of the numerator of the bound, $f(S_{\text{potential}}) - f(S_{\text{current}})$, is determined. This maximum can easily be deducted from the information of the potential solution. Two edges are

replaced with two new ones. To maximise, both edges should be replaced with longer edges and this can be at most equal to the difference between the longest edge and the shortest edge. So, the estimate of the maximum of the numerator in the weight bound is

$$2 \max f(S_{\text{potential}}) - f(S_{\text{current}}) = 2(c_{\max} - c_{\min})$$

where $c_{\max}$ is the highest costs and $c_{\min}$ the shortest costs possible.
The denominator of the maximum lower bound is the minimum of $g(S_{\text{current}}) - g(S_{\text{potential}})$ with $g(S) = \sum_{i \in I} \min_{\substack{k,l \neq i \in I; \\ X_{kl}(S)=1}} \frac{c_{ki}+c_{il}}{2}$. Hence, we have

$$\begin{aligned}
g(S_{\text{current}}) - g(S_{\text{potential}}) &= \sum_{i \in I} \min_{\substack{k,l \neq i \in I; \\ X_{kl}(S_{\text{current}})=1}} \frac{c_{ki}+c_{il}}{2} - \sum_{i \in I} \min_{\substack{k,l \neq i \in I; \\ X_{kl}(S_{\text{potential}})=1}} \frac{c_{ki}+c_{il}}{2} \\
&= \frac{1}{2} \sum_{i \in I} \min_{\substack{k,l \neq i \in I; \\ X_{kl}(S_{\text{current}})=1}} c_{ki} + c_{il} - \min_{\substack{k,l \neq i \in I; \\ X_{kl}(S_{\text{potential}})=1}} c_{ki} + c_{il} \\
&\geq \frac{1}{2} \min_{\substack{k,l \neq i^* \in I; \\ X_{kl}(S_{\text{current}})=1}} c_{ki^*} + c_{i^*l} - \min_{\substack{k,l \neq i^* \in I; \\ X_{kl}(S_{\text{potential}})=1}} c_{ki^*} + c_{i^*l} \\
&\geq \frac{1}{2}
\end{aligned}$$

where i^* is a node for which the minimal difference between the minimal costs of the current solution and the potential solution is higher than zero. In order words, for node i^*, the minimal costs to an edge in the solution is lower for the potential solution in comparison with the current solution. It is known that such a node exists since a potential solution has a better potential KPI than the current solution by definition. Note that one of the costs could stay the same and that only one cost is lower for the potential KPI. It has been verified that for all tested instances, the difference between the costs of that edge for such a node i^* is at least 1, since the edge costs are integer numbers.
Thus, the maximum lower bound for the weight for the ClosestEdge potential KPI is as follows:

$$\nu \geq 4(c_{\max} - c_{\min})$$

The performance of this potential KPI is now investigated with the value of the weight equal to a fraction of this bound. The chosen fractions range from 0.2 to 2. The fractions can help understand the behaviour of the weights versus the objective value. Moreover, the calculated estimate is a rough estimate and is therefore probably different than the 'real' bound. The fractions can help in investigating this difference and account for the gap in estimation.

ConnectingNodes vs ClosestNodes

The estimate for the maximum of the lower bound of weight ν for CoNoCloNo can be determined in a similar way as it was determined for CloEd. The estimate for the numerator $f(S_{\text{potential}}) - f(S_{\text{current}})$ is the same as it was for the ClosestEdge, namely $2(c_{\max} - c_{\min})$.
Now, an estimate for the minimum of the denominator is determined. We consider two cases:

1. $c_{ij}X_{ij}^{\text{current}} \leq c_{ji}X_{ji}^{\text{current}}$ and $c_{ij}X_{ij}^{\text{potential}} \geq c_{ji}X_{ji}^{\text{potential}}$ (or both inequalities interchanged).

2. $c_{ij}X_{ij}^{\text{current}} \geq c_{ji}X_{ji}^{\text{current}}$ and $c_{ij}X_{ij}^{\text{potential}} \geq c_{ji}X_{ji}^{\text{potential}}$ (or both inequalities interchanged).

Considering the first case, the denominator of the lower bound of the weight can be estimated as follows:

$$\begin{aligned} g(S_{\text{current}}) - g(S_{\text{potential}}) &= \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} c_{ji}X_{ji}^{\text{current}} + c_{ij}X_{ij}^{\text{current}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} - \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} c_{ji}X_{ji}^{\text{potential}} + c_{ij}X_{ij}^{\text{potential}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \\ &\geq \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} 2c_{ij}X_{ij}^{\text{current}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} - \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} 2c_{ij}X_{ij}^{\text{potential}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \\ &= \sum_{i\in I} \sum_{j\in I, j\neq i} \frac{2c_{ij}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \left(X_{ij}^{\text{current}} - X_{ij}^{\text{potential}}\right). \end{aligned}$$

To determine the weight bound, note that $X_{ij}^{\text{current}} - X_{ij}^{\text{potential}}$ is either -1, 0 or 1. Moreover, the difference in potential KPI value of the current and potential solution should be greater than zero by definition of a potential solution. So, to minimise the difference for case 1., only one node $i^* \in I$ and one node $j^* \neq i^* \in I$ for which $X_{ij}^{\text{current}} - X_{ij}^{\text{potential}} = 1$ can be considered. Hence,

$$\begin{aligned} g(S^{\text{current}}) - g(S^{\text{potential}}) &\geq \frac{2c_{i^*j^*}}{\min_{k\neq l\neq i\in I} c_{ki^*} + c_{i^*l}} \\ &\geq \frac{2c_{\min}}{2c_{\max}}. \end{aligned}$$

Now, we consider the second case. The denominator of the lower bound of the weight can be estimated for this case as follows:

$$\begin{aligned} g(S_{\text{current}}) - g(S_{\text{potential}}) &= \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} c_{ji}X_{ji}^{\text{current}} + c_{ij}X_{ij}^{\text{current}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} - \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} c_{ji}X_{ji}^{\text{potential}} + c_{ij}X_{ij}^{\text{potential}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \\ &\geq \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} 2c_{ji}X_{ji}^{\text{current}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} - \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} 2c_{ij}X_{ij}^{\text{potential}}}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \\ &= \sum_{i\in I} \sum_{j\in I, j\neq i} \frac{2}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \left(c_{ji}X_{ji}^{\text{current}} - c_{ij}X_{ij}^{\text{potential}}\right). \end{aligned}$$

Note that to minimise the difference for case 2., only one node $i^* \in I$ and one node $j^* \neq i^* \in I$ for which the difference is not zero can be considered. Moreover, for these nodes, the difference is minimal when $X_{j^*i^*}^{\text{current}} = 1$ and $X_{i^*j^*}^{\text{potential}} = 1$. Hence,

$$\begin{aligned} g(S^{\text{current}}) - g(S^{\text{potential}}) &\geq \frac{2}{\min_{k\neq l\neq i\in I} c_{ki} + c_{il}} \left(c_{j^*i^*} - c_{i^*j^*}\right) \\ &\geq \frac{2}{2c_{\max}}. \end{aligned}$$

It should be noted that the minimal difference between two costs, which do not yield the same value, is equal to one, since the edge costs of the considered instances are integer.
Thus, an estimate for the minimum of the denominator is the minimum of the estimates of both cases. Again, since the costs are integer, this minimum is equal to the estimate of case 2., $\frac{1}{c_{\max}}$.
Hence, an estimate for the maximum lower bound for the CoNoCloNo function for symmetric instances is given by

$$\nu \geq \frac{2(c_{\max} - c_{\min})}{\frac{1}{c_{\max}}} = 2c_{\max}(c_{\max} - c_{\min}).$$

ConnectingNodes vs ClosestEdge

As mentioned before, CoNoCloEd is a combination of the two other functions, i.e. $g(S) = \sum_{i\in I} \min_{\substack{k,l\in I;\\ X_{kl}(S)=1}} \frac{\sum_{j\in I, j\neq i} c_{ji}X_{ji}+c_{ij}X_{ij}}{c_{ki}+c_{il}}$. Similar as by the estimate for the weight bound for the other functions, it is assumed that the difference in potential KPI value between the current and potential solution is greater than zero due to the definition of a potential solution. So, to minimise the difference, only the minimum difference for one node $i^* \in I$ and one node $j^* \neq i^* \in I$ has to be determined. Note that for these nodes, $X^{\text{current}}_{i^*j^*} = 1$, $X^{\text{current}}_{j^*i^*} = 1$, $X^{\text{potential}}_{j^*i^*} = 1$ and $X^{\text{potential}}_{i^*j^*} = 1$. This gives the following difference.

$$
\begin{aligned}
g(S_{\text{current}}) - g(S_{\text{potential}}) &= \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} c_{ji}X^{\text{current}}_{ji} + c_{ij}X^{\text{current}}_{ij}}{\min\limits_{\substack{k,l\in I;\\ X^{\text{current}}_{kl}(S)=1}} c_{ki}+c_{il}} - \sum_{i\in I} \frac{\sum_{j\in I, j\neq i} c_{ji}X^{\text{potential}}_{ji} + c_{ij}X^{\text{potential}}_{ij}}{\min\limits_{\substack{k,l\in I;\\ X^{\text{potential}}_{kl}(S)=1}} c_{ki}+c_{il}} \\
&\geq \frac{c_{j^*i^*}+c_{i^*j^*}}{\min\limits_{\substack{k,l\in I;\\ X^{\text{current}}_{kl}(S)=1}} c_{ki^*}+c_{i^*l}} - \frac{c_{j^*i^*}+c_{i^*j^*}}{\min\limits_{\substack{k,l\in I;\\ X^{\text{potential}}_{kl}(S)=1}} c_{ki^*}+c_{i^*l}} \\
&\geq \frac{2c_{i^*j^*}}{c_{k^c i^*}+c_{i^*l^c}} - \frac{2c_{j^*i^*}}{c_{k^p i^*}+c_{i^*l^p}} \\
&\geq \frac{2c_{\min}}{c_{\max}+c_{\max}-1} - \frac{2c_{\min}}{c_{\max}+c_{\max}} \\
&= 2c_{\min}\left(\frac{1}{2c_{\max}-1} - \frac{1}{2c_{\max}}\right)
\end{aligned}
$$

where $c_{\max}$ ($c_{\min}$) the cost of the longest (shortest) edge, k^c, l^c the node that minimises the costs in the current solution for i^* and k^p, l^p the node that minimises the costs in the potential solution for i^*. Note that we assumed that $c_{j^*i^*} \geq c_{i^*j^*}$ and that this assumption is valid since $c_{j^*i^*} < c_{i^*j^*}$ would give the same estimate. Moreover, the difference is minimal when $c_{j^*i^*} = c_{i^*j^*} = c_{\min}$. Also note that the difference in potential KPI should be greater than zero, i.e., $c_{k^c i^*} + c_{i^*l^c} < c_{k^p i^*} + c_{i^*l^p}$. Moreover, $\min \frac{1}{c_{k^p i^*}+c_{i^*l^p}} = \frac{1}{2c_{\max}}$. Since the costs are discrete in the used instances, the minimal difference in costs is one and hence $\min c_{k^c i^*} + c_{i^*l^c} = 2c_{\max} - 1$. Thus, to minimise, we have $c_{i^*j^*} = c_{j^*i^*} = c_{\min}$, $c_{k^p i^*} = c_{i^*l^p} = c_{\max}$, and $c_{i^*l^c} = c_{\max}$ and $c_{k^c i^*} = c_{\max} - 1$ or $c_{i^*l^c} = c_{\max} - 1$ and $c_{k^c i^*} = c_{\max}$.
Thus, an estimate for the maximum of the lower bound for the weight of the CoNoCloEd potential KPI is as follows.

$$
\begin{aligned}
\nu \geq \frac{f(S_{\text{potential}}) - f(S_{\text{current}})}{g_{\text{current}}) - g(S_{\text{potential}})} &\approx \frac{2c_{\max} - c_{\min}}{2c_{\min}\left(\frac{1}{2c_{\max}-1} - \frac{1}{2c_{\max}}\right)} \\
&= \frac{(c_{\max} - c_{\min})(2c_{\max}-1)2c_{\max}}{c_{\min}(2c_{\max} - 2(c_{\max}-1))} \\
&= \frac{2(c_{\max} - c_{\min})(2c_{\max}-1)c_{\max}}{c_{\min}+2}.
\end{aligned}
$$

The performance of the CoNoCloEd function can now be investigated for a weight dependent on the instance. This is done by using different fractions of this bound to understand the impact of the weight on the objective value. The used fractions range from 0.2 to 2. However, this potential KPI function is similar to the original objective function and it is expected that this function does not perform a lot better than the local search with original objective function.

The results of the experiments are primarily presented using boxplots, with the mean represented in green and the median in orange. These boxplots follow a similar format to those utilised for the bucketised planning problem, as explained in Section 4.4. They range from -1 to 1, where a value of 1 indicates that the found objective value matches that of the optimal solution, a value of 0 means that the found objective value is equal to the objective value of the method without potential KPI and a value of -1 denotes equality with the objective value of the initial solution.
For the optimal solution, the best-known solution described in the benchmark data is used, as the optimal solution is not known for all instances. It should be noted that for some instances, this best-known solution is equal to the optimal solution. It is observed that, when using the random initial solution, there is one instance in the dataset where the basic local search algorithm finds the optimal solution. Similarly, using the greedy initial solution, there are three instances for which this is the case. These instances are still taken into account in the results.
In addition to the boxplots, histograms are created to compare the different methods and functions in terms of runtime and number of iterations. The histograms show the average number of iterations or average time per method, divided in the number of iterations/time with or without potential KPI.

Statistical tests

The results found with the experiments described above, are tested with a statistical hypothesis test. This test is similar as the one used for the BPP, see Section 4.4. For TSP, the null hypothesis is the same as for the BPP, namely:

$$H_0 : \frac{f(S_{\text{Method 1}}) - f(S_{\text{optimum}})}{f(S_{\text{start}}) - f(S_{\text{optimum}})} = \frac{f(S_{\text{Method 2}}) - f(S_{\text{optimum}})}{f(S_{\text{start}}) - f(S_{\text{optimum}})}$$

where f is the objective value, S_{Method} the solution found with 'Method'. Note that the scaled objective value, as defined in Section 4.4, is tested since the goal is to determine whether the solution found with one method is significantly closer to the optimum than another method. Hence, it is tested whether there is a significant difference in performance between two methods. This is mainly useful to see if adding a potential KPI is a good technique that works better than the basic local search method. However, also other methods can be compared, for example whether a well-designed function performs better as potential KPI than some random function. With a statistical test, a test statistic and a p-value, a probability that the hypothesis is rejected when it is true, can be determined. The null hypothesis can be rejected when the found p-value smaller is than α. For the performed tests, an α of 0.05 is chosen.
The data that is tested for the defined null hypothesis, are the results found as described above. The scaled objective value (see Section 4.4) is used as independent variable and the dependent variables are the costs between nodes. For this data, the same observations hold true as for the BPP:

- The dependent variable is ordinal. Hence, the costs between nodes have a set order.
- The independent variables are two independent, categorical groups, namely a group with objective values found with one method and a group with objective values found with the other method.
- The data points are independent, so the objective values found with method 1 of different instances do not dependent on each other or on the objective values found with method 2.

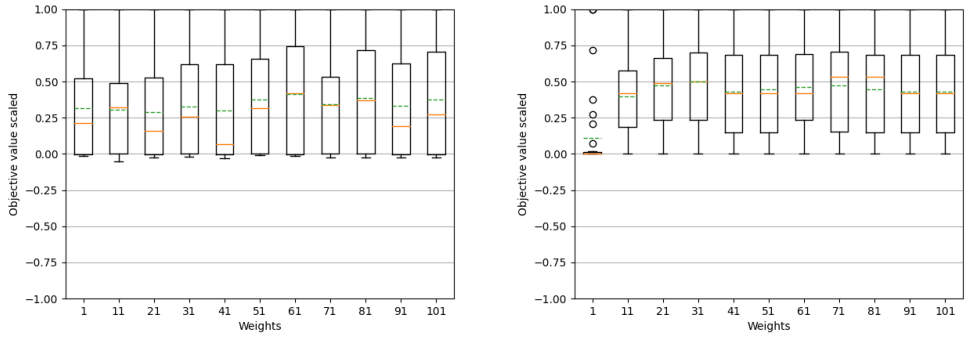
- The data points are not normally distributed. Although the exact distribution of the objective values is uncertain, it is expected that they are not normally distributed since the objective values are always integers, making a normal distribution highly unlikely. To check this assumption, the Kolmogorov-Smirnov test can be used to test for normality. However, this test can only determine if a distribution is likely to be normal, not whether it is not normal.
- The two groups of data points follow the same distribution since they are both found with a similar local search approach on the same input.

Since the same assumptions hold true as for the BPP, the same statistical hypothesis test can be used to compare the different methods and function. Hence, the used test is the Mann-Whitney U test [40], which is good in determining a significant difference between two different groups of data points. This test is explained in more detail in Section 4.4.

5.5 Results and discussion

The results of the performed experiments for the designed potential KPI functions (see Section 5.3) are shown in this section. First, the results of the experiments per potential KPI function are shown, then those results are compared. The set-up of these experiments are explained in Section 5.4.

5.5.1 ClosestEdge



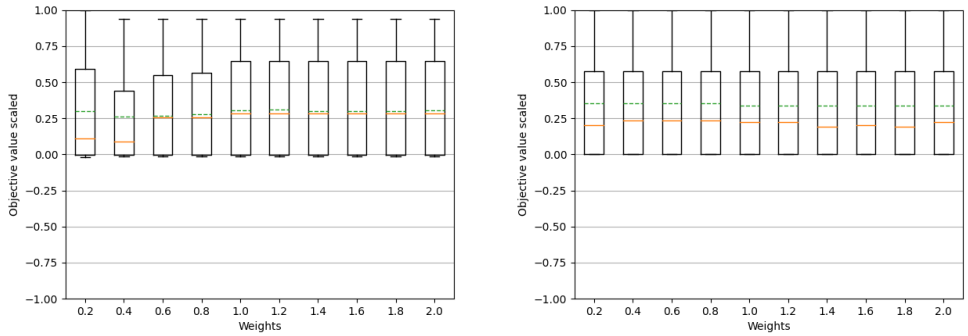
(a) wω-method

(b) ωwω-method

Figure 5.7: Boxplots of the objective value of the method per weight for the ClosestEdge potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The first potential KPI function investigated is the ClosestEdge function, which represents the sum of the distances from each node to its closest edge. In Figure 5.7, the results of the ClosestEdge function with a random initial solution and various general weights ranging from 1 to 101 are displayed. It is evident from the figures that both methods outperform the ω-method, as most of the boxes lie above 0.

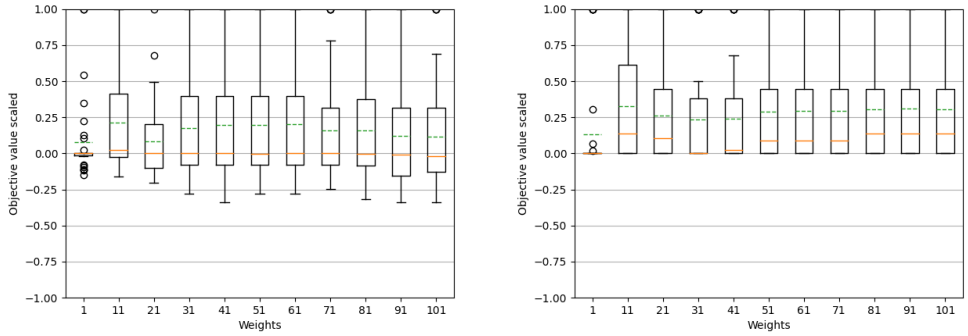
The ωwω-method performs very well for weights higher than 1. Especially the lower quartile is high, this quartile is namely just below 0.25. Moreover, the mean and median are around 0.5. Hence, the solution found with the ωwω-method is a lot closer to the optimum than the solution found without potential KPI.
The wω-method performs less well, as the median and mean are always below 0.5 and the lower quartile is around 0. However, the method still performs almost always better than the method without potential KPI. Some negative values, i.e. worse solutions, occur, but those are small, which means these solutions are only a bit worse than the solutions found with the ω-method.
The results of the methods differ significantly per weight, particularly for the wω-method. For instance, with a weight of 41, the median of the results of the wω-method is very low, whereas with a weight of 51, the median surpasses 0.25. Hence, there appears to be no clear relation between the weight value and objective value. These discrepancies in results can be explained by the variations in instances, such as differences in the number of nodes and type of costs. Therefore, an investigation is conducted to assess how the methods perform with a weight that is dependent on the instance. The weight determined in Section 5.4 is utilised for this purpose. The results of this experiment can be seen in Figure 5.8.



(a) wω-method (b) ωwω-method

Figure 5.8: Boxplots of the objective value of the method per fraction of the upper bound as value of weight for the ClosestEdge potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

With weights dependent on the instances, the results are more consistent. For the wω-method, the boxes are almost identical for weights higher than the lower bound. As for the ωwω-method, the boxes become similar for weights higher than approximately 40% of the lower weight bound. This discrepancy in fraction could stem from the estimate, which may differ from the actual bound.
The difference in fraction between the methods can be explained by the dependency on the potential solution. The bound is based on the (adapted) objective value of a potential solution such as the one shown in Figure 5.3b. This potential solution is based on the local minimum, hence the solution found with local search without potential KPI. In the ωwω-method, first the local search without potential KPI is run, and therefore the situation described in Section 5.3 is much more likely to occur. It might be that the bound of the weight is different for the wω-method since different types of potential solutions can occur.



(a) wω-method

(b) ωwω-method

Figure 5.9: Boxplots of the objective value of the method per weight for the ClosestEdge potential KPI function with a greedy initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The results of the algorithm starting with a greedy solution (see Section 5.4) are shown in Figure 5.9.
The wω-method appears to perform similarly as the ω-method, as the median is around zero for all weights. However, the box above the median, i.e. zero, is bigger than the box below. This indicates that the 'good' solutions, those better than the solutions found without potential KPI, are closer to the optimal solution compared to how close the 'bad' solutions, those worse than the ω-method, are to the initial solution. Therefore, the mean is above zero and it might still be beneficial to use the wω-method in comparison with using no potential KPI.
The ωwω-method performs slightly better. It is evident that the median for this method is notably above zero. It is important to note that this method cannot perform worse than the ω-method, as the best solution is considered the final solution, and the ωwω-method starts with the basic local search. While the improvement may not be as substantial as for the method with a random initial solution, the ωwω-method with a greedy initial solution does perform significantly better than the method without potential KPI.

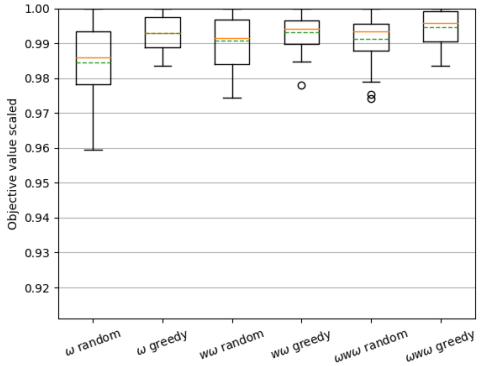
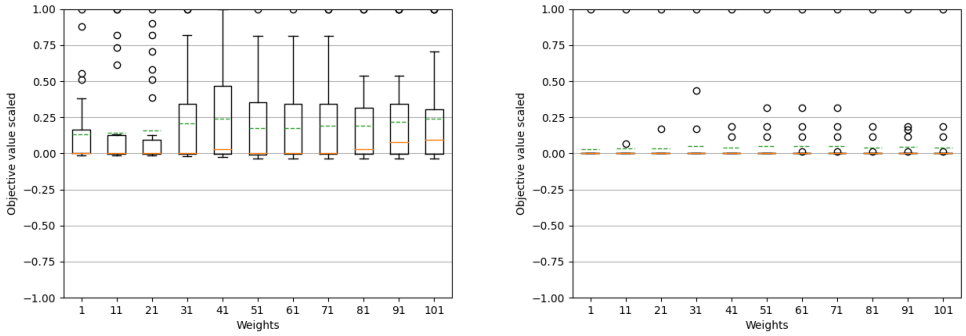


Figure 5.10: The results of the CloEd function with a random and greedy initial solution where 0 represents the objective value of the random initial solution and 1 represents the optimal objective value. The figure is cut off by the average scaled objective value of the greedy initial solution.

Figure 5.10 presents some boxplots that facilitate a better comparison of the methods, particularly highlighting the differences between a random initial solution and a greedy initial solution. The boxplot ranges from 0, which equals the objective value of the random initial solution, to 1, which is equal to the optimum objective value. Similar to the earlier figures, it is evident from this figure that the solutions found with the methods with potential KPI are, in general, closer to the optimum than those found with the methods without potential KPI. However, the greedy solution already has a good objective value compared to the random solution, since the greedy solution has an average scaled objective value of around 0.91. Furthermore, the local search algorithm without potential KPI also performs admirably well, especially when using a greedy initial solution. Therefore, only marginal improvement on the algorithm seems possible or necessary.

5.5.2 ConnectingNodes vs ClosestNodes

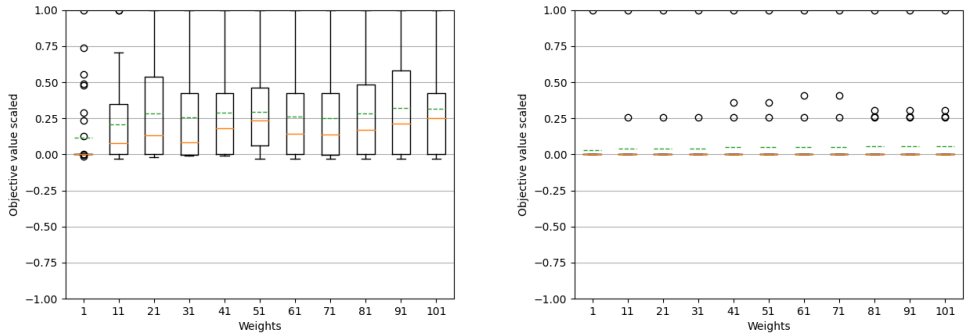


(a) wω-method (b) ωwω-method

Figure 5.11: Boxplots of the objective value of the method per weight for the ConnectingNodes vs ClosestNodes potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The results of the experiments described in Section 5.4 for the ConnectingNodes vs ClosestNodes potential KPI are shown in Figure 5.11.
The wω-method seems to perform better than the method without potential KPI, although the median is very close to zero. For weights 91 and 101, the median is slightly higher, but still not significantly more. Additionally, the optimum is only reached occasionally and the upper quartile is just above 0.25. Furthermore, for some instances, the solution found with the wω-method is slightly worse than the solution found without potential KPI. Hence, the method performs only slightly better than the ω-method.
The ωwω-method, however, performs even worse. Note that this method can never perform worse than the ω-method since the method starts with a phase without potential KPI, but, from the figure, it appears that the method also does not perform better except for some outliers.
In Section 5.3, it was already mentioned that the CoNoCloNo function is strikingly similar to the original objective function. Consequently, it is unsurprising that the methods employing CoNoCloNo as a potential KPI do not exhibit a notable improvement over the basic local search algorithm. By incorporating the potential KPI, the direction of the search remains largely unchanged, with similar solutions being explored. Consequently, and given the lack of promising results, further investigation into the method's performance with a instance-dependent weight is not pursued. Similarly, it is not tested how the method performs when the algorithm starts with a greedy solution instead of a random solution. The focus of the thesis is namely on other research questions.

5.5.3 ConnectingNodes vs ClosestEdge



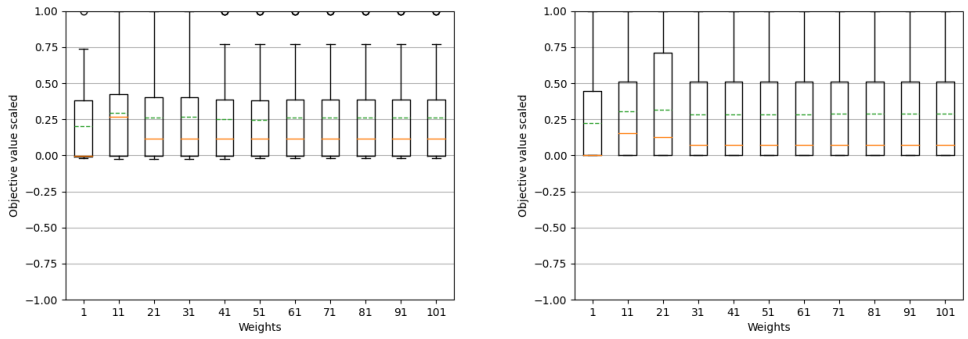
(a) wω-method (b) ωwω-method

Figure 5.12: Boxplots of the objective value of the method per weight for the ConnectingNodes vs ClosestEdge potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median

The last potential to consider is the ConnectingNodes vs ClosestEdge function. The results of the wω- and ωwω-methods for this function are shown in Figures 5.12a and 5.12b.
For this function, the wω-method also performs better than the ω-method for most instances and a weight higher than 1. The median is around 0.15 for most weights weights and the optimum is regularly reached. However, the boxes are practically always below 0.5 and the method also sometimes performs worse than the method without potential KPI. Thus, adding the CoNoCloEd

as potential KPI combined with the wω-method, does improve the basic local search algorithm. The ωwω-method hardly improves the basic local search even though the method cannot perform worse than the basic method. Hence, using the ωwω-method has almost no impact on the local search algorithm with 2-opt.
In Section 5.4, a lower bound for the weight for this function was established. However, given the unpromising results of this function and the similarities between this function and the original objective function, testing this lower bound is omitted. It is expected that the weight's influence is not significant enough to yield markedly improved results for an instance-dependent weight. Likewise, the methods utilising this function are not tested for a greedy initial solution.

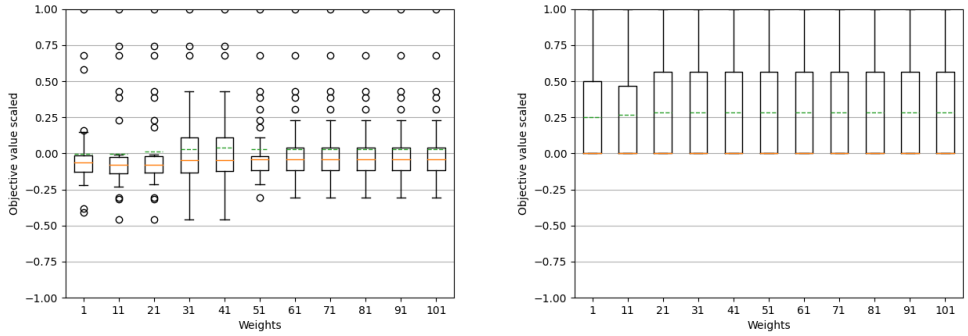
5.5.4 Random function



(a) wω-method (b) ωwω-method

Figure 5.13: Boxplots of the objective value of the method per weight for a random function as potential KPI with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

For the travelling salesman problem, also a random function is tested as potential KPI function as explained in Section 5.4. In Figures 5.13, 5.14 and 5.15, the results of these tests are shown.
In the results shown in Figure 5.13, a random initial solution is utilised. It appears that both methods outperform the method without potential KPI. The median for the wω-method is approximately 0.1, and the upper quartile exceeds 0.25. However, the optimum is rarely achieved, and some minor negative values are observed. Conversely, the ωwω-method does not exhibit any negative values and attains the optimal value more frequently. The median of the ωwω-method is slightly lower than that of the wω-method, just surpassing zero.



(a) wω-method (b) ωwω-method

Figure 5.14: Boxplots of the objective value of the method per weight for the random potential KPI function with a greedy initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

The random function is also tested for a greedy initial solution, see Figure 5.14. The wω-method with a greedy initial solution performs often worse than the basic local search method, as indicated by a median below zero. On the other hand, the ωwω-method with a greedy initial solution performs better, particularly notable is the relatively high upper quartile. Nevertheless, the median is very close to zero, suggesting that for many instances, the ωwω-method performs the same as for the method without potential KPI.

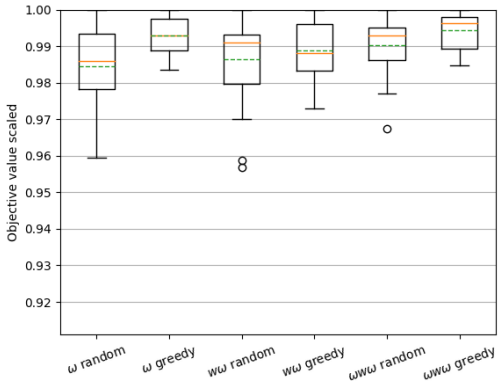


Figure 5.15: The results of the random function with a random and greedy initial solution where 0 represents the objective value of the random initial solution and 1 represents the optimal objective value. The figure is cut off by the average scaled objective value of the greedy initial solution.

In Figure 5.15, the results of the various methods and initial solutions are displayed in one figure for easier comparison. It is evident from the figure that the basic local search method already performs remarkably well, with a median exceeding 0.99. Therefore, it is logical to conclude that adding a potential KPI does not significantly improve the local search algorithm. From the figure, it also appears that the ωwω-method with this random function works slightly better than the wω-method.

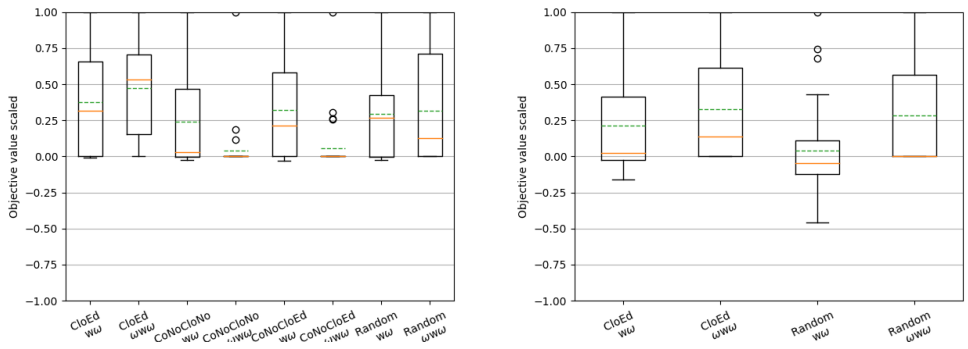
5.5.5 Comparing methods and functions

To further compare the three different potential KPI functions and the wω- and ωwω-methods, a weight is assigned to each function and method based on its performance. Table 5.1 displays the selected weights.

Table 5.1: A table with for every potential KPI function and every method, the weight that is chosen in order to compare the different methods and functions. The weight is considered to be the best instance-independent weight for that function and method.

Potential KPI	**Method**	**Best weight**
ClosestEdge	wω	51
	ωwω	71
ConnectingNodes vs ClosestNodes	wω	41
	ωwω	41
ConnectingNodes vs ClosestEdge	wω	91
	ωwω	91
Random linear function	wω	11
	ωwω	21

In Figures 5.16 and 5.17, the results of the different potential KPI functions with their best weights are shown. In these figures, both the performance in objective function as well as the performance in runtime can be assessed.
Firstly, the figures are used to compare the different potential KPI functions. It is clear from the figures that the CoNoCloNo and CoNoCloEd function do not perform very well in terms of the objective value. Especially the ωwω-method for these functions performs practically similar as the method without potential KPI. The wω-method performs somewhat better, but the median of the results of this method is lower than the median of the results of the random potential KPI function. The poor performance of these methods can be attributed to the limited number of iterations they undergo with potential KPI. Specifically, the ωwω-method runs very few iterations with potential KPI, as illustrated in Figure 5.17. Consequently, these methods exhibit low runtimes. However, the CoNoCloNo function always has a low runtime since this function is linear.
The CloEd potential KPI does perform very well in terms of the objective value as depicted in Figure 5.16. For both methods, using the CloEd function with a random initial solution performs practically always better than the method without potential KPI, with a median around 0.5. Moreover, the CloEd potential KPI function performs a lot better than the random potential KPI function in terms of objective value. Hence, this function seems to be the most effective as potential KPI. However, its performance in terms of runtime is mediocre. Adding CloEd greatly increases the runtime, as shown in Figure 5.17. Especially the iterations with potential KPI takes a substantial amount of time to run due to the non-linearity of the function.
The random potential KPI function also shows good performance, with the scaled objective value consistently remaining above 0. This efficacy is likely attributable to the significant number of iterations with potential KPI. As depicted in Figure 5.17, the methods using a random potential KPI function have numerous iterations with potential KPI. This implies that a lot of solutions are explored that were not explored with the local search algorithm without potential KPI. Consequently, there is a high likelihood that the random potential KPI function helps in finding a superior solution compared to the local minimum found with the ω-method.



(a) Random initial solution

(b) Greedy initial solution

Figure 5.16: A boxplot of the objective value for different methods with their best weight. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

Secondly, the two different methods, starting with or without potential KPI, are compared. In terms of objective value, the performance of the methods seems dependent on the used potential KPI. For the CoNoCloNo and CoNoCloEd functions, the wω-method seems to perform the best since their median and upper quartile are a lot higher than for the ωwω-method. For the CloEd function, the ωwω-method performs a bit better as the median and lower quartile are higher. Determining the best method for the random function is less straightforward, as the median of the wω-method is slightly higher, while the upper quartile of the ωwω-method is significantly higher. However, for the methods with a greedy initial solution, it is clear that the ωwω-method performs the best for both functions.

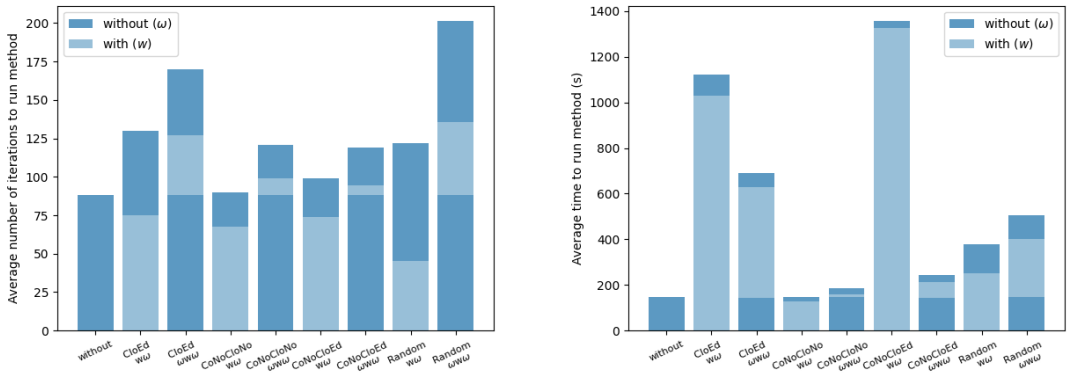


Figure 5.17: Histograms with the average runtime and number of iterations per method using a random initial solution.

In terms of runtime, the performance of the methods is also dependent on the used function. The runtime is primarily influenced by the number of iterations with potential KPI. For instance, with the CoNoCloEd function and CloEd, the ωwω-method operates considerably faster due to

fewer iterations with potential KPI compared to the wω-method. Conversely, with the random function, the number of iterations with potential KPI remains roughly consistent, resulting in a longer runtime for the ωwω-method. Generally, it is anticipated that the ωwω-method has a shorter runtime since it starts without potential KPI, and therefore the number of iterations with potential KPI is expected to be lower than that of the wω-method. Note that for most potential KPI functions, the iterations with potential KPI takes more time since computing the value for the potential KPI takes more effort than computing the original objective function. However, the exact runtime depends on how significantly the potential KPI steers the algorithm away from the local minimum. For CoNoCloNo, both methods show a similar runtime because the function is linear, and iterations with potential KPI proceed at a pace similar to those without potential KPI.

Statistical tests

In this section the results of the statistical tests for the TSP described in Section 5.4 are shown and discussed. In Tables 5.2 and 5.3, the methods that are compared and their p-value and test statistic U are depicted. Multiple methods and functions are compared using a statistical test. The ClosestEdge and random function as potential KPI are compared with the basic local search algorithm for both methods (wω and ωwω) and their best weight. Moreover, this is done for both initial solutions. Note that the other two functions, CoNoCloNo and CoNoCloEd, are not tested on significance, since it is clear from the figures that those functions perform barely better than the basic local search algorithm. Moreover, the random function is compared with the ClosestEdge function to determine if a well-designed function is significantly better than favouring random, predetermined solutions, i.e., a partly random objective function.

Table 5.2: Table with all the p-values and test statistics U of the statistical tests where method X is compared with method Y where both methods use the same random initial solution.

Method X	Method Y	U	p-value	Null hypothesis
ClosestEdge wω $\nu = 51$	No potential KPI ω	734.5	0.0151	rejected
ClosestEdge ωwω $\nu = 71$	No potential KPI ω	757.0	0.00654	rejected
Random wω $\nu = 11$	No potential KPI ω	604.0	0.449	
Random ωwω $\nu = 21$	No potential KPI ω	733.5	0.0156	rejected
Random wω $\nu = 11$	ClosestEdge wω $\nu = 51$	659.0	0.144	
Random ωwω $\nu = 21$	ClosestEdge ωwω $\nu = 71$	581.0	0.644	

In Table 5.2, with a random initial solution, it can be seen that three p-values are smaller than $\alpha = 0.05$. So, the null hypothesis is rejected for comparing the ClosestEdge potential KPI with the basic local search for both methods. Hence, using the ClosestEdge potential KPI is significantly better than using the basic local search with random initial solution regardless of the exact method of employment. Moreover, using the ωwω-method with a random function as potential KPI is significantly better than the basic local search method. For the other comparisons, the differences between the methods are not significantly. Hence, for example, the ClosestEdge function does not significantly perform better than a random potential KPI function.

Table 5.3: Table with all the p-values and test statistics U of the statistical tests where method X is compared with method Y where both methods use the same greedy initial solution.

Method X	Method Y	U	p-value	Null hypothesis
ClosestEdge wω $\nu = 51$	No potential KPI ω	532.0	0.878	
ClosestEdge ωwω $\nu = 71$	No potential KPI ω	647.5	0.188	
Random wω $\nu = 11$	No potential KPI ω	325.0	0.00497	rejected
Random ωwω $\nu = 21$	No potential KPI ω	632.0	0.264	
Random wω $\nu = 11$	ClosestEdge wω $\nu = 51$	715.0	0.0292	rejected
Random ωwω $\nu = 21$	ClosestEdge ωwω $\nu = 71$	556.0	0.888	

In Table 5.3, with a greedy initial solution, it can be seen that two p-values are smaller than $\alpha = 0.05$. The null hypothesis is not rejected for the comparison between the methods with ClosestEdge function and the basic local search algorithm. Thus, an algorithm with the ClosestEdge function as potential KPI performs somewhat better than the basic local search algorithm, but the difference is not significant when starting with a good initial solution such as a greedy solution. However, the ClosestEdge function does perform significantly different than a random function when using the wω-method with a greedy initial solution as the null hypothesis is rejected for this comparison. Moreover, in Figure 5.16, the two potential KPI functions can be compared and it is clear from this figure that the ClosestEdge performs better. Thus, the ClosestEdge potential KPI performs significantly better than the random function for the wω-method.
Also, the null hypothesis for the wω-method with a random solution and the basic local search method is rejected. In Figure 5.14a, it can be seen that the median of the boxes are below zero. Hence, by the Mann-Whitney U test and the figure, the wω-method with a random function as potential KPI and a greedy initial solution is significantly worse than the basic local search method.

5.6 Conclusion

Based on the results discussed in Section 5.5, some conclusions can be made. For the travelling salesman problem, three different constructed functions and one random function were tested. The four functions were first tested separately.
The first tested function was the ClosestEdge function. From the results, it seems like this function might be a good addition to improve a simple local search algorithm. The algorithm with CloEd added and starting from a random initial solution, is almost always better than the algorithm without any potential KPI and the difference with the basic local search algorithm is significant based on the Mann-Whitney U test. However, this is not always the case when starting the algorithm with a greedy solution, as the wω-method often performs worse than the ω-method for the greedy initial solution. The CloEd function was also tested for multiple weight values and also a value dependent on the instance. Using the instance-dependent weight, does not give a big difference in performance compared to the instance-independent weight, but the results are more consistent. It is unclear if the used bound, that was determined in Section 5.5, is a good bound since the results are similar as the results found with general weights. Overall, the experiments suggest that adding ClosestEdge to the objective function improves the local search algorithm. However, when the local search starts with a greedy solution, the results are less convincing.
The other two designed functions that have been tested, are the ConnectingNodes vs ClosestNodes and ConnectingNodes vs ClosestEdge. These two functions do not seem to perform well. Especially for the ωwω-method, the method performs practically the same as the method without potential KPI. The wω-method performs a bit better for higher weights. However, it performs significantly worse than a random potential KPI function. The poor performance was expected since the functions are somewhat similar to the original objective function. Due to the poor performance, these methods were not investigated further.
Also, a random function was tested as potential KPI. Adding the random function seems to improve the basic local search algorithm, for the ωwω-method, the difference with the basic local search algorithm is even significant when starting with a random initial solution. The method with random function and random initial solution is almost always better than the method without an addition and it even finds the optimal solution in some cases. The wω-method with a greedy initial solution, however, performs significantly worse than the basic local search algorithm according to the Mann-Whitney U test.
Thus, the ClosestEdge seems to work the best as potential KPI. However, the runtime of the method with this function is very high in comparison with other methods, like the basic local search without potential KPI or the method with the CoNoCloNo function. This is the case, because the CloEd function is non-linear and has a long computation time. For most methods, the runtime of the ωwω-method is shorter, since this method has a lower number of iterations with potential KPI function and the iterations with potential KPI function takes the most time since those functions are more complex than the original objective function. However, the ωwω-method with a random potential KPI function has a longer runtime than the wω-method. This is because both methods have roughly the same number of iterations with potential KPI. This also explains why this function performs so well, a lot of solutions, different from the ones explored with the original local search, are explored. The CoNoCloNo and CoNoCloEd functions do not explore a lot of solutions since the number of iterations with potential KPI are small. Hence, these functions do not perform very well.
In conclusion, a potential KPI can already be effective if just a random function is used as long as it is different enough from the original objective function. This has to be the case, since otherwise not enough other solutions are explored with the algorithm and the results would probably be

similar to the results of the CoNoCloNo and CoNoCloEd potential KPI functions. However, using a potential KPI that explores the right solutions, closer to the global optimum, can improve the local search algorithm even more than a random function. The first potential KPI, ClosestEdge, performs namely a lot better than the random linear function in terms of objective value. The runtime does increase due to the addition of a potential KPI.

Chapter 6

Discussion

In this chapter, the results obtained from the various experiments conducted in previous chapters are compared and discussed. Initially, several general considerations regarding potential KPIs, such as the methodology for identifying effective potential KPI functions and their implementation, are made. This was discussed in Chapter 3. Subsequently, these assertions were tested through practical optimisation problems. The used problems are the bucketised planning problem (Chapter 4) and the travelling salesman problem (Chapter 5). Now, it is evaluated whether the hypotheses proposed in Chapter 3 are indeed valid. Moreover, it is analysed, in this chapter, how the outcomes of the two problems compare across different dimensions: the potential KPI function, the impact of the initial solution, the method of implementation, and the effect of the weight value.

6.1 Potential KPI function

For the experiments with the optimisation problems, the methodology discussed in Section 3.2 to define useful potential KPI functions, is used. Through this approach, various functions were identified and developed. For the bucketised planning problem (BPP), the SquaredLoad function was constructed, while for the travelling salesman problem (TSP), three functions were constructed, namely the CloEd, CoNoCloNo, and CoNoCloEd functions. Moreover, for both problems, a random function was tested and compared with the constructed functions.
It is essential to acknowledge that this method of defining a potential KPI function does not consistently give effective functions. For instance, both CoNoCloNo and CoNoCloEd closely resemble the original objective function and, consequently, did not perform well in the conducted experiments. The other two functions, SquaredLoad and ClosestEdge, on the other hand, does perform well.
Furthermore, the findings align with the expectation that defining a suitable function becomes increasingly challenging for more complex problems, particularly when the characteristics of local optima are less understood. In such cases, the process of function derivation becomes more intricate and may require extensive experimentation and refinement. It was, for example, a lot harder to construct useful functions for the TSP than it was for the BPP.
Also, a random function was tested for both problems. The results of these function differ for both problems. For the BPP, the performance is highly dependent on the value of weight. For some weights, the random function for the BPP performs even better than the SquaredLoad function, while for other weights, the random function performs worse than using no potential KPI at all. For TSP, the random function performs better than the basic local search but worse than the CloEd function for all weights. However, for both problems, the performance is worse for higher weights. Hence, adding some random noise to the objective function performs well, but to much noise not.

In practice, defining a function involves some trial and error, coupled with an understanding of the characteristics of the local and global optima of the optimisation problem. This underscores the importance of iterative refinement and careful consideration of the problem's unique characteristics while defining a function that may be effective as potential KPI.

6.2 Impact of the initial solution

In Chapter 2, it was concluded that the initial solution is very important for the final solution of the local search algorithm. In this thesis, two different initial solutions were used to test potential KPIs, namely a random solution and a greedy solution dependent on the problem.
Generally, a local search algorithm with a greedy initial solution tends to perform better than with a random initial solution since the greedy solution is a better solution than the random one. Therefore, it is reasonable to expect that the methods with potential KPIs also perform better with a greedy initial solution, although to a lesser extent compared to the basic local search since less improvement is possible.
While the wω- and ωwω-methods with a greedy initial solution outperform the random solution in absolute terms for TSP, the improvements compared to the ω-method are more substantial with the random solution.
For BPP, the performance of the methods is similar for both initial solutions because the random solution is already fairly decent. However, due to the minor discrepancies, similar trends can be observed as with TSP.
Thus, potential KPIs tend to enhance the algorithm's performance more with a random initial solution than with a greedy one. However, the methods with a greedy initial solution and potential KPI yields the best overall solution among all methods.

6.3 Method of employment

In Chapter 3, the choice was made to use only a hard cut approach as method of employing the potential KPI. Moreover, it was concluded that the algorithm should always end with a phase with the original objective function. Thus, for BPP and TSP, two methods were tested, wω and ωwω. It seems like the effectiveness of the methods is largely dependent on the choice of potential KPI function, rather than the specific optimisation problem.
For instance, it is observed that the ωwω-method performed better when paired with the CloEd function in terms of objective value. On the other hand, the wω-method outperformed the ωwω-method when used for the other constructed functions, SquaredLoad, CoNoCloNo, and CoNoCloEd, as well as with the random functions for their best weights. This discrepancy likely stems from the nature of the potential KPI function and its impact on the behaviour of the local search algorithm. Specifically, functions that steer the algorithm away from local minima tend to benefit the ωwω-method, while those that guide the algorithm towards the global optimum favour the wω-method.
In terms of runtime, something similar seems to hold. The ωwω-method has a faster performance for the ClosestEdge function, while the wω-method is faster for the SquaredLoad, CoNoCloNo and random functions. However, the difference in runtime is also influenced by the number of iterations with the potential KPI, as these iterations have the most significant impact on the total runtime. Iterations with the potential KPI take more time to compute, because calculating the adapted objective function takes longer than calculating the original objective function, especially for non-linear functions. In general, it is expected that the ωwω-method has fewer iterations with potential KPI, as consequence of starting without potential KPI. However, the number of iterations

with potential KPI can vary a lot, and in some cases, the ωwω-method may have more of those iterations than the wω-method, as observed with the random function for TSP.
An advantage of the ωwω-method is its consistency; it always outperforms the ω-method due to its start without potential KPI.
Additionally, it was found that adding an extra phase with potential KPI (and one without) only marginally improves the local search method without this extra phase, while increasing the runtime, especially for large instances. Consequently, it may not be beneficial to implement an additional phase in the algorithm.
Overall, the tested methods, wω and ωwω, perform well. Therefore, there is no necessity to resort to complex methods to implement the potential KPI function.

6.4 Effect of weight value

In the potential KPI algorithm, it is possible to choose a value of weight ν in the adapted objective function in order to influence the importance of the potential KPI relative to the original objective function.
In Section 3.4, it was deduced that the weight value, resulting in the best performance of the methods, is likely dependent on the specific problem instance. Consequently, a general lower bound for the weight for minimisation problems and an upper bound for maximisation problems was established. However, the other bound was not found.
The lower bound was computed for both problems and applied in the experiments of the potential KPI methods. Initially, some general weight values, independent of the instance, were tested. For the SquaredLoad for the BPP, as well as for the CoNoCloNo and the CoNoCloEd for the TSP, higher weights yielded better results. In the case of the CloEd function, the outcomes varied significantly across different weight values. However, when using instance-dependent weights, the results of this function exhibited more consistency and an improvement for higher weight values was visible. In the case of the random functions, the performance improves for higher weights, until a certain point, after which the performance declines. This phenomenon was particularly evident in the BPP. Therefore, we can conclude that there is an upper bound for the weight in the case of random functions.
Although the instance-dependent weight approach proved effective for both problems, it was observed that the bound calculated for the weight tended to be lower than anticipated. This discrepancy likely arises from the rough estimation process. Moreover, the actual difference in potential KPI between the current and potential solutions is almost always higher the estimated minimum.
Interestingly, there appears to be no observable upper bound for the weight ν for the constructed functions, at least within the range of tested values. The results either improved or remained unchanged with higher weight values, indicating the absence of a clear upper bound.

Chapter 7

Conclusion

In this chapter, some conclusions are drawn and the research question is addressed. The research question posed is: "How can potential KPIs be used in local search algorithms to find better solutions and escape local optima using the same local operators?". Before delving into this question, several sub-questions, as formulated in Chapter 1, are addressed.

The first sub-question relates to whether the concept of a potential KPI already exists in literature (see Chapter 2). More specifically, it has been investigated whether there exist some algorithms with two types of features, 1) escaping local optimum, and 2) modifying the objective function. There are numerous local search algorithms and techniques with the first feature, i.e., that are able to escape local optima. These algorithms can generally be categorised into two main types: population-based and trajectory-based algorithms. Methods with the other feature are less prevalent. Furthermore, the combination of these approaches, essentially, the concept of a potential KPI, remains unexplored in the literature. However, there are somewhat similar concepts known, such as the guided local search (GLS). GLS introduces a penalty to the objective function once it becomes stuck in a local optimum that is not yet globally optimal. Similar to a potential KPI, the GLS adds something to the objective function based on the local optimum. In contrast, GLS only modifies the objective function when a local optimum is reached and the addition to the objective function is dependent on this local optimum, while a potential KPI function is already determined beforehand and can already be added to the objective function at the start of the algorithm. Thus, the concept of a potential KPI is not yet known in literature.

The second sub-question concerns the definition of a potential KPI. In essence, a potential KPI can be characterised as an addition to the objective function that rewards solutions that are closer to the global optimum in terms of some characteristics in comparison with another local optimum.
The necessity for a potential KPI arises when the used algorithm ends in an optimum that is not yet globally optimal and when the direction of searching should be changed, since simply accepting some worse solutions is not enough. Hence, a local search method can benefit from the use of a potential KPI. However, adding a potential KPI might not be useful for all optimisation problems, as constructing a potential KPI function necessitates some understanding of both the local and global optima.
The effectiveness of a local search algorithm with potential KPI heavily relies on the design of the function used as the potential KPI. Constructing such a function involves analysing the traits of both global and local optima. In the case of complex optimisation problems, defining such a function can be particularly challenging. Nevertheless, the results from the TSP experiments suggest

that utilising a well-designed function as a potential KPI can yield great improvements in the basic local search algorithm, more than when a random function is used as potential KPI, i.e. when some random solutions are accepted even though they are worse. Verifying if the constructed function is indeed a good potential KPI function is challenging, but could be done by empirically testing the function with data and/or comparing the function with a random noise function.
Employing the potential KPI, as previously defined, can be done in multiple ways. Choosing the method of employment involves determining whether to initiate and to end the algorithm with or without the potential KPI. It also entails deciding how to integrate the potential KPI into the algorithm, such as immediately applying the function or gradually modifying the objective function. Furthermore, it involves establishing the number of iterations required to navigate away from local optima while maintaining computational efficiency.
In this thesis, only a hard cut approach is examined, where the algorithm immediately switches between using the objective function with or without the potential KPI once the algorithm is converged. This approach is tested when starting with potential KPI as well as when starting with the original objective function. All the tested methods end with a phase without potential KPI, since this ensures that the final solution is as good as possible for the original objective. The results of the BPP and the TSP (see Chapters 4 and 5) indicate that both tested methods perform well. The decision to initiate the algorithm with or without the potential KPI is dependent upon the specific function and likely reflects the nature of the function. From a runtime perspective, the same observation holds. However, the runtime is highly dependent on the number of iterations with potential KPI, which indicates that in general starting without the potential KPI gives slightly better results. Furthermore, initiating without the potential KPI always produces equivalent or better solutions compared to basic local search alone.
Additionally, it has been established that incorporating an additional phase with the potential KPI does not give significant improvements in the algorithm performance relative to the increase in runtime.
Within the algorithm, the importance of the potential KPI relative to the original objective function can be determined through the weight parameter, denoted as ν. The performance of potential KPIs is tested with both instance-independent and instance-dependent weights. The outcomes indicated that for sufficiently high weights, the results of the instance-independent weight do not improve anymore for higher weights. The instance-dependent weight was explored by using a lower bound, derived from the difference between the current solution and its corresponding potential solution. The experiments detailed in Chapters 4 and 5 indicate that this lower bound serves as a practical weight value for constructed functions, since that value is higher than the actual lower bound for the weight and is in no proximity to an upper bound as far as visible. Moreover, the dependency of the weight on the instance appears to be better than a more general weight due to the consistency within the results. Thus, the methods perform good when the weight has the same value as this bound.

The last sub-question of this thesis is as follows: “How good does a local search algorithm with potential KPI perform in comparison with that algorithm without potential KPI?”. With the right potential KPI function, significant improvements in solution quality of the basic local search algorithm can be achieved. The final solution, found with local search with potential KPI, can have an objective value that is over 0.5 closer to the optimal value compared to the solution found without potential KPI.
However, adding a potential KPI does not always improve the basic local search algorithm, but does always increase the runtime. This increase in runtime can be particularly substantial when the potential KPI function is non-linear. Another disadvantage of a potential KPI is the difficulty in

determining an optimal weight value, although the impact of this value on the performance of the method is minor. Also, constructing a useful function is challenging and does greatly determine the performance of the algorithm. Constructing and verifying the effectiveness of such a function can be though, prolonging the time it takes before a potential KPI can be effectively applied in practice.

Now that all the sub-questions have been addressed, it is time to answer the main research question. A potential KPI serves as an additional component of the objective function, enabling the local search algorithm to escape local optima by changing the direction of the search. Various methods exist to find better solutions when using potential KPIs in a local search algorithm. Among the tested methods, the ωwω-method seems to be the most effective option due to its reduced runtime and its ability to be at least as good as the basic method. Note that the ωwω-method involves conducting a complete run of the basic local search followed by a run with the potential KPI added, and concluding the algorithm without the potential KPI. It is crucial to emphasise that the specific function used as the potential KPI plays an important role in the efficacy of this technique and which method of employment performs best.

Chapter 8

Recommendations for future research

In this chapter, some recommendations are discussed that can be used for future research. This thesis was the first research into the use of potential KPIs, so there is still a lot to investigate.

In this research, several conclusions have been drawn. However, to establish definitive general statements about potential KPIs and their usage, further research is needed, given the relatively limited scope of this thesis. Therefore, the methods examined in this study, wω, ωwω, wωwω and wωwω, should be investigated in more detail. This could involve testing them on a broader range of datasets. For instance, in the case of the bucketised planning problem, real-life data could be used to test the SquaredLoad function instead of artificially generated data, providing a more realistic assessment of the methods' performance.
Additionally, further investigation into the relationship between objective value or runtime and the weight value could yield valuable insights. Conducting additional tests to assess how the weight value impacts the algorithm's performance could provide valuable guidance for determining how to use potential KPIs in local search algorithms.

Two different methods of employing the potential KPI were tested in this thesis, wω and ωwω. However, there are lot of other possibilities to employ the potential KPI. For example, in this thesis, it was tested for BPP whether running the algorithm with additional wω-phases is useful. This could be investigated more, like investigating how many times a wω cycle needs to run in order to find the best possible solution.
Another variation in employing the potential KPI is to run the algorithm with potential KPI for a predetermined number of iterations instead of running the algorithm until convergence. This method might be beneficial, since the algorithm is probably faster since the predetermined number of iterations is probably smaller than the number of iterations needed for convergence. Moreover, running the algorithm multiple times with and without potential KPI in this case, might be more beneficial than it was for the tested methods, since with a predetermined number of iterations, not all solutions are explored yet in the first phase. Hence, investigating this method would be interesting. However, there are also multiple challenges with this approach. It is difficult, for instance, to determine the precise number of iterations required to escape the local optimum. Running only one iteration with the potential KPI may not suffice, as there is a risk of the local search reverting back to the previously encountered local optimum. Further research is needed to determine the best number of iterations for this approach.
Another option for employment is to gradually adjust the weight to a specific value, known as a soft cut (refer to Section 3.3) instead of an immediate transition. This method has been shortly

tested, but was not investigated in more detail, since the first results were similar to the results of the hard cut method but with increased runtime. Nevertheless, further investigation into the soft cut method may yield valuable insights.
Alternatively, the weight could be set as a constant percentage of the current objective value. In this scenario, the percentage remains constant, but the absolute value of the weight adjusts based on the objective value. An advantage of this method is its instance-dependent weight, which is expected to work best. A disadvantage is that it is hard to determine an useful percentage. Moreover, there is a risk that the algorithm fails to converge due to this adjustment. To address this concern, it is advisable to implement a predetermined number of iterations after which the algorithm should terminate.

Another possibility for a useful future research would be to test the potential KPI in other algorithms. While this thesis focused on a simple local search algorithm, there are numerous alternative local search algorithms that are interesting to investigate, like continuing with the first neighbour that is better instead of first investigating all the neighbourhood solutions. Furthermore, it would be interesting to asses whether potential KPIs can improve more complex local search heuristics, like simulated annealing or tabu search. However, it is expected that adding a potential KPI would be less useful for these more complex methods because they are already capable of escaping local optima.

It would also be intriguing to conduct a comparative analysis between the simple local search algorithm with potential KPI and other heuristics, such as simulated annealing or tabu search. It can be investigated whether a local search algorithm with potential KPI is a good alternative for these heuristics and which method is better to use for practical problems. Such a comparison could be based on several factors including solution quality, runtime, and ease of understanding.
However, comparing different heuristics can be challenging due to their high dependency on various parameters, making it difficult to establish whether differences in methods are attributable to genuine distinctions or simply variations due to the parameter values. An approach to address this problem, is to select parameter values for the heuristic that are known to produce good results. For well-known problems like the TSP, such values may be available in existing literature. With these parameters, it becomes easier to compare the heuristics.

In this thesis, only a couple of different functions are tested as potential KPI. For the bucketised planning problem only one function, SquaredLoad was tested, and for the travelling salesman problem three functions were tested, CloEd, CoNoCloNo and CoNoCloEd. It might be interesting to test more different functions and compare them to investigate the importance of a correct function. For the BPP, for instance, another function, that could work as potential KPI, is a function based on the guided local search heuristic (see Chapter 2). This GLS-based function would then look something like

$$\sum_{j \in J} I_j(S) \qquad \text{with } I_j(S) = \begin{cases} 1 & \text{If } \sum_{i \in I} L_{ij} \geq T \\ 0 & \text{Else} \end{cases}$$

for solution S.
For the TSP, a possible other potential KPI function would be to count the number of shortest edges in your path or look more into the difference between the current edge and the shortest edge. For the second option, the closer the costs of the edge in your path are to the shortest edge possible for this node, the lower the penalty in the potential KPI function. This function would

likely resemble something like

$$\sum_{i \in I} c_{ij} - c_i^{\min}$$

where $j \in I$ is the node next to node i in your path and $c_i^{\min}$ the minimum costs for node i.

Furthermore, the potential KPI as an addition to the objective function could be tested across different optimisation problems. If the methods with potential KPI are applied and tested to more problems, it can be determined whether certain statements hold true for all types of problems, such as which method of employment works the best. It might also be possible to conclude that a potential KPI is only effective for certain types of problems. It is expected that a potential KPI is more useful for easily understandable problems, like scheduling or planning problems. It is namely easier to find a useful potential KPI function for these problems, given their greater understanding of their local and global optima. For a similar reason, a problem with better relaxation bounds might work better as well. Thus, it would be interesting to investigate more of these types of problem versus other problems to see if the expectation indeed holds.
A suggested scheduling problem that might be worth investigating is a single machine scheduling problem with setup times. This problem is frequently encountered by OMP. However, the local optimum for this problem is not as evident as it seems.
Another problem that might be worthwhile to investigate is a maximisation problem, as the behaviour of a potential KPI could be dependent on whether the objective is to minimise or maximise. An example of a potentially interesting maximisation problem is the Maximal Covering Location Problem.

Moreover, it might be interesting to assess the quality of solutions obtained with just the potential KPI function. This can be accomplished by first computing the optimal value of the optimisation problem with the potential KPI function as objective function, instead of the original objective function with or without potential KPI. Then, this optimum can be used as initial solution in the local search algorithm with original objective function. This approach can be used to investigate whether it is necessary to retain some information from the original objective function in the search process and whether the optimal solution for the potential KPI function performs effectively as a potential solution.
However, determining the optimum for the potential KPI might be challenging, particularly since most potential KPI functions tend to be non-linear, as most tested functions in this thesis were. Consequently, defining an ILP and employing an ILP solver would be cumbersome.

The last suggestion for future research involves investigating the use of multiple potential KPIs for a single problem. Given that a problem typically has multiple local optima with distinct characteristics, employing various types of potential KPI functions may be necessary. Each potential KPI function could be designed to target a specific local optimum and its corresponding potential solution, thereby facilitating the escape from different types of local optima. Integrating multiple functions can be done in various ways, such as adding all functions to the original objective function simultaneously or adding them consecutively to the original objective function.

Thus, there remains much to explore regarding the utilisation of potential KPIs in local search algorithms and heuristics. Before using an algorithm with potential KPI in practice, it is important to assess its effectiveness more thoroughly, since the scope of this research is limited.

Bibliography

[1] P. S. Oliveto, T. Paixão, J. Pérez Heredia, D. Sudholt, and B. Trubenová, "How to escape local optima in black box optimisation: when non-elitism outperforms elitism," *Algorithmica*, vol. 80, pp. 1604–1633, 2018.

[2] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers Operations Research*, vol. 13, no. 5, pp. 533–549, 1986. Applications of Integer Programming.

[4] E. F. Codd, "Multiprogram scheduling: Parts 1 and 2. introduction and theory," *Communications of the ACM*, vol. 3, no. 6, pp. 347–350, 1960.

[5] S. M. Goldfeld, R. E. Quandt, and H. F. Trotter, "Maximization by quadratic hill-climbing," *Econometrica: Journal of the Econometric Society*, pp. 541–551, 1966.

[6] J. Baxter, "Local optima avoidance in depot location," *Journal of the Operational Research Society*, vol. 32, no. 9, pp. 815–819, 1981.

[7] P. Hansen and N. Mladenović, "Variable neighborhood search for the p-median," *Location Science*, vol. 5, no. 4, pp. 207–226, 1997.

[8] T. Bäck and H.-P. Schwefel, "EAAn Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[9] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM journal on computing*, vol. 2, no. 2, pp. 88–105, 1973.

[10] T. Friedrich, T. Kötzing, M. S. Krejca, and A. Rajabi, "Escaping local optima with local search: a theory-driven discussion," in *International Conference on Parallel Problem Solving from Nature*, pp. 442–455, Springer, 2022.

[11] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, pp. 3–52, 2002.

[12] F. Hoffmeister and T. Bäck, "Genetic algorithms and evolution strategies: Similarities and differences," in *International conference on parallel problem solving from nature*, pp. 455–469, Springer, 1990.

[13] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied soft computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[14] P. Moscato *et al.*, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, no. 1989, p. 37, 1989.

[15] A. Chakraborty and A. K. Kar, "Swarm intelligence: A review of algorithms," *Nature-inspired computing and optimization: Theory and applications*, pp. 475–494, 2017.

[16] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pp. 39–43, IEEE, 1995.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[18] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of insect behavior*, vol. 3, pp. 159–168, 1990.

[19] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm—a novel tool for complex optimisation problems," in *Intelligent production machines and systems*, pp. 454–459, Elsevier, 2006.

[20] R. Oftadeh and M. Mahjoob, "A new meta-heuristic optimization algorithm: Hunting search," in *2009 fifth international conference on soft computing, computing with words and perceptions in system analysis, decision and control*, pp. 1–5, IEEE, 2009.

[21] O. K. Erol and I. Eksin, "A new optimization method: big bang–big crunch," *Advances in engineering software*, vol. 37, no. 2, pp. 106–111, 2006.

[22] X.-S. Yang, "Harmony search as a metaheuristic algorithm," *Music-inspired harmony search algorithm: theory and applications*, pp. 1–14, 2009.

[23] H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti, and D. N. Jawawi, "Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm," *Swarm and Evolutionary Computation*, vol. 26, pp. 8–22, 2016.

[24] A. Kaveh and T. Bakhshpoori, "Water evaporation optimization: a novel physically inspired optimization algorithm," *Computers & Structures*, vol. 167, pp. 69–85, 2016.

[25] M. Abdechiri, M. R. Meybodi, and H. Bahrami, "Gases brownian motion optimization: an algorithm for optimization (gbmo)," *Applied Soft Computing*, vol. 13, no. 5, pp. 2932–2946, 2013.

[26] B. Alatas, "Acroa: artificial chemical reaction optimization algorithm for global optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13170–13180, 2011.

[27] O. C. Martin and S. W. Otto, "Combining simulated annealing with local search heuristics," *Annals of operations research*, vol. 63, no. 1, pp. 57–75, 1996.

[28] C. Voudouris, *Guided local search for combinatorial optimisation problems.* PhD thesis, University of Essex, 1997.

[29] S. Kakutani, "A generalization of Brouwer's fixed point theorem," *Duke Mathematical Journal*, vol. 8, no. 3, pp. 457 – 459, 1941.

[30] P. Meseguer, F. Rossi, and T. Schiex, "Soft constraints," in *Foundations of Artificial Intelligence*, vol. 2, pp. 281–328, Elsevier, 2006.

[31] J. D. Knowles and D. W. Corne, "M-paes: A memetic algorithm for multiobjective optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, pp. 325–332, IEEE, 2000.

[32] I. Y. Kim and O. L. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and multidisciplinary optimization*, vol. 29, pp. 149–158, 2005.

[33] Delft High Performance Computing Centre (DHPC), "DelftBlue Supercomputer (Phase 2)." `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2`, 2024.

[34] J. P. Klein, M. L. Moeschberger, J. P. Klein, and M. L. Moeschberger, *Hypothesis testing*. Springer, 2003.

[35] E. S. Pearson, "" student" as statistician," *Biometrika*, vol. 30, no. 3/4, pp. 210–250, 1939.

[36] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[37] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[38] V. W. Berger and Y. Zhou, "Kolmogorov–smirnov test: Overview," *Wiley statsref: Statistics reference online*, 2014.

[39] P. A. Pappas and V. DePuy, "An overview of non-parametric tests in sas: when, why, and how," *Paper TU04. Duke Clinical Research Institute, Durham*, pp. 1–5, 2004.

[40] F. Wilcoxon, "Some uses of statistics in plant pathology," *Biometrics Bulletin*, vol. 1, no. 4, pp. 41–45, 1945.

[41] P. Orponen and H. Mannila, *On approximation preserving reductions: Complete problems and robust measures*. University of Helsinki, 1987.

[42] G. Reinelt, "Tsplib95," *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg*, vol. 338, pp. 1–16, 1995. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`.

[43] N. Christofides, "The optimum traversal of a graph," *Omega*, vol. 1, no. 6, pp. 719–732, 1973.

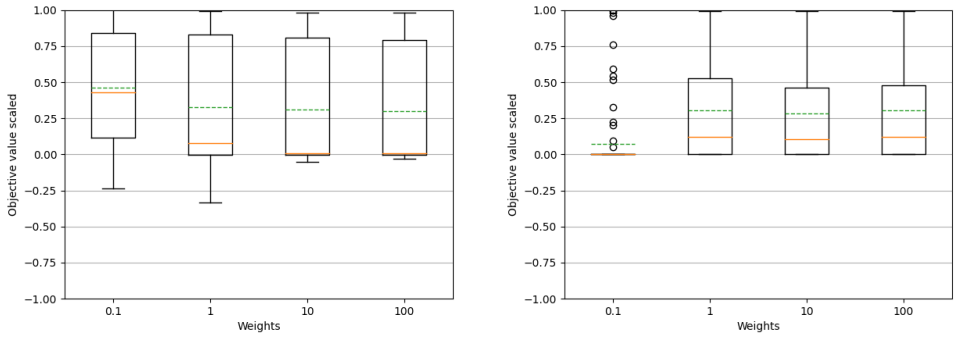
Appendix A

Extra results of the Bucketised Planning Problem

In this appendix, additional results for the bucketized planning problem are presented. Extra experiments have been conducted, such as those using simple data to test whether the SquaredLoad function works for the most basic examples. Additionally, other variants of the local search algorithm have been tested (see Section 4.2). This appendix also includes additional figures of the results discussed in Section 4.5.

A.1 Simple data

The SquareLoad potential KPI is tested for some simple data. This data consists of 100 randomly generated instances of 20 tasks with varying process times and due dates where the process times are between 0 and $\frac{15 \cdot 7}{10}$ to avoid weird instances and due dates between 0 and $15 \cdot 7$. A time frame of 7 buckets of 15 time units are used.



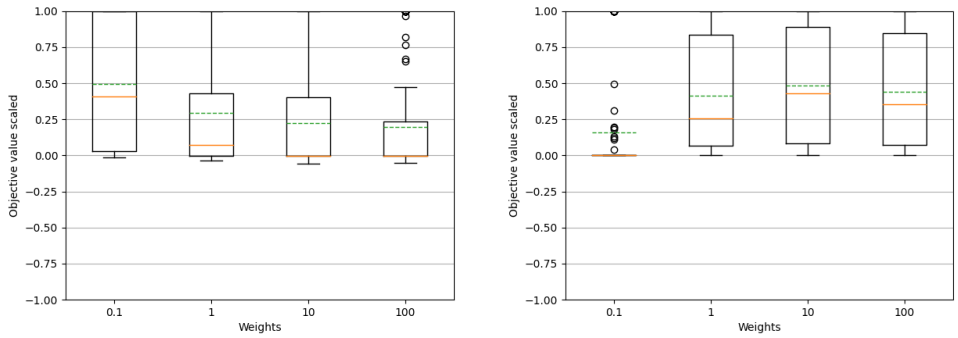
(a) wω-method. (b) ωwω-method.

Figure A.1: A boxplot of the objective value of the method with the SquaredLoad potential KPI per weight based on 100 instances of 20 tasks and 7 buckets for a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

In Figure A.1, the results of the SquaredLoad potential KPI with this data is shown for the two local search methods with potential KPI: the wω-method and the ωwω-method (see Section 3.3). In the figure, a scaled objective value is shown. The scaled value is as follows:

$$\begin{cases} \frac{f(S_{\text{initial}})-f(S)}{f(S_{\text{initial}})-f(S_{\text{without}})} - 1 & \text{if } f(S) \leq f(S_{\text{without}}) \\ \frac{f(S_{\text{without}})-f(S)}{f(S_{\text{without}})-f(S_{\text{optimum}})} & \text{if } f(S) \geq f(S_{\text{without}}) \end{cases}$$

where f is the objective function, S_{initial} the initial solution, S_{without} the solution found with the standard local search and S_{optimum} the optimal solution.
The objective value of the optimum is calculated with an integer linear problem solver. The average optimum objective value for 7 buckets is equal to approximately 21500 and there are only 9 instances for which the optimal value is below 1000. Hence, for most instances, it is impossible to have no overload. This is the case because of how the data is generated. It is often not possible to stay within the capacity of the buckets because the process times of all tasks combined is higher than the total available time $15 \cdot 7$. In reality, this is not possible, you can always use extra time buckets to be able to plan everything. Therefore, also some data is generated where we allow a large enough time frame, namely 14 buckets instead of 7. In Figure A.2, the results for the data with 14 buckets is shown in a similar format as for 7 buckets.



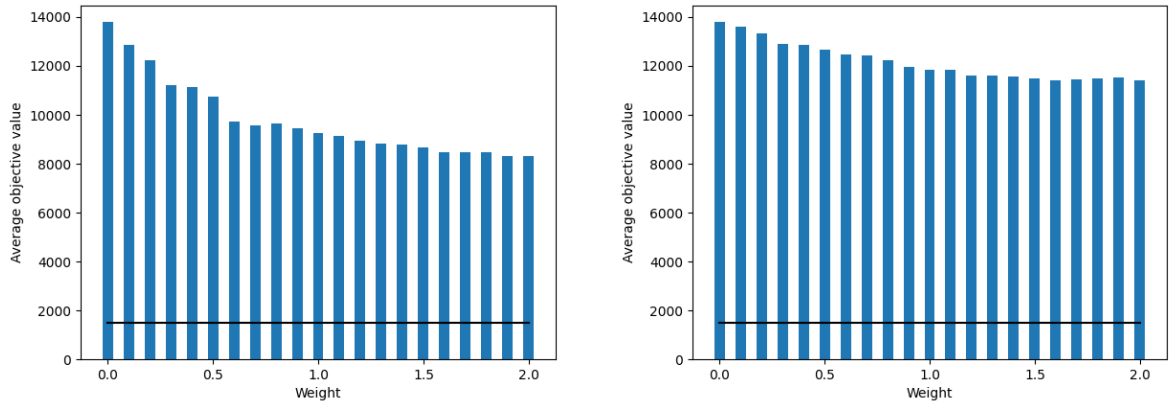
(a) wω-method.

(b) ωwω-method.

Figure A.2: A boxplot of the objective value of the method with the SquaredLoad potential KPI per weight based on 100 instances of 20 tasks and 14 buckets for a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

A.2 Complicated data

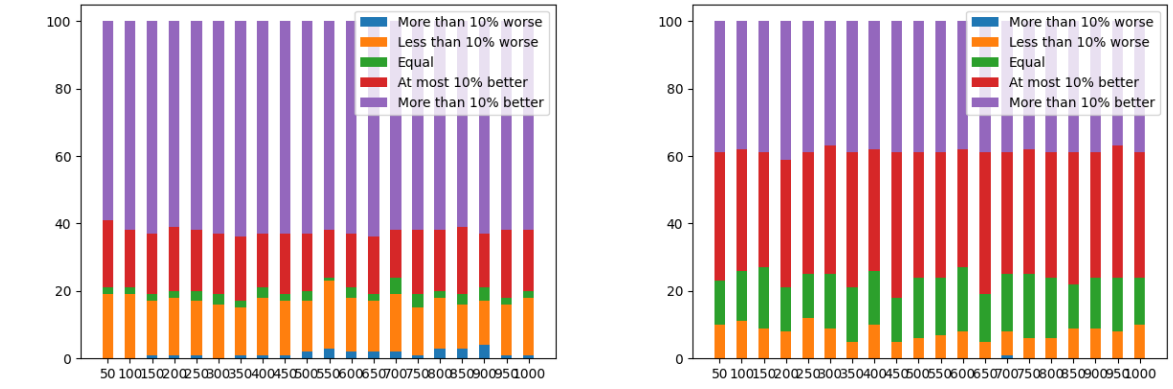
In this appendix, some extra figures for the more complicated data are shown. These figures can give some extra insight in the results discussed in Section 4.5.



(a) wω-method. (b) ωwω-method.

Figure A.3: A histogram with the average objective value of 100 instances per weight for small weights.

In Figure A.3, the average objective value is plotted for different weights. In the figure also the optimum is shown as a black line. In the figure it can be seen that the objective value improves for higher weights.



(a) wω-method. (b) ωwω-method.

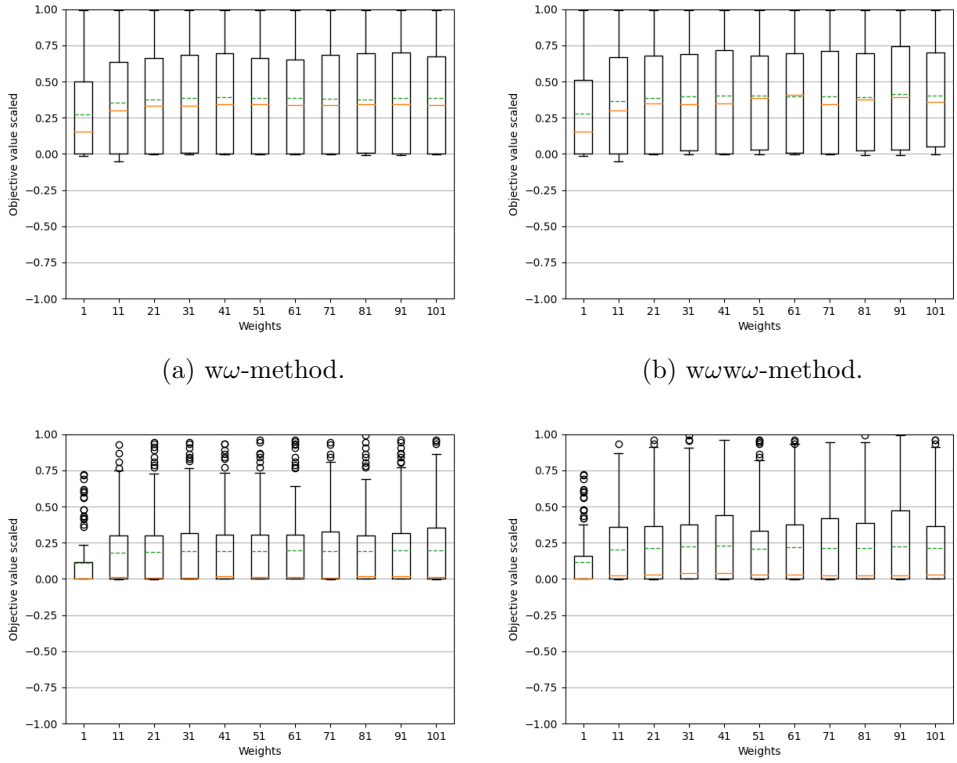
Figure A.4: A histogram with per weight the number of times the method with potential KPI is better with more than 10% of the objective value, better with less than 10%, equal, worse with less than 10% or worse with more than 10% than the method without potential KPI in terms of objective value.

In Figure A.4, a histogram is shown where the number of times the method with potential KPI is worse or better is count. From the figure it can be seen how many times the new method is better

than the basic local search.

A.2.1 Variant 1 of the local search algorithm

There are three different variants of the local search algorithm used in this thesis, see Section 4.2. For the results discussed in Section 4.5, the second variant of the algorithm is used since this variant is expected to give the best results. In this section, the results of the first variant are shown in similar figures as the results of variant 2 (see Section 4.4 for explanation of the experiments), see Figure A.5. It can be seen from the figure, that the results are very similar to the results of variant 2 of the alorithm as expected.



(a) wω-method.

(b) wωwω-method.

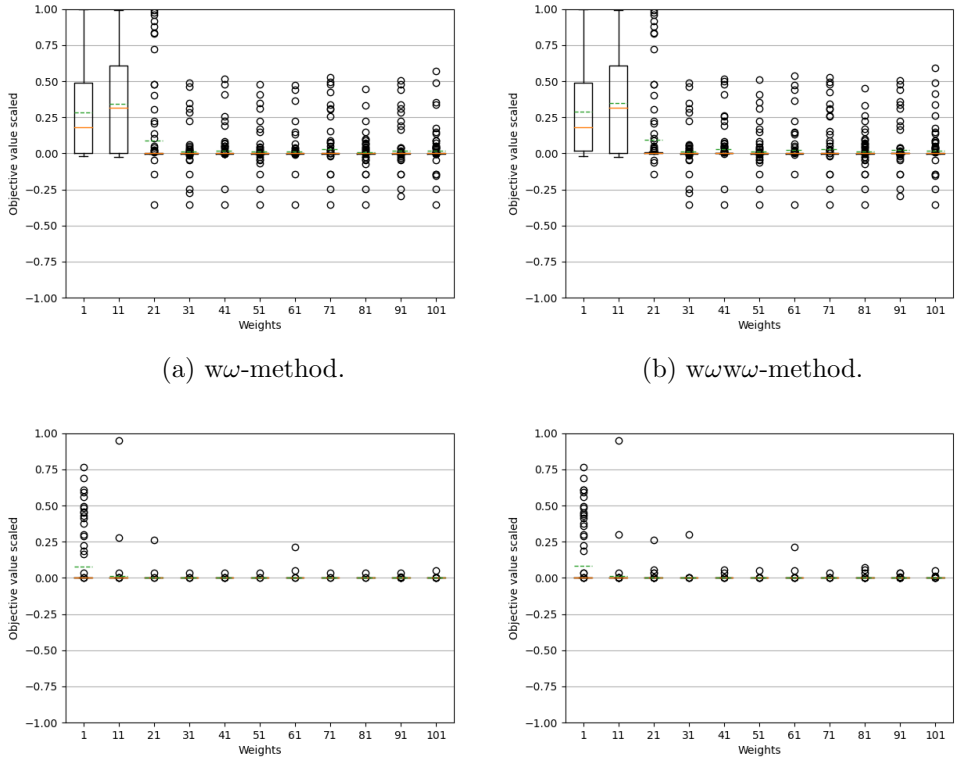
(c) ωwω-method.

(d) ωwωwω-method.

Figure A.5: A boxplot of the objective value of the method per weight based on 100 instances of varying size with a random start solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

A.2.2 Variant 3 of the local search algorithm

The local search algorithm used in this thesis, see Section 4.2, has three different variants. The results of the third variant are shown in this section. The results are shown in similar figures as the results of variant 1 and 2 (see Section 5.5 and Section A.2.1), see Figure A.6. As expected, the results are very similar to the ω-method, since often the solution found with potential KPI is worse than the one found without and the algorithm will continue with that solution.



(a) wω-method.

(b) wωwω-method.

(c) ωwω-method.

(d) ωwωwω-method.

Figure A.6: A boxplot of the objective value of the method per weight based on 100 instances of varying size with a random start solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

Appendix B

Extra results of the Travelling Salesman Problem

In this appendix, additional results for the travelling salesman problem are presented. Extra experiments using simple data have been conducted. This has been done to test whether the three designed functions; ClosestEdge, ConnectingNodes vs. ClosestNodes and ConnectingNodes vs. ClosestEdge (see Section 5.3) work for the most basic examples. Additionally, other variants of the local search algorithm have been tested (see Section 5.2). This appendix also includes additional figures of the results discussed in Section 5.5.

B.1 Simple data

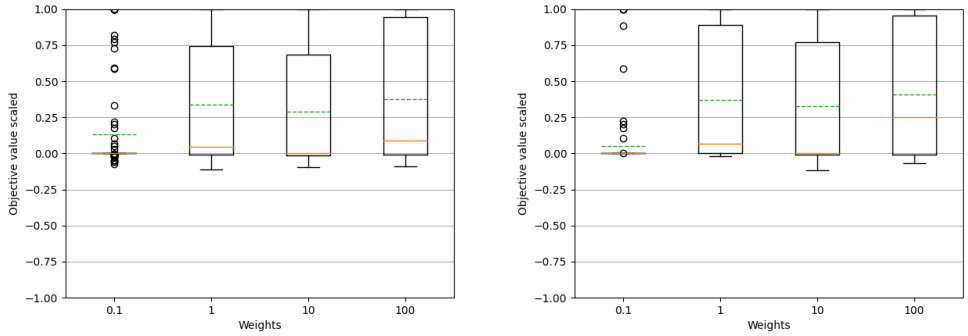
The data used for the first tests for the travelling salesman problem is a set of 100 randomly generated instances consisting of 25 cities in a 10 by 10 space. The objective value of the optimum is calculated with an integer linear program solver of Gurobi. For this dataset the first variant of the 2-opt local search method as described in Section 5.2 is used. Hence, note that the ωwω-method can also perform worse than the basic local search method, even though first the basic local search algorithm is run.

B.1.1 ClosestEdge

In this section, the results of the ClosestEdge potential KPI function for the simple dataset created as explained above, are shown. In Figure B.1, the results are shown in boxplots where a scaled objective value is used to compare the method with potential KPI with the basic local search method and with the optimal solution. The scaled value is defined as follows:

$$\begin{cases} \frac{f(S_{\text{initial}})-f(S)}{f(S_{\text{initial}})-f(S_{\text{without}})} - 1 & \text{if } f(S) \leq f(S_{\text{without}}) \\ \frac{f(S_{\text{without}})-f(S)}{f(S_{\text{without}})-f(S_{\text{optimum}})} & \text{if } f(S) \geq f(S_{\text{without}}) \end{cases}$$

where f is the objective function, S_{initial} the initial solution, S_{without} the solution found with the standard local search and S_{optimum} the optimal solution. The different methods, wω and ωwω (see Section 3.3), are tested with weights 0.1, 1, 10, 100.



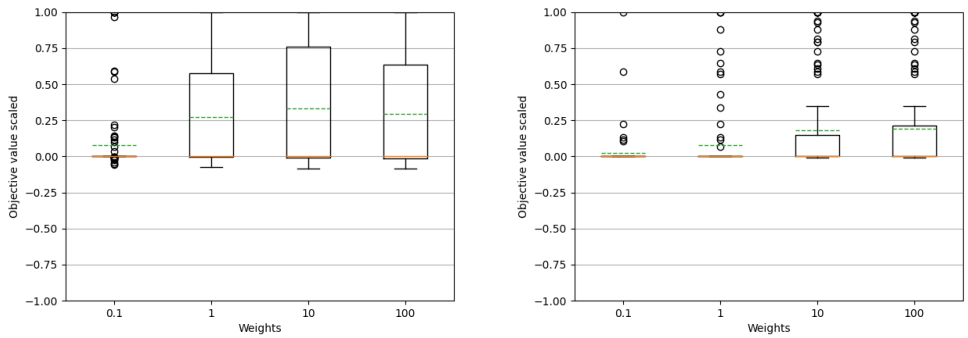
(a) wω-method (b) ωwω-method

Figure B.1: Boxplots of the objective value of the method per weight for the ClosestEdge potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

From the figure, it is clear that the methods with potential KPI and with weights higher than 0.1 are practically always better than the basic local search method as the median and mean are above zero.

B.1.2 ConnectingNodes vs. ClosestNodes

Here, the results of the local search algorithm with as potential KPI the CoNoCloNo function (see Section 5.3) based on the simple data, is shown.



(a) wω-method (b) ωwω-method

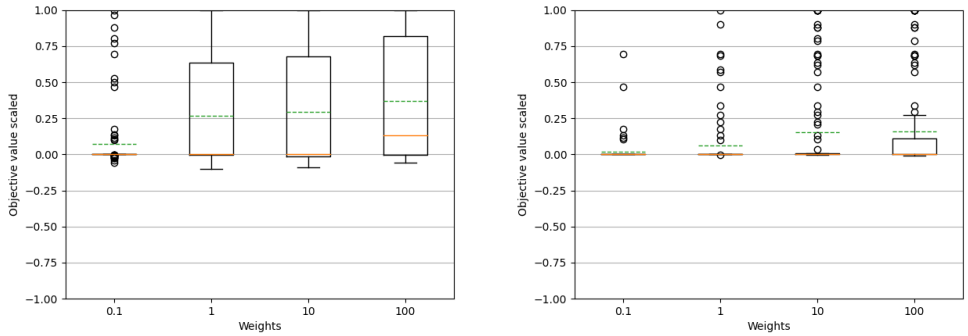
Figure B.2: Boxplots of the objective value of the method per weight for the ConnectingNodes vs. ClosestNodes potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

In Figure B.2, the results are shown in boxplots where the scaled objective value (see previous

subsection) is used to compare the wω- and ωwω-method with the basic local search method and with the optimal solution. Four different value of weight are used, 0.1, 1, 10 and 100. The wω-method for weights 1, 10 and 100 seems to perform better than the basic local search according to Figure B.2 and occasionally finds the optimal solution. However, the ωwω-method seems to perform barely better than the basic local search method.

B.1.3 ConnectingNodes vs. ClosestEdge

In Figure B.3, boxplots are shown with the results of the CoNoCloEd potential KPI based on the simple data created above. In the boxplots, the scaled objective values as defined above for the wω- and ωwω-methods per weight (0.1, 1, 10, 100) are shown. The results are similar as the results for the CoNoCloNo potential KPI, i.e., the wω-method does perform relatively better than the basic local search method, but the ωwω-method does not perform better or worse.



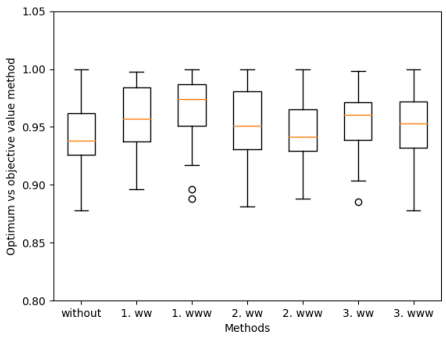
(a) wω-method

(b) ωwω-method

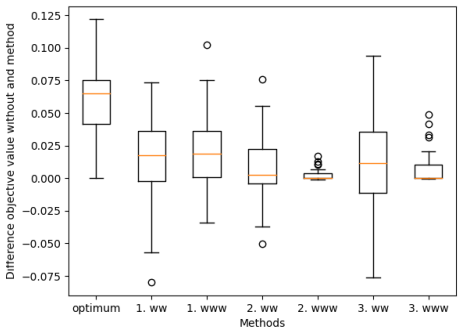
Figure B.3: Boxplots of the objective value of the method per weight for the ConnectingNodes vs. ClosestEdge potential KPI function with a random initial solution. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

B.2 Data from internet

In this section, some extra figures for the more complicated data, i.e. the dataset from internet [42] (see Section 5.4 for more information on this dataset), are shown. These figures can give some extra insight in the results discussed in Section 5.5.



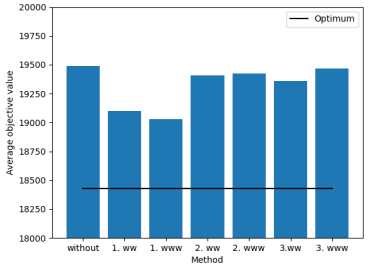
(a) Boxplots of the optimum objective value in comparison with the objective value of different methods.



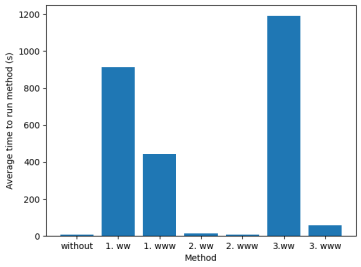
(b) Boxplots of the scaled difference in objective value of ω-method and the methods with potential KPI.

Figure B.4: Figures with the results of the different potential KPIs for the different methods as described in Sections 3.3 and 5.3. The weights of the methods are the weights that are the best for that method based on earlier results found in Sections 5.5.1, B.2.2 and B.2.3. The results are based on 33 instances from a benchmark set from the internet TSPlib95.

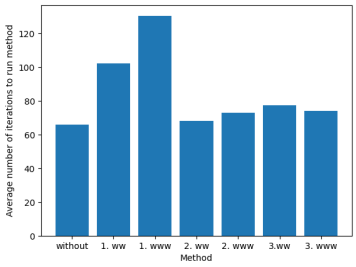
In Figure B.4, boxplots of the different potential KPI methods are shown. On the y-axis in Figure B.4a, a scaled objective value is shown, which is the optimal value divided by the objective value found with the local search method. Thus, values closer to 1 are desirable since a value of 1 means that the solution found with potential KPI is equal to the optimal solution. In Figure B.4b, the y-axis is equal to the difference in objective value between the local search with potential KPI and the method without potential KPI divided by the objective value of the basic local search method. Hence, if the potential KPI method performs worse than the basic local search, than a negative value occurs in the plot.



(a) A histogram with the average objective values for different methods.



(b) A histogram with the average runtime for different methods.

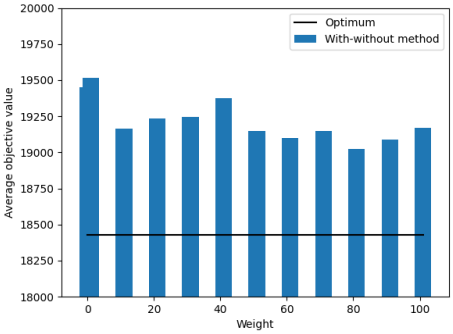


(c) A histogram with the average number of iterations for different methods.

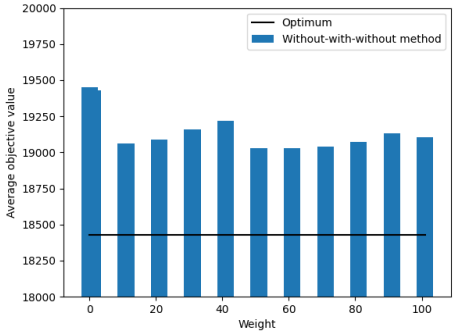
Figure B.5: Figures with the results of the different potential KPIs for the different methods as described in Sections 3.3 and 5.3. The weights of the methods are the weights that are the best for that method based on earlier results found in Sections 5.5.1, B.2.2 and B.2.3. The results are based on 33 instances from a benchmark set from the internet TSPlib95.

B.2.1 ClosestEdge

In this section, some extra figures for the ClosestEdge function, that can clarify the results discussed in Section 5.5, are shown, see Figures B.6 and B.7. Moreover, the results of the other variants of the used algorithm (see Section 4.2) are shown here in Figures B.8, variant 1 and B.9, variant 3.

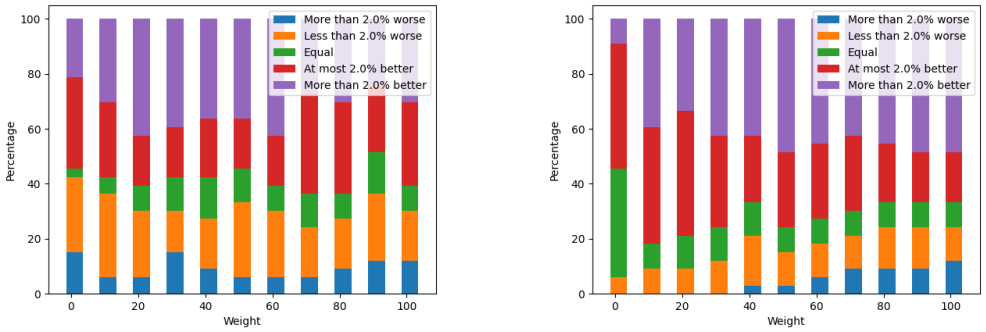


(a) wω-method.



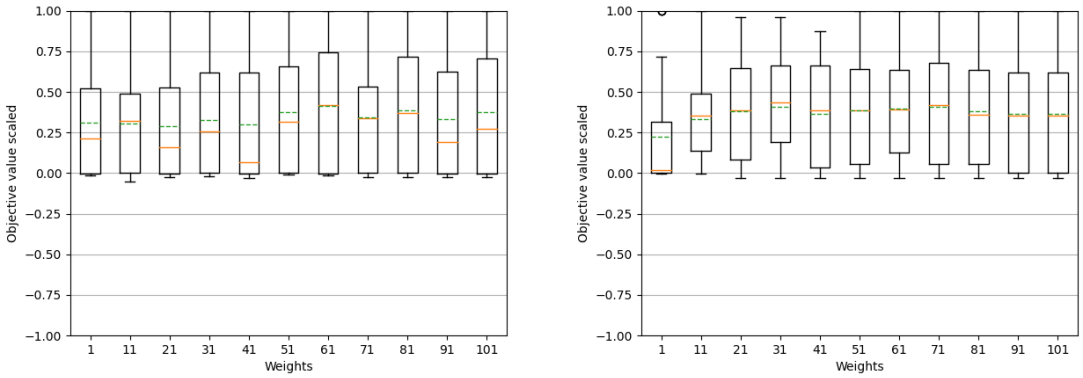
(b) ωwω-method.

Figure B.6: A histogram with the average objective values found with the local search algorithm starting from a random start solution and the ClosestEdge Potential KPI for different values of weight. The black line is equal to the average optimum value for all instances. The results are based on 33 instances from a benchmark set from the internet, TSPlib95.



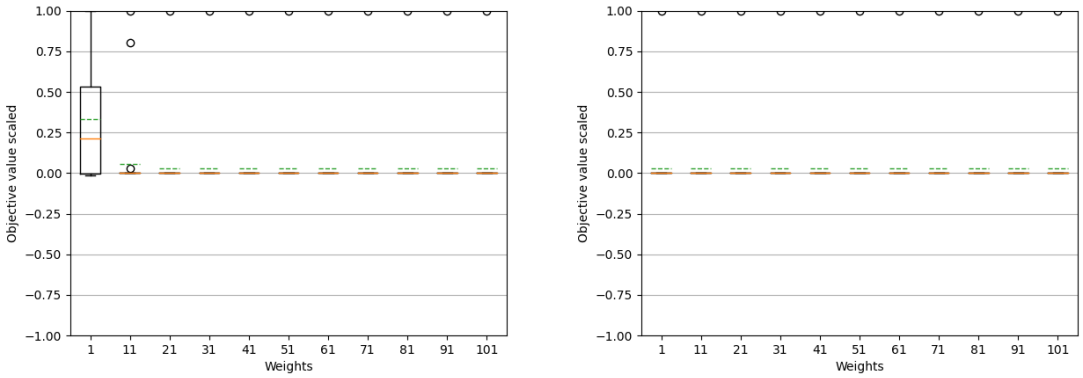
(a) wω-method. (b) ωwω-method.

Figure B.7: A histogram with the number of times the used method with the ClosestEdge potential KPI is better (or worse) than the ω-method. The results are based on 33 instances from a benchmark set from the internet, TSPlib95.



(a) wω-method (b) ωwω-method

Figure B.8: Boxplots of the objective value of the method per weight for the ClosestEdge potential KPI function with a random start solution and the first variant of the local search algorithm. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

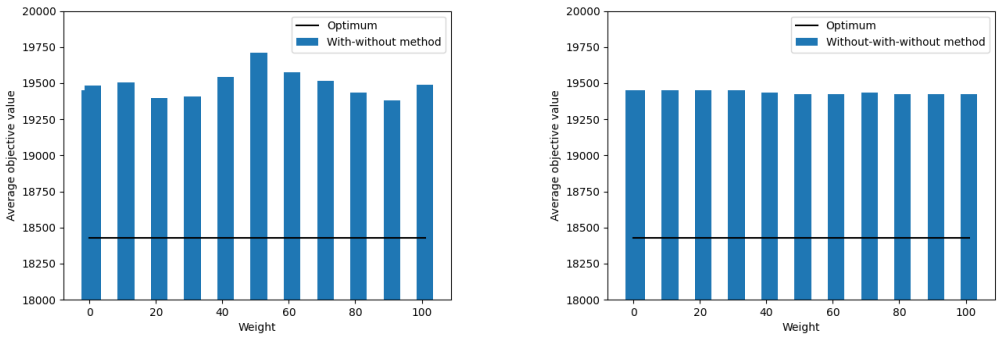


(a) wω-method (b) ωwω-method

Figure B.9: Boxplots of the objective value of the method per weight for the ClosestEdge potential KPI function with a random start solution and the third variant of the local search algorithm. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

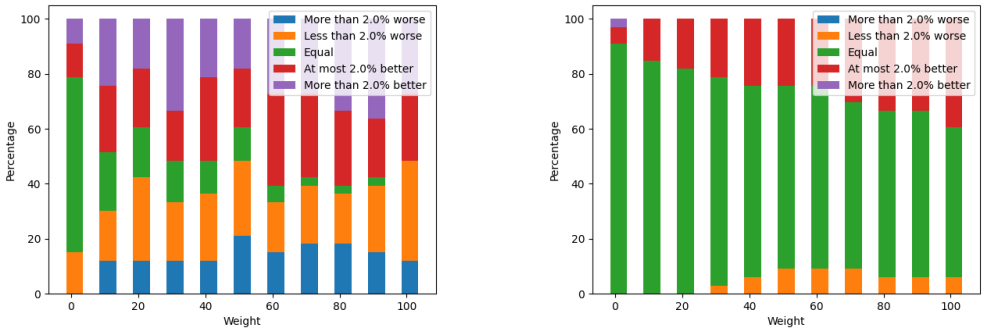
B.2.2 ConnectingNodes vs ClosestNodes

In this section, some extra figures for the CoNoCloNo potential KPI, that can clarify the results discussed in Section 5.5, are shown.
Furthermore, some boxplots of the objective value found with the first and third variant of the local search algorithm (see Section 5.2) are shown. Figures B.12 and B.13 show the results in boxplots that are the same as used for the second variant shown in Section 5.5.



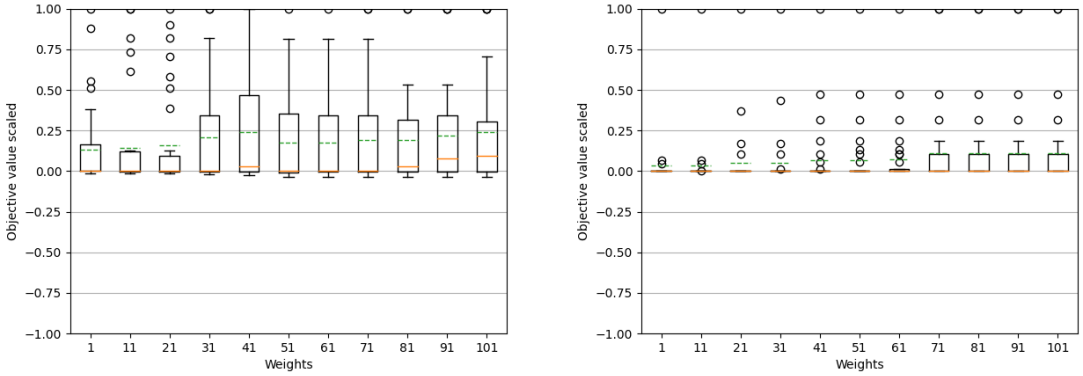
(a) wω-method. (b) ωwω-method.

Figure B.10: A histogram with the average objective values found with the local search algorithm starting from a random start solution and the CoNoCloNo potential KPI for different weights. The black line is equal to the optimum value. The results are based on 33 instances from a benchmark set from the internet, TSPlib95.



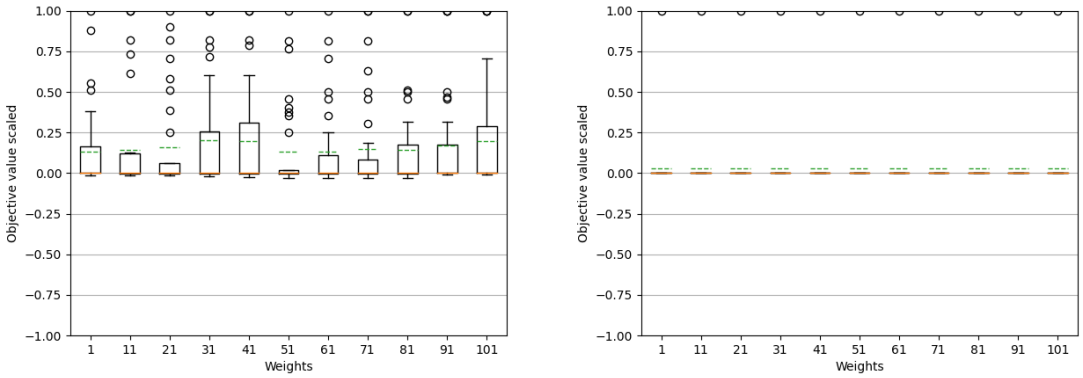
(a) wω-method. (b) ωwω-method.

Figure B.11: A histogram with the number of times the used method with the CoNoCloNo potential KPI is better (or worse) than the ω-method. The results are based on 33 instances from a benchmark set from the internet, TSPlib95.



(a) wω-method (b) ωwω-method

Figure B.12: Boxplots of the objective value of the method per weight for the CoNoCloNo potential KPI function with a random start solution and the first variant of the local search algorithm. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

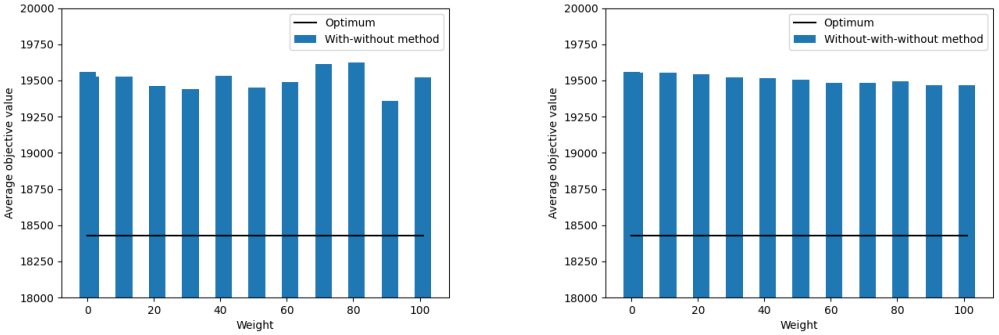


(a) wω-method (b) ωwω-method

Figure B.13: Boxplots of the objective value of the method per weight for the CoNoCloNo potential KPI function with a random start solution and the third variant of the local search algorithm. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.

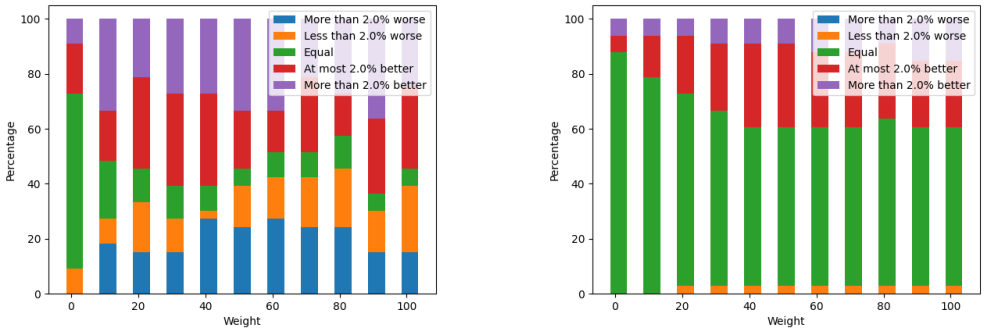
B.2.3 ConnectingNodes vs ClosestEdge

In this section, some extra figures with the CoNoCloEd function as potential KPI, that can clarify the results discussed in Section 5.5, are shown.
In the results in Section 5.5, the second variant of the local search algorithm is used, see Section 5.2 for the explanation of the algorithms. In this section, the results of the first variant are shown in similar boxplots as used for the results of the second variant of the algorithm, see Figure B.16.



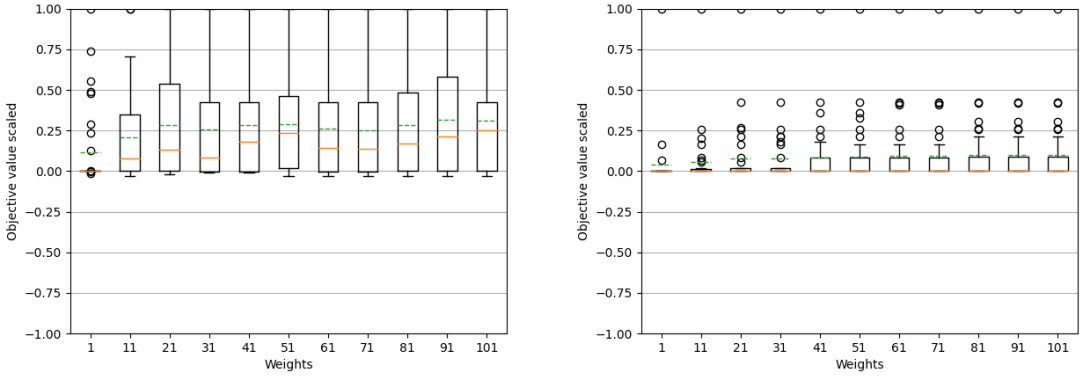
(a) wω-method. (b) ωwω-method.

Figure B.14: A histogram with the average objective values found with the local search algorithm starting from a random start solution and the CoNoCloEd potential KPI for different weights. The black line is equal to the optimum value. The results are based on 33 instances from a benchmark set from the internet, TSPlib95.



(a) wω-method. (b) ωwω-method.

Figure B.15: A histogram with the number of times the used method with the CoNoCloEd potential KPI is better (or worse) than the ω-method. The results are based on 33 instances from a benchmark set from the internet, TSPlib95.



(a) wω-method (b) ωwω-method

Figure B.16: Boxplots of the objective value of the method per weight for the CoNoCloEd potential KPI function with a random start solution and the first variant of the local search algorithm. A value of 1 means that the objective value is equal to the optimal objective value, a value of 0 means it is equal to the objective value of the method without potential KPI and a value of -1 means it is equal to the objective value of the initial solution. The green dotted line is the mean and the orange line is the median.