UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Heuristics for the Black and White Travelling Salesman Problem

Sofia Soares Henriques

Mestrado em Estatística e Investigação Operacional
Especialização em Investigação Operacional

Dissertação orientada por:
Ana Maria Duarte Silva Alves Paias

2022

Agradecimentos

Quero agradecer especialmente à professora Doutora Ana Paias por ter dedicado tempo e atenção a sugerir melhorias no meu trabalho. Foi uma honra tê-la como minha orientadora e poder aprender mais consigo.

Aos professores do Departamento de Estatística e Investigação Operacional, bem como do Departamento de Matemática, da FCUL que procuraram transferir-me conhecimento durante a minha licenciatura e mestrado. Quero dar uma atenção especial ao professor Doutor Luís Gouveia e agradecer as oportunidades que me proporcionou.

À minha mãe, Lília, e ao meu pai, Jorge, por me apoiarem desde que me lembro e por me ensinarem a acreditar no meu potencial. Agradeço também ao resto da minha família pelo carinho.

Ao Francisco, por me ter ajudado à mínima dificuldade durante a licenciatura e mestrado, e por continuamente me motivar a ser a melhor versão de mim.

Abstract

The *Black and White Travelling Salesman Problem* (BWTSP) constitutes a variant of the *Travelling Salesman Problem* (TSP). Similarly to the TSP, it is formally defined on a directed graph with a set of vertices V, a set of arcs A and each arc has an associated cost. Each vertex from V is coloured as either black or white, thus V can be partitioned into two subsets W and B, the former containing all white nodes in V and the latter containing all the black ones. The objective of the BWTSP is to determine the Hamiltonian circuit with minimal cost in the graph which satisfies two conditions: it must not contain more than Q white nodes between two consecutive black vertices, and the total length between two consecutive black vertices must not exceed a value L. The BWTSP has real-life applications in the design of telecommunication networks and in the scheduling of aircraft operations.

In the computational complexity theory, the BWTSP is classified as an NP-hard problem. This motivates the development of heuristic methods in order to obtain feasible solutions with an associated value close to the global optimum within a reasonable amount of computational time. Our goal is to develop heuristics which can be applied on both symmetric and asymmetric instances of the BWTSP, so that these methods can be used to solve more real-life problems. Due to the additional constraints of the BWTSP in comparison to the classical TSP, determining an initial feasible solution for a given instance of the problem is not a simple task, specially when the values of both parameters Q and L are tight. Therefore, we propose three constructive heuristics for the BWTSP in this dissertation.

An Iterated Local Search (ILS) algorithm was proposed as an improvement heuristic. In order to justify our choice of parameters for the algorithm, we compared the performance of the ILS considering different combinations of parameters. Furthermore, we studied in more detail the performance of the ILS we proposed and analysed the differences regarding the quality of the final solutions between symmetric and asymmetric instances.

Keywords: *Black and White Travelling Salesman Problem*, Combinatorial optimization, Heuristic methods, Iterated local search.

Resumo

O *Black and White Travelling Salesman Problem* (doravante denotado por BWTSP) constitui uma variante do Problema do Caixeiro-Viajante e, à sua semelhança, é formalmente definido num grafo orientado com um conjunto de vértices V e um conjunto de arcos A, sendo que cada arco tem um custo associado. Nesta variante, contudo, associa-se a cada vértice do grafo uma de duas cores, branco ou preto. Assim, temos que V pode ser particionado em dois subconjuntos W e B, onde o primeiro contém todos os nodos brancos de V e o último contém apenas nodos pretos. O objetivo do BWTSP é determinar um circuito Hamiltoniano no grafo considerado de modo a que não existam mais do que Q nodos brancos entre cada dois vértices pretos consecutivos e que não exista mais do que L unidades de comprimento entre cada dois nodos pretos consecutivos no mesmo circuito, sendo que se considera que os parâmetros Q e L são previamente conhecidos. Este problema possui aplicações práticas no desenho de redes de telecomunicações e no planeamento de rotas, principalmente de meios de transporte aéreos. Na teoria da complexidade computacional, o BWTSP é um problema de otimização combinatória classificado como NP-difícil. A complexidade deste problema motiva o desenvolvimento de algoritmos não-exatos que determinem uma solução admissível num intervalo de tempo razoável e que, na ausência de garantias de otimalidade, seja uma solução com um valor próximo do ótimo global.

O BWTSP não só é uma variante do Problema do Caixeiro-Viajante relativamente recente na literatura, já que o primeiro artigo conhecido que o aborda tem menos de duas décadas, como é pouco estudado no universo da Investigação Operacional. A maioria dos artigos científicos conhecidos procuram propor métodos exatos eficientes para a obtenção de um ótimo e o leque de métodos heurísticos conhecidos para o problema é muito reduzido. Adicionalmente, todas as heurísticas desenvolvidas ao longo dos anos relativas ao BWTSP apenas são aplicáveis a instâncias simétricas. Nesse sentido, este trabalho pretende dar o seu contributo ao desenvolver mais heurísticas construtivas e melhorativas para o problema. Pretende-se também que todas as heurísticas desenvolvidas no âmbito desta dissertação possam ser aplicáveis a instâncias assimétricas de modo a que possam dar resposta a mais casos reais.

Uma heurística construtiva tem como objetivo determinar uma solução inicial admissível para o problema a que é aplicada, consumindo preferencialmente a menor quantidade de tempo computacional possível. Devido às restrições adicionais que o BWTSP contém relativamente ao clássico Problema do Caixeiro-Viajante, encontrar uma solução admissível por si só não é uma tarefa simples, sobretudo quando os limites superiores de cardinalidade (Q) e comprimento (L) de uma instância são apertados. Portanto, considerou-se oportuno no âmbito desta dissertação propor três métodos heurísticos distintos para obtenção de soluções iniciais admissíveis para o BWTSP e efetuar um estudo comparativo sobre os seus respetivos tempos computacionais, a avaliação do sucesso na obtenção de soluções admissíveis e a qualidade das soluções iniciais que cada uma das três heurísticas fornece, comparativamente ao valor ótimo ou a um limite inferior do mesmo. As três heurísticas construtivas propostas nesta dissertação provêm de adaptações a algoritmos conhecidos na literatura para o Problema do Caixeiro-Viajante, em particular as heurísticas *Nearest Neighbor* (em português é conhecida como a "Heurística Do Vizinho Mais Próximo"), *Farthest Insertion* e *Random Insertion*. Após a construção de um ciclo Hamiltoniano inicial, cada um dos três algoritmos propostos contém uma fase de correção onde se procura eliminar eventuais violações das restrições de cardinalidade e de comprimento. Isto é, caso exista algum segmento entre dois nodos pretos consecutivos com mais do que Q nodos brancos ou com mais do que L unidades de comprimentos, efetuam-se pequenas alterações à solução para a tornar admissível.

Posteriormente, procurou-se desenvolver um algoritmo eficiente de Iterated Local Search (ILS) como heurística melhorativa. Este procedimento revela-se muito interessante pois permite aliar um método

de pesquisa local inicial com perturbações sucessivas, nas quais é aplicado novamente o método de pesquisa local selecionado e, assim, mais mínimos locais podem ser visitados no decorrer do algoritmo melhorativo. Para conseguir tomar partido das perturbações é necessário garantir a combinação certa de parâmetros de modo a assegurar que não se visite repetidamente o mesmo mínimo local. Uma condição importante para garantir que tal não acontece é escolher uma perturbação "forte", que neste contexto significa uma perturbação que não consegue ser facilmente desfeita pelo algoritmo de pesquisa local utilizado. A perturbação escolhida neste trabalho tem uma componente aleatória, onde $\lceil 100\omega \rceil\%$, $0 < \omega < 1$, dos nodos do ciclo Hamiltoniano considerado são selecionados aleatoriamente para serem retirados do mesmo. Efetua-se uma permutação ao conjunto de nodos selecionados e, pela mesma ordem em que estes aparecem na permutação, são inseridos na aresta do ciclo que conduz ao menor aumento de distância total. Neste contexto, o parâmetro ω apelida-se de "força da perturbação", já que à medida que o seu valor é aumentado também aumenta o número de componentes alteradas na solução. No estudo computacional, foi tomada a decisão de testar dois valores diferentes considerados razoáveis para o parâmetro ω e, para além disso, também se procurou afinar qual o número máximo adequado de iterações, denotado por $MaxIt$, para o algoritmo de ILS desenvolvido. Após uma escolha devidamente fundamentada para estes dois parâmetros, avaliou-se a eficiência da heurística de ILS.

No estudo computacional foi possível concluir que a nossa amostra de instâncias assimétricas era mais vulnerável a alterações nos parâmetros ω e $MaxIt$ do que as instâncias simétricas. Os resultados do ILS em instâncias assimétricas também se mostraram mais dependentes da qualidade da solução inicial considerada. Estas duas observações conjugadas levaram-nos a concluir que o método de pesquisa local incorporado no algoritmo de ILS tem mais impacto em instâncias simétricas do que em assimétricas, já que no primeiro conjunto de instâncias a obtenção de bons resultados não depende tanto da quantidade e qualidade das perturbações. De um modo geral, a heurística melhorativa de ILS mostra-se eficiente pois permite diminuir significativamente o valor inicial considerado num tempo computacional simpático para os dois tipos de simetria, mas há que realçar que os resultados finais tendem a ser mais próximos do ótimo no caso simétrico.

Palavras-chave: *Black and White Travelling Salesman Problem*, Otimização combinatória, Métodos heurísticos, Pesquisa local iterada.

Contents

List of Figures

List of Tables

List of Acronyms

TSP Travelling Salesman Problem

ATSP Asymmetric Travelling Salesman Problem

BWTSP Black and White Travelling Salesman Problem

ARP Aircraft Rotation Problem

VRP Vehicle Routing Problem

RATSP Asymmetric Travelling Salesman Problem with Replenishment Arcs

NN Nearest Neighbor heuristic

FI Farthest Insertion heuristic

RI Random Insertion heuristic

ANN Adapted Nearest Neighbor heuristic

AFI Adapted Farthest Insertion heuristic

ARI Adapted Random Insertion heuristic

ILS Iterated Local Search heuristic

Chapter 1

Introduction

1.1 Motivation

The Travelling Salesman Problem (TSP) is a well-known combinatorial optimization problem in the field of Operational Research. Let us consider a directed graph G, containing a set of vertices V and a set of arcs A. The objective of the TSP is to determine a tour which visits each vertex exactly once and it has minimal cost. For this purpose, a cost matrix is considered and the TSP can be classified as either symmetric or asymmetric depending on the symmetry of this matrix. Throughout the years this optimization problem has been an object of research due to its computational complexity. Not only exact and heuristic methods have been proposed to obtain an optimal, or near-optimal, solution to the classical TSP, but also to many of its variants. One of these variants is the so-called Black and White Travelling Salesman Problem (BWTSP).

The BWTSP can be seen as an extension of the TSP since both aim to obtain the cheapest Hamiltonian tour in a graph. However, in the BWTSP, the vertex set V is partitioned into two subsets: the set W containing only white vertices and set B containing only black ones. A feasible solution has to satisfy two additional conditions: the number of white vertices between each two consecutive black vertices in the tour is bounded above by a value Q, and the length of the path considered between two consecutive black vertices cannot exceed a value L. For that matter, the distance between each pair of vertices is also known.

At first, it might seem unclear to the reader what is the real-life application of this problem due to how abstract the previous description appears to be. In fact, the Aircraft Rotation Problem (ARP), which aims to determine the route flown by one single aircraft for a given airline company, can be seen as a particular case of the BWTSP applied to airline operations. Let us assume the set of arcs A in the graph G is a set of flight legs and the set of vertices V is a set of connection points, or stations. The reason why designing a route for a single airplane is so important to the airline company is mainly due to the economic benefits that arise from assigning a big number of flight legs to one single vehicle, explained by the "through-value" associated. A "through-value" was defined by Clarke et al. [1997] as *"the revenue that would be expected to be gained from additional passengers who would be attracted to the service because of being able to stay on the same airplane rather than having to change airplanes at the stopover point"*. However, each vehicle must not fly more than a maximum number of L hours and it must not sequence more than $Q + 1$ flight legs (arcs in the graph) without maintenance. Unfortunately, not all connection points (vertices in the graph) are able to do the required maintenance. The nodes in V which represent hubs suitable to do the maintenance of an aircraft are the black nodes present in set $B \subset V$. To schedule a sequence of flights to an aircraft satisfying all the safety constraints is to solve an instance of the BWTSP. Applications can be found not only in short-haul airline operations as just described, but also in telecommunications. When designing fiber networks, it is considered an undirected graph, which means there is a set of edges instead of arcs between each pair of nodes. The objective is to find the shortest Hamiltonian cycle, knowing the white vertices represent standard hubs and the black ones are ring offices. The cycle needs to be determined in a way that ensures the distance between two consecutive ring offices does not exceed the value L, and the number of hubs between two consecutive ring offices must not be superior to Q.

Particularly, if we consider an instance of the BWTSP where all the black vertices are positioned exactly in the same place, an instance of the Vehicle Routing Problem (VRP) is obtained. In this scenario, the objective is to find $\lceil |W|/Q \rceil$ simple circuits, knowing that each white vertix represents a client with a unit demand, each vehicle has a capacity of Q units and the length of each route must not exceed L.

1.2 Literature review

The "Black and White Travelling Salesman Problem" name was firstly used by Bourgeois et al. [2003], even though many previously studied problems could be modeled as the BWTSP (to exemplify, see Wasem [1991], Clarke et al. [1997], Talluri [1998] and Mak and Boland [2000]). In particular, Mak and Boland [2000] developed and tested a Simulated Annealing heuristic for the Asymmetric Travelling Salesman Problem with Replenishment Arcs (RATSP), which is closely related to the BWTSP. Let us consider a directed graph $G = (V, A)$, where V is a set of nodes, A is a set of arcs, $C = \{c_{ij} : (i, j) \in A\}$ is the cost matrix and each node $i \in V$ has a weigth w_i. The arc set is partitioned into new subsets: the set $\mathcal{R}$ containing replenishment arcs and $\mathcal{A} = A \setminus \mathcal{R}$ containing ordinary arcs. The objective of the RATSP is to determine the Hamiltonian circuit with minimal cost. However, a circuit is only feasible if it does not accumulate more than W units of weigth before using a new replenishment arc.

Bourgeois et al. [2003] created three different constructive heuristics to determine a feasible solution for a given instance of the symmetric BWTSP, and then a 2-opt procedure applied only to feasible solutions is chosen as an improvement heuristic. In this article, different values of $|B|$, Q and L were considered, as well as several levels of dispersion of the black vertices (i.e., choosing black nodes closer to the center of the square, or choosing positions further from the center). The procedures were tested on instances with up to 200 vertices. The conducted tests using all of the three constructive heuristics showed that few feasible solutions were determined not only when the values of the parameters Q and L were tighter but also when the level of dispersion of the black nodes was lower (i.e., when the black nodes were positioned closer to each other).

More than a decade later another heuristic approach for the symmetric BWTSP was proposed by Li and Alidaee [2016], who developed a Tabu Search heuristic and integrated a Branch & Cut algorithm in it. Even though more computational time was required to solve instances with this hybrid algorithm, feasible solutions very close to the global optimum were found for instances with a considerable number of vertices and edges, knowing that the largest instance had 439 vertices. An interesting remark about the performance of this hybrid approach is that, in instances with a maximum of 100 nodes, it tends to obtain a feasible solution with a smaller cost when compared to the Tabu Search heuristic or the Branch & Cut algorithm alone.

Regarding exact methods, Ghiani et al. [2006] modeled the problem, considering an undirected graph as well, by only associating a binary variable to each edge of the edge set E. Many non-trivial constraints can be found in the formulation, which motivates the authors to prove all of them are valid inequalities for the problem. Furthermore, an exact Branch & Cut algorithm is proposed and tested in instances with up to 100 vertices. On the other hand, İbrahim Muter [2015] proposed an extended formulation for the BWTSP in which, in addition to the binary variables considered before, each white node has a set of associated variables referencing the black nodes visited before and after. The constraints in this model are significantly less than the number of constraints in the model proposed by Ghiani et al. [2006], however more variables are considered. For every pair of black vertices $(b_i, b_j) \in B^2$ $(i \neq j)$ a block structure is found in the formulation, and İbrahim Muter [2015] takes advantage of this fact to apply the Dantzig-Wolfe decomposition to the problem. The column-generation algorithm is embedded in a Branch & Price algorithm, and was tested on instances with up to 80 vertices.

Gouveia et al. [2017] explored more extended formulations for the BWTSP. An initial formulation was presented by associating variables to each arc disaggregated by black vertices. Position-dependent and distance-dependent formulations are also studied, as well as a position-and-distance-dependent reformulation. Valid inequalities are introduced to the different models in order to improve the LP bounds. Furthermore, Branch & Cut algorithms were implemented and tested on a set of instances generated previously by İbrahim Muter [2015]. The authors also created two sets of randomly generated instances: the first set of instances was supposed to have a bigger dispersion of the black vertices, in which the first black node was selected randomly and a new black node was selected iteratively and it had to maximize the minimal distance to the previously constructed set of black vertices; in the second set the closest vertices to the center of the square were chosen to be coloured black. A similar observation to Bourgeois et al. [2003] was made since the authors noted that instances with black nodes closer to each other were harder to solve. The distance and the cost of each edge on both sets of randomly generated instances in Gouveia et al. [2017] are set independently from one another; which means that for every edge $e \in E$ its distance d_e and its cost c_e are not necessarily equal, whereas all the previously mentioned articles consider $d_e = c_e, \forall e \in E$.

1.3 Scientific contribution and dissertation overview

Due to the computational complexity of the BWTSP, finding the optimal solution for an instance of the problem with a larger set of nodes and edges/arcs becomes very time-consuming. It is important to develop algorithms which allow us to determine a feasible solution with a value in the objective function close to the global minimum in a reasonable amount of time. As mentioned before, few heuristic approaches have been proposed for the problem, and the existing methods do not consider asymmetric distances or costs between vertices.

We intend to expand the existing heuristic methods available for the BWTSP through this work and we want to make sure that the developed algorithms can be applied on asymmetric matrices in order to tackle even more real-life problems. Within the scope of this dissertation, distances and costs of arcs will be considered to be the same, since most of the literature related to the problem adopted a similar assumption.

On the current Chapter, our goal was to provide some context to the reader about what the BWTSP consists of, why it is an interesting problem to study and what are its real-world applications. Furthermore, we summarized the scientific contributions over the last two decades in regards to the problem. On Chapter 2, we will formally define the BWTSP, address its computational complexity and present an Integer Linear Programming model for the problem. On Chapter 3, three constructive heuristics are proposed: we detail the selection and insertion methods of each algorithm, as well as the mechanisms we developed to attempt to establish feasibility for the resulting solutions. On Chapter 4, we propose an Iterated Local Search (ILS) algorithm as an improvement heuristic, where we detail the Local Search procedure embedded in it as well as the perturbation method we adopted. Finally, on Chapter 5 we develop a computational study in which, firstly, we compare the performance of the three constructive heuristics and, secondly, we justify our choices regarding the perturbation method of the ILS and the maximum number of iterations of the entire algorithm by trying 6 different combinations of parameters, leading us to the conclusion that our choice matches our purpose for the ILS heuristic. For every instance we considered during the computational study, we used the feasible solutions provided by each one of the three constructive heuristics as the initial solution for the ILS in order to analyse the differences in their respective performances. We also studied the differences in performance of the ILS for symmetric and asymmetric instances of the BWTSP.

Chapter 2

The Black and White Travelling Salesman Problem

The Travelling Salesman Problem (TSP) can be defined on a directed graph $G = (V, A)$, where V is a set of nodes and A is a set of arcs. If an undirected graph is considered with a set of edges E, a directed graph can be easily obtained by assigning both directions to each edge; which means that for every edge $\{i, j\} \in E$ two arcs $(i, j) \in A$ and $(j, i) \in A$ are created. A distance matrix $D = \{d_{i,j} : (i, j) \in A\}$ containing the travelling time of each arc in the graph is considered. The objective of this problem is to determine the shortest Hamiltonian circuit in G. Note that a circuit is classified as Hamiltonian if all the vertices in V are visited exactly once during the tour.

The Black and White Travelling Salesman Problem (BWTSP) is an extension of the TSP. Each node $i \in V$ is coloured as either black or white and the set of vertices is then partitioned into new subsets W and B, in which the former contains all the white vertices in the graph and the latter contains all the black vertices. The goal is to find the Hamiltonian circuit in the graph with minimal cost satisfying two conditions:

(i) The number of white nodes between two consecutive black nodes in the circuit cannot exceed $Q \in \mathbb{N}$;

(ii) The total distance between two consecutive black nodes in the circuit cannot be more than $L \in \mathbb{R}$.

In fact, the BWTSP reduces to the TSP considering $Q = L = +\infty$. Since the TSP is classified as an NP-hard problem according to the computational complexity theory [Garey and Johnson, 1979], then so is the BWTSP.

If a Hamiltonian circuit s is determined on the graph G, it only constitutes a feasible solution for an instance of the BWTSP if it satisfies all the **cardinality constraints** (condition (i)) and all the **length constraints** (condition (ii)).

Given an instance for the BWTSP, let us consider a Hamiltonian circuit s. In this case, s can be divided into $|B|$ segments starting from a black node to the next one in the tour. In short, we will denote this kind of segments as **path segments**. All the path segments will be named after their first black node: if a path starts with the black node $b \in B$, then it will be denoted as $\mathcal{P}_b$ to simbolize "the path segment of node b". If one of the path segments of a Hamiltonian circuit violates cardinality constraints then it is **cardinality infeasible**; if, on the other hand, it violates length constraints then it is **length infeasible** - in both scenarios the path segment is infeasible. If at least one of the path segments of a circuit is infeasible, then the entire circuit is infeasible for the given instance of the problem.

To illustrate this concept, let G be a directed graph with a set of nodes $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $A = \{(i, j) \in V^2 : i, j \in V \wedge i \neq j\}$. Let us consider the set $B = \{1, 3, 5\}$, $W = V \setminus B$ and the Hamiltonian circuit $s = \{(1, 3), (3, 2), (2, 4), (4, 6), (6, 5), (5, 7), (7, 8), (8, 1)\}$, which is represented in figure 2.1. This circuit has three path segments: $\mathcal{P}_1 = \{(1, 3)\}$, $\mathcal{P}_3 = \{(3, 2), (2, 4), (4, 6), (6, 5)\}$ and $\mathcal{P}_5 = \{(5, 7), (7, 8), (8, 1)\}$.

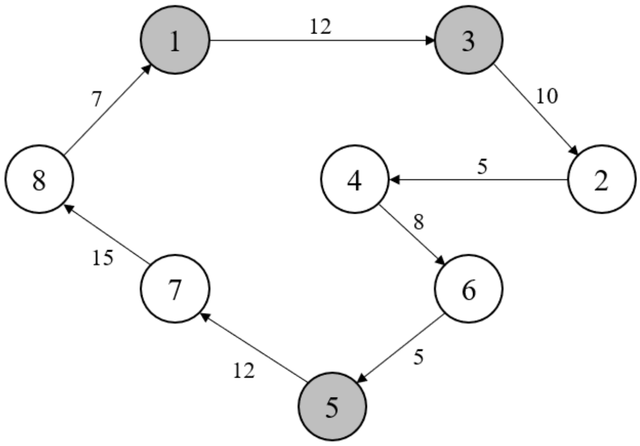

Figure 2.1: Example of a Hamiltonian circuit.

For $Q \geq 3$ and $L \geq 34$, s constitutes a feasible solution. If $Q = 2$, for example, then the path segment $\mathcal{P}_3$ becomes cardinality infeasible; if L is set to less than 34, then the path $\mathcal{P}_5$ becomes length infeasible. On both scenarios, the overall circuit is infeasible.

2.1 Problem formulation

In this section, an Integer Linear Programming model for the directed BWTSP will be presented and its structure can be summarized as follows:

1. A formulation for the Asymmetric Travelling Salesman Problem (ATSP);

2. Further addition of cardinality and length constraints to the previous model.

To model the ATSP, a distance matrix (equivalently, cost matrix) is considered for a graph $G = (V, A)$, and a binary variable x_{ij} is associated to each arc $(i, j) \in A$. The first set of variables can be defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is included in the Hamiltonian circuit} \\ 0 & \text{otherwise} \end{cases} \qquad \forall (i, j) \in A$$

In order to guarantee the connectivity of the Hamiltonian tour, the improved version of the Miller-Tucker-Zemlin (MTZ) sub-tour elimination contraints, proposed by Desrochers and Laporte [1991], will be incorporated. Non-negative integer variables u_i will be associated to each node i in the vertex set V to identify its position in the circuit. An extended formulation for the ATSP can be obtained as follows:

$$\begin{aligned}
\text{Min} \quad & \sum_{(i,j)\in A} d_{ij}x_{ij} && && (2.1)\\
\text{s.a.:} \quad & \sum_{i\in V} x_{ij} = 1 && \forall j \in V && (2.2)\\
& \sum_{i\in V} x_{ji} = 1 && \forall j \in V && (2.3)\\
& u_i - u_j + (|V|-1)x_{ij} + (|V|-3)x_{ji} \leq |V|-2 && \forall (i,j) \in A, i,j \in \{2,...,n\} && (2.4)\\
& u_i \geq 1 && \forall i \in \{2,...,n\} && (2.5)\\
& u_i \leq n-1 && \forall i \in \{2,...,n\} && (2.6)\\
& u_i \geq 0 && \forall i \in V && (2.7)\\
& x_{ij} \in \{0,1\} && \forall (i,j) \in A && (2.8)
\end{aligned}$$

The objective of this problem is to minimize the total travelled distance which is established by (2.1). (2.2) and (2.3) are degree constraints, which aim to assure that each vertex in the set V is visited once, and only once, during the tour. Constraints (2.4)-(2.6) are the strengthened MTZ sub-tour elimination constraints. All the variables are non-negative, which is guaranteed by (2.7) and (2.8). Since the set of variables $\mathbf{x} = \{x_{ij} : (i,j) \in A\}$ is set to be binary, it becomes unnecessary to specify that the set of variables $\mathbf{u} = \{u_i : i \in V\}$ only assumes integer values.

The previous formulation will be taken as the basis of the model for the BWTSP. To incorporate the additional constraints of the problem, the path segment model proposed in Gouveia et al. [2017] will be considered. The authors chose to formulate the problem by identifying all the existent path segments on a Hamiltonian tour. As mentioned before, the path that starts with the black vertex $b \in B$ will be denoted as $\mathcal{P}_b$. For each arc $(i,j) \in A$, new variables y_{ij}^k, $\forall k \in B$, will be created. This new set of variables can be defined as follows:

$$y_{ij}^k = \begin{cases} 1 & \text{if } (i,j) \text{ is included in } \mathcal{P}_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall (i,j) \in A, \forall k \in B$$

The BWTSP can be formulated as follows:

$$\begin{aligned}
\text{Min} \quad & \sum_{(i,j)\in A} d_{ij}x_{ij} && && (2.9)\\
\text{s.a.:} \quad & (2.2) - (2.8) && && \\
& \sum_{k\in B} y_{ij}^k = x_{ij} && \forall (i,j) \in A && (2.10)\\
& \sum_{j\in V} y_{kj}^k = 1 && \forall k \in B && (2.11)\\
& \sum_{j\in B\setminus\{k\}} \sum_{i\in V} y_{ik}^j = 1 && \forall k \in B && (2.12)\\
& \sum_{j\in V} y_{ji}^k = \sum_{j\in V} y_{ij}^k && \forall k \in B, \forall i \in W && (2.13)
\end{aligned}$$

$$\sum_{(i,j)\in A} y_{ij}^k \leq Q+1 \qquad \forall k \in B \tag{2.14}$$

$$\sum_{(i,j)\in A} d_{ij} y_{ij}^k \leq L \qquad \forall k \in B \tag{2.15}$$

$$y_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in B \tag{2.16}$$

(2.10) is a set of linking constraints between the set of variables $\mathbf{y} = \{y_{ij}^k : (i,j) \in A \wedge k \in B\}$ and $\mathbf{x} = \{x_{ij} : (i,j) \in A\}$, and it tries to establish that a given arc $(i,j) \in A$ cannot belong to any path segment if it isn't used in the circuit. Constraints (2.11) ensure that for every black node in the graph there is one selected outgoing arc. By the previous definion of a path segment every black node is, not only the first visited vertex in its own path, but also the last to be visited in the path segment of another black node. The set of constraints (2.12) aims to make sure that it occurs in any feasible solution to the problem. Another set of logical constraints is (2.13) and it ensures both the ingoing and outgoing arc of every white node to belong to the same path segment.

Set (2.14) contains all the cardinality constraints: for a generic path segment in the Hamiltonian circuit P_k, with $k \in B$, all the visited vertices are white except for the first and the last one; meaning that to restrict the number of white nodes in $\mathcal{P}_k$ to Q or less is equivalent to restrict the total number of arcs in the path to $Q+1$ or less. Length constraints are represented in (2.15), by summing the distances of all arcs in a given path segment and bounding this value to L. Finally, (2.16) is included in the model to make sure that all the variables in the set $\mathbf{y} = \{y_{ij}^k : (i,j) \in A \wedge k \in B\}$ are binary.

Chapter 3

Constructive heuristics for the BWTSP

The main purpose of a constructive heuristic is to obtain an initial feasible solution for a given problem. The resulting solution might not hold the best overall value, which motivates the development of a further improvement heuristic.

Many constructive heuristics for the TSP have been studied, being the Nearest Neighbor (NN) one of the most intuitive and well-known algorithms. Essentially, a Hamiltonian tour is obtained by iteratively adding the arc connecting the last visited node to its nearest neighbor. Another important constructive heuristic is the Farthest Insertion (FI), which starts with the longest tour containing only two vertices and iteratively chooses the node maximizing the minimal distance to the nodes of the current tour. The selected node is inserted in the position which leads to the minimal increase of total cost. It is particularly interesting to study these two heuristics since they differ in the selection method of the next vertex to insert in the current tour at each step and their results tend to be quite different. The FI procedure seems to select vertices in a very counterintuitive manner, unlike the NN; however, its resultant solution tends to hold a value closer to the global optimum when compared to the NN (see Rosenkrantz et al. [1977]).

Since the Black and White variant of the TSP imposes additional restrictions to the original problem, determining an initial feasible Hamiltonian circuit satisfying all constraints is expected to be a harder task. This becomes more evident when the proportion of black nodes in the graph is smaller and the values of the parameters Q and L are tighter, due to the reduced pool of feasible solutions. Therefore, every constructive heuristic we developed in this dissertation can be divided in two separate "stages":

1. **Construction stage:** A Hamiltonian circuit is iteratively constructed, which means that a nonvisited vertex in the graph is selected and inserted in the tour in each iteration. This process stops when all the vertices in the graph have been already visited by the tour.

2. **Correction stage:** Cardinality and length feasibility of the previous Hamiltonian circuit is evaluated. If this solution happens to be infeasible, the tour will go throught attempts to establish cardinality and/or length feasibility (see Section 3.4).

Every constructive heuristic within the scope of this dissertation is going to share the same procedures in the correction stage, thus they only differ amongst each other in the construction stage. Initially two deterministic heuristics will be proposed. Both are going to be adaptations of the NN and the FI heuristics with alterations on their selection methods. We will denote these heuristics as Adapted Nearest Neighbor (ANN) and Adapted Farthest Insertion (AFI), relatively, and they are detailed on Sections 3.1 and 3.2. Furthermore, a randomized constructive heuristic is proposed (see Section 3.3) as an adaptation of the Random Insertion heuristic of the TSP and we named it the Adapted Random Insertion (ARI) heuristic.

3.1 Adapted Nearest Neighbor heuristic

The Nearest Neighbor (NN) heuristic aims to determine a Hamiltonian tour for a given TSP instance, defined on a directed graph $G = (V, A)$, following the steps:

Step 1 Select an initial node. Let it be denoted by a^*. Let $\mathcal{I}$ be the set of visited nodes, then set $\mathcal{I} = \{a^*\}$;

Step 2 Select a node b^* in the set $V \setminus \{a^*\}$ such that $d_{a^*,b^*} = \min\limits_{b \in V \setminus \{a^*\}} \{d_{a^*,b}\}$. Include arc $(a^*, b^*) \in A$ in the circuit and do $\mathcal{I} = \mathcal{I} \cup \{b^*\}$;

Step 3 Let x^* be the last inserted vertex. Choose a node $y^* \in V \setminus \mathcal{I}$ for which $d_{x^*,y^*} = \min\limits_{y \in V \setminus \mathcal{I}} \{d_{x^*,y}\}$ and add arc $(x^*, y^*) \in A$ to the tour. Do $\mathcal{I} = \mathcal{I} \cup \{y^*\}$. If $V \setminus \mathcal{I} \neq \emptyset$, repeat *Step 3*; otherwise, go to *Step 4*;

Step 4 Add the arc connecting the last visited node to a^*. **STOP**.

On the BWTSP, the vertex set V is partitioned into new subsets B and W, the former containing all the black nodes and the latter containing all the white nodes. Determining an initial feasible solution for the BWTSP through the original NN can be a hard task, because there might be a tendency, depending on the distance matrix, to obtain a circuit which visits all the white nodes first and then all the black ones, for example, which is unbalanced. The Adapted Nearest Neighbor (ANN) heuristic is different from the original NN because it forces the algorithm to search for the nearest black vertex if Q white nodes have been already visited since the last black one. However, it only happens if there are still vertices in B left to include in the tour. This greedy search mechanism tries to ensure that at least the first $|B| - 1$ path segments are cardinality feasible. Length feasibility is not guaranteed for any of the built path segments of the tour. The suggested procedure is detailed in **Algorithm 1**, and vertex 1 will be chosen as the first visited node. The notation on the pseudocode is described as follows:

- $Tour \rightarrow$ The current circuit in every iteration of the algorithm, which is represented as a set of visited arcs;
- $Cost \rightarrow$ The cost of the current tour;
- $LastNode/NextNode \rightarrow$ The last/next visited vertex;
- $\mathcal{I} \rightarrow$ set of already included vertices in the $Tour$;
- $n_b \rightarrow$ Number of black vertices already included in the tour;
- $n_w \rightarrow$ Number of white vertices visited since the last visited black node.

Algorithm 1 Adapted Nearest Neighbor Heuristic

```
Tour ← {}
Cost ← 0
LastNode ← 1
I ← {1}
n_b ← 1
n_w ← 0
```

while $V \setminus \mathcal{I} \neq \emptyset$ **do**
    if $n_w = Q$ and $n_b < |B|$ **then**
        $NextNode \leftarrow \arg\min_{b \in B \setminus \mathcal{I}} \{d_{(LastNode,b)}\}$
    else
        $NextNode \leftarrow \arg\min_{v \in V \setminus \mathcal{I}} \{d_{(LastNode,v)}\}$
    end if
    $NextArc \leftarrow (LastNode, NextNode)$
    $Tour \leftarrow Tour \cup \{NextArc\}$
    $Cost \leftarrow Cost + d_{NextArc}$
    if $NextNode \in W$ **then**
        $n_w \leftarrow n_w + 1$
    else
        $n_b \leftarrow n_b + 1$
        $n_w \leftarrow 0$
    end if
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{NextNode\}$
    $LastNode \leftarrow NextNode$
end while
$Tour \leftarrow Tour \cup \{(LastNode, 1)\}$
$Cost \leftarrow Cost + d_{(LastNode,1)}$
$Tour \leftarrow$ ESTABLISHFEASIBILITY($Tour$) ▷ detailed in Section 3.4
Update $Cost$

3.2 Adapted Farthest Insertion heuristic

The Farthest Insertion (FI) heuristic aims to determine a Hamiltonian tour for a given TSP instance, defined on a directed graph $G = (V, A)$, following the steps:

Step 1 Select two initial nodes $a^*, b^* \in V$ such that $d_{a^*,b^*} + d_{b^*,a^*} = \max_{a,b \in V, a \neq b} \{d_{a,b} + d_{b,a}\}$. The initial tour only contains the arcs connecting these two selected vertices, $(a^*, b^*) \in A$ and $(b^*, a^*) \in A$. Create the set of nodes already included in the circuit $\mathcal{I} = \{a^*, b^*\}$;

Step 2 Calculate the minimal distance of every vertex in $V \setminus \mathcal{I}$ to the tour. The distance of a vertex $i \in V \setminus \mathcal{I}$ to the tour is calculated as follows:

$$d_{i,\text{Tour}} = \min_{v \in \mathcal{I}} \{\max\{d_{i,v}, d_{v,i}\}\} \qquad \forall i \in V \setminus \mathcal{I} \tag{3.1}$$

Note that an asymmetric distance matrix can be considered and, for a pair of vertices $i \in V$ and $j \in V \setminus \{i\}$, the distance between i and j is not necessarily equal to the distance from j to i; therefore, we will consider the maximum between these two values as the distance between i and j. Go to *Step 3*;

Step 3 After determining the distance of every vertex not included in the set $\mathcal{I}$ to the tour, choose the node x^* such that $d_{x^*,\text{Tour}} = \max_{x \in V \setminus \mathcal{I}} \{d_{x,\text{Tour}}\}$. Select an arc (i^*, j^*) included in the circuit such that $d_{i^*,x^*} + d_{x^*,j^*} - d_{i^*,j^*} = \min_{(i,j) \in \text{Tour}} \{d_{i,x^*} + d_{x^*,j} - d_{i,j}\}$. Replace arc (i^*, j^*) with arcs (i^*, x^*) and (x^*, j^*) in the circuit. Do $\mathcal{I} = \mathcal{I} \cup \{x^*\}$. As long as $V \setminus \mathcal{I} \neq \emptyset$, repeat *Step 2*; otherwise, **STOP**.

In order to apply this heuristic to the BWTSP, the selection method will be slightly altered. Initially, a circuit containing only black vertices will be determined according to the selection and insertion methods of the FI; which means, that *Steps 1, 2* and *3* of the FI heuristic are executed by exclusively considering the set of vertices B, instead of V. When all black vertices are included in the circuit, *Steps 2* and *3* of the FI heuristic are repeated for white nodes. This version of the FI will be called the Adapted Farthest Insertion (AFI).

In conclusion, the only difference between the original FI and the AFI heuristic is that on the first algorithm all vertices in V are candidates to be inserted at any stage as long as they have not been visited yet; on the second algorithm, the selection pool is limited to black vertices until all of them are included in the tour, only then we consider inserting white nodes. The proposed AFI heuristic is detailed in **Algorithm 2**. The notation on the pseudocode is described as follows:

- $Tour$ $\rightarrow$ The current circuit in every iteration of the algorithm, which is represented as a set of visited arcs;
- $Cost$ $\rightarrow$ The cost of the current tour;
- $\mathcal{I}$ $\rightarrow$ set of already included vertices in the $Tour$;
- $next$ $\rightarrow$ The selected vertex to be inserted in the current iteration of the algorithm;
- (i^*, j^*) $\rightarrow$ Selected arc to insert node $next$.

Algorithm 2 Adapted Farthest Insertion Heuristic

$Cost \leftarrow 0$
for all $(i, j) \in B^2$ such that $i \neq j$ **do**
  if $d_{i,j} + d_{j,i} > Cost$ **then**
    $Cost \leftarrow d_{i,j} + d_{j,i}$
    $Tour \leftarrow \{(i, j), (j, i)\}$
    $\mathcal{I} \leftarrow \{i, j\}$
  end if
end for
while $B \setminus \mathcal{I} \neq \emptyset$ **do**
  $next \leftarrow \arg \max_{b \in B \setminus \mathcal{I}} \{\min_{i \in \mathcal{I}} \{\max\{d_{i,b}, d_{b,i}\}\}\}$
  $(i^*, j^*) \leftarrow \arg \min_{(i,j) \in Tour} \{d_{i,next} + d_{next,j} - d_{i,j}\}$
  Replace arc (i^*, j^*) in the $Tour$ with arcs $(i^*, next)$ and $(next, j^*)$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{next\}$
  $Cost \leftarrow Cost + d_{i^*,next} + d_{next,j^*} - d_{i^*,j^*}$
end while
while $V \setminus \mathcal{I} \neq \emptyset$ **do**
  $next \leftarrow \arg \max_{v \in V \setminus \mathcal{I}} \{\min_{i \in \mathcal{I}} \{\max\{d_{i,v}, d_{v,i}\}\}\}$
  $(i^*, j^*) \leftarrow \arg \min_{(i,j) \in Tour} \{d_{i,next} + d_{next,j} - d_{i,j}\}$
  Replace arc (i^*, j^*) in the $Tour$ with arcs $(i^*, next)$ and $(next, j^*)$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{next\}$
  $Cost \leftarrow Cost + d_{i^*,next} + d_{next,j^*} - d_{i^*,j^*}$
end while

$Tour \leftarrow$ ESTABLISHFEASIBILITY($Tour$) ▷ detailed in Section 3.4
Update $Cost$

3.3 Adapted Random Insertion heuristic

The Random Insertion (RI) heuristic aims to determine a Hamiltonian tour for a given TSP instance, defined on a directed graph $G = (V, A)$, following the steps:

Step 1 Consider a random permutation of the set of nodes $\{1, ..., |V|\}$. Let i_1 and i_2 be the first two elements of the permutation. Consider the initial tour $\{(i_1, i_2), (i_2, i_1)\}$. Do $k = 3$ and go to *Step 2*;

Step 2 Choose the next node, i_k, in the permutation. Select an arc (x^*, y^*) included in the circuit such that $d_{x^*,i_k} + d_{i_k,y^*} - d_{x^*,y^*} = \min\limits_{(x,y)\in \text{Tour}} \{d_{x,i_k} + d_{i_k,y} - d_{x,y}\}$. Replace arc (x^*, y^*) with arcs (x^*, i_k) and (i_k, y^*) in the circuit. As long as $k < |V|$, set $k = k + 1$ and repeat *Step 2*; otherwise, **STOP**.

Similarly to the AFI procedure, the Random Insertion heuristic is going to be adapted to the BWTSP by organizing, in the first place, all of the black nodes in the tour and only then white notes will be selected. This adaptation will be denoted as Adapted Random Insertion (ARI). In this context, two different permutations are considered: one permutation to the set of black nodes B and the other one to the set of white nodes W. The proposed ARI heuristic is detailed in **Algorithm 3**. The notation on the pseudocode is described as follows:

- $Tour$ $\rightarrow$ The current circuit in every iteration of the algorithm, which is represented as a set of visited arcs;
- $Cost$ $\rightarrow$ The cost of the current tour;
- $\mathcal{B}^p$ $(\mathcal{W}^p)$ $\rightarrow$ Permutation of black (white) nodes;
- $\mathcal{B}^p(x)\big(\mathcal{W}^p(x)\big)$ $\rightarrow$ The x-ith element in the permutation $\mathcal{B}^p$ $(\mathcal{W}^p)$.

Algorithm 3 Adapted Random Insertion Heuristic

$Tour \leftarrow \{(\mathcal{B}^p(1), \mathcal{B}^p(2)), (\mathcal{B}^p(2), \mathcal{B}^p(1))\}$
$Cost \leftarrow d_{\mathcal{B}^p(1),\mathcal{B}^p(2)} + d_{\mathcal{B}^p(2),\mathcal{B}^p(1)}$
$x \leftarrow 2$
while $x < |B|$ **do**
    $x \leftarrow x + 1$
    $(i^*, j^*) \leftarrow \arg\min\limits_{(i,j)\in Tour} \{d_{i,\mathcal{B}^p(x)} + d_{\mathcal{B}^p(x),j} - d_{i,j}\}$
    Replace arc (i^*, j^*) in the $Tour$ with arcs $(i^*, \mathcal{B}^p(x))$ and $(\mathcal{B}^p(x), j^*)$
    $Cost \leftarrow Cost + d_{i^*,\mathcal{B}^p(x)} + d_{\mathcal{B}^p(x),j^*} - d_{i^*,j^*}$
end while
$x \leftarrow 0$
while $x < |W|$ **do**
    $x \leftarrow x + 1$
    $(i^*, j^*) \leftarrow \arg\min\limits_{(i,j)\in Tour} \{d_{i,\mathcal{W}^p(x)} + d_{\mathcal{W}^p(x),j} - d_{i,j}\}$

Replace arc (i^*, j^*) in the $Tour$ with arcs $(i^*, \mathcal{W}^p(x))$ and $(\mathcal{W}^p(x), j^*)$
$Cost \leftarrow Cost + d_{i^*,\mathcal{W}^p(x)} + d_{\mathcal{W}^p(x),j^*} - d_{i^*,j^*}$
end while
$Tour \leftarrow$ ESTABLISHFEASIBILITY($Tour$) ▷ detailed in Section 3.4
Update $Cost$

3.4 Establishing feasibility

Mechanisms with the purpose of correcting both cardinality and length infeasibilities of a Hamiltonian circuit will be developed in this section. A similar approach to Li and Alidaee [2016] will be taken as the establishment of cardinality feasibility, if unexistent, is done before the attempts to restore length feasibility.

Algorithm 4 Procedure to establish feasibility
Require: a Hamiltonian circuit, denoted by $Tour$
function ESTABLISHFEASIBILITY($Tour$)
  if $Tour$ is cardinality infeasible **then**
    $Tour \leftarrow$ CARDINALITYCORRECTION($Tour$) ▷ detailed in Subsection 3.4.1
  end if
  if $Tour$ is length infeasible **then**
    $Tour \leftarrow$ LENGTHCORRECTION($Tour$) ▷ detailed in Subsection 3.4.2
  end if
  return $Tour$
end function

In the following subsections, we will detail the procedures we used to attempt establishing cardinality and length feasibility for a given BWTSP solution.

3.4.1 Correcting cardinality infeasibility

Restoring cardinality feasibility in a Hamiltonian circuit is always possible as long as $|W| \leq Q \times |B|$, considering Q to be the maximum number of white nodes in each path segment, B and W the set of black and white vertices, respectively. This statement can be easily proven by taking into consideration that there are as many path segments in a circuit as there are black vertices in a graph (since each black node $b \in B$ determines the beginning of a new path segment $\mathcal{P}_b$). According to a "worst case scenario", if $|W| = Q|B|$, then Q white nodes can be distributed along all the $|B|$ path segments, making the solution cardinality feasible. If there is at least one more white node in the graph, then at least one of the path segments has to have $Q + 1$ white vertices or more.

Cardinality feasibility can be established by removing a white vertex $w \in W$ from a path segment with more than Q white vertices and by reinserting it in another path with $Q - 1$ white vertices or less. Now, it is important to define which white node will be removed and what is the right position to place it. Note that after restoring cardinality feasibility, the resulting circuit will go through an attempt to restore length feasibility (which means that each path segment must have a final total length of L or less). During the procedure to fix cardinality infeasibility, we can try to facilitate length feasibility by ensuring that the removed white vertex translates into the maximal saving of length in all infeasible paths. Following the same mindset, we should aim to insert the removed white vertex in the position in the cardinality feasible path which minimizes the additional length associated with it.

The proposed algorithm to correct cardinality infeasibility is detailed in **Algorithm 5**, and it uses the following notation:

- $MaxRemoval \rightarrow$ maximum length saving by removing a white vertex from a cardinality infeasible path segment;
- $w^r \rightarrow$ white node selected to be removed;
- $P^r \rightarrow$ path segment that includes w^r;
- $prev(w^r), succ(w^r) \rightarrow$ vertex which precedes and succeeds, respectively, w^r in path P^r;
- $MinInsert \rightarrow$ minimum length increase, of all cardinality feasible path segments, by reinserting node w^r;
- $(x^i, y^i) \rightarrow$ selected arc to insert node w^r;
- $P^i \rightarrow$ path segment that includes (x^i, y^i);
- $\mathcal{Q}(*) \rightarrow$ number of white nodes in the path $*$;
- $\mathcal{L}(*) \rightarrow$ total length of the path $*$.

Algorithm 5 Cardinality infeasibility correction

Require: a Hamiltonian circuit, denoted by $Tour$

function CARDINALITYCORRECTION($Tour$)
    while $Tour$ is cardinality infeasible **do**
        $MaxRemoval \leftarrow 0$
        for all $\mathcal{P}_b$ with $\mathcal{Q}(\mathcal{P}_b) > Q$ **do**
            for all $w \in W$ in the path $\mathcal{P}_b$ **do**
                if $d_{prev(w),w} + d_{w,succ(w)} - d_{prev(w),succ(w)} > MaxRemoval$ **then**
                    $MaxRemoval \leftarrow d_{prev(w),w} + d_{w,succ(w)} - d_{prev(w),succ(w)}$
                    $P^r \leftarrow \mathcal{P}_b$
                    $w^r \leftarrow w$
                end if
            end for
        end for
        $MinInsert \leftarrow +\infty$
        for all $\mathcal{P}_b$ with $\mathcal{Q}(\mathcal{P}_b) \leq Q - 1$ **do**
            for all $(x, y) \in A$ in path $\mathcal{P}_b$ **do**
                if $d_{x,w^r} + d_{w^r,y} - d_{x,y} < MinInsert$ **then**
                    $MinInsert \leftarrow d_{x,w^r} + d_{w^r,y} - d_{x,y}$
                    $P^i \leftarrow \mathcal{P}_b$
                    $(x^i, y^i) \leftarrow (x, y)$
                end if
            end for
        end for
        Replace arcs $(prev(w^r), w^r)$ and $(w^r, succ(w^r))$ with the arc $(prev(w^r), succ(w^r))$ in path P^r
        Replace the arc (x^i, y^i) with arcs (x^i, w^r) and (w^r, y^i) in path P^i

$\mathcal{Q}(P^r) \leftarrow \mathcal{Q}(P^r) - 1$
$\mathcal{Q}(P^i) \leftarrow \mathcal{Q}(P^i) + 1$
Update lengths $\mathcal{L}(P^r)$ and $\mathcal{L}(P^i)$
end while
return $Tour$
end function

3.4.2 Correcting length infeasibility

After correcting cardinality infeasibility, length feasibility should also be established in the circuit. Both the selection and the insertion methods will be similar to the ones present in the cardinality infeasibility correction procedure; the only difference is that now a black node is also a candidate to be removed from its path and to be reinserted in another position. The selected vertex is inserted on the same path in a new position or in another one with strictly less than Q white nodes (to make sure that all paths remain cardinality feasible).

A 2-exchange move exclusive to the path $\mathcal{P}_b$, for $b \in B$, is defined as a path which can de obtained from $\mathcal{P}_b$ by replacing, at most, 2 edges. If the distance matrix is symmetric, then all arcs in the path can be viewed as edges, since arcs (i, j) and (j, i) share the same distance $d_{i,j}$. After the repositioning of a node in the circuit, if the distance matrix is symmetric, for every path segment violating length contraints (i.e., with a total length exceeding the value L) it will be determined the 2-exchange applied to that specific path segment resulting in the minimal final length. However, in the context of the BWTSP, we will restrict a 2-exchange move to be considered feasible if the black vertices remain in their original positions; which means that the first (last) black vertex remains in the beginning (end, respectively) of the path. This additional constraint is important to make sure that other path segments in the tour do not go through length changes during this procedure. In order to exemplify this concept, a path segment with three white vertices is presented and all its feasible 2-exchanges are illustrated in figure 3.1.

When figure 3.1 is analysed, it is evident that a 2-exchange also consists of reversing a subpath of the original path segment. Since both the first and last black nodes of a path are forced in advance to remain in the exact same position, then in this particular case a 2-exchange consists of reversing a subpath which only includes white vertices. Once again, this search procedure will only be performed when a symmetric distance matrix is considered. If it happens to be asymmetric, then it would be required to recalculate the cost of the reversed subpath (since all the included arcs would be reversed and their costs are not guaranteed to be the same), thus this process would demand a lot of computational effort.

The proposed algorithm to correct length infeasibility is detailed in **Algorithm 6**, and it uses the following notation:

- $n_{it} \rightarrow$ number of attempts to correct length infeasibility;
- $MaxRemoval \rightarrow$ maximal saving of length by removing a white vertex from a cardinality infeasible path segment;
- $w^r \rightarrow$ white node selected to be removed;
- $P^r \rightarrow$ path segment that includes w^r;
- $prev(w^r), succ(w^r) \rightarrow$ vertex which precedes and succeeds, respectively, w^r in path P^r;
- $MinInsert \rightarrow$ minimal increased length, of all cardinality feasible path segments, by reinserting node w^r;

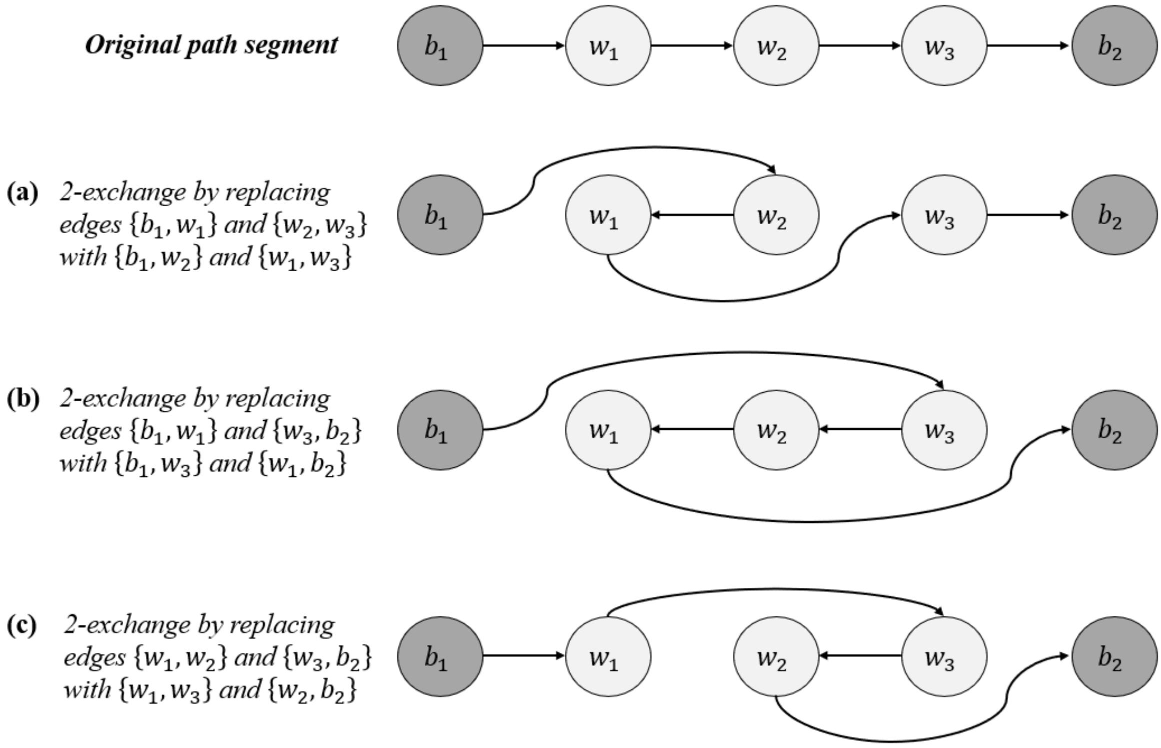

Figure 3.1: Example of a path segment $\mathcal{P}$ with three white vertices and all the feasible 2-exchange moves on $\mathcal{P}$, which are different from $\mathcal{P}$.

- $(x^i, y^i) \rightarrow$ selected arc to insert node w^r;
- $P^i \rightarrow$ path segment that includes (x^i, y^i);
- $\mathcal{Q}(*) \rightarrow$ number of white nodes in the path $*$;
- $\mathcal{L}(*) \rightarrow$ total length of the path $*$.

Algorithm 6 Length infeasibility correction

Require: a cardinality feasible Hamiltonian circuit denoted by $Tour$ and the total number of nodes n

function LENGTHCORRECTION($Tour$)
  $n_{it} \leftarrow 1$
  while $Tour$ is length infeasible and $n_{it} < 10ln(n)$ **do**
    $MaxRemoval \leftarrow 0$
    for all $\mathcal{P}_b$ with $\mathcal{L}(\mathcal{P}_b) > L$ **do**
      for all node $v \in \mathcal{P}_b$ **do**
        if $d_{prev(v),v} + d_{v,succ(v)} - d_{prev(v),succ(v)} > MaxRemoval$ **then**
          $MaxRemoval \leftarrow d_{prev(v),v} + d_{v,succ(v)} - d_{prev(v),succ(v)}$
          $P^r \leftarrow \mathcal{P}_b$
          $v^r \leftarrow v$
        end if
      end for
    end for

$MinInsert \leftarrow +\infty$
for all $\mathcal{P}_b$, with $\mathcal{L}(\mathcal{P}_b) \leq L$ and $\mathcal{Q}(\mathcal{P}_b) \leq Q-1$, and for the path P^r **do**
  for all $(x, y) \in A$ in the current path $\mathcal{P}$ **do**
    if $d_{x,v^r} + d_{v^r,y} - d_{x,y} < MinInsert$ **then**
      $MinInsert \leftarrow d_{x,v^r} + d_{v^r,y} - d_{x,y}$
      $P^i \leftarrow \mathcal{P}$
      $(x^i, y^i) \leftarrow (x, y)$
    end if
  end for
end for
if cardinality feasibility of the circuit is not violated **then**
  Replace arcs $(prev(v^r), v^r)$ and $(v^r, succ(v^r))$ with the arc $(prev(v^r), succ(v^r))$ in P^r
  Replace the arc (x^i, y^i) with arcs (x^i, v^r) and (v^r, y^i) in P^i
  Update the lengths of every path segment that went through changes
end if
if the distance matrix is symmetric **then**
  for all $\mathcal{P}_b$ with $\mathcal{L}(\mathcal{P}_b) > L$ and $\mathcal{Q}(\mathcal{P}_b) > 1$ **do**
    $\mathcal{P}_b \leftarrow$ 2-EXCHANGE($\mathcal{P}_b$) ▷ defined on Algorithm 7
    Update length $\mathcal{L}(\mathcal{P}^b)$
  end for
end if
$n_{it} \leftarrow n_{it} + 1$
end while
end function

There are some BWTSP instances to which we cannot determine a length feasible Hamiltonian circuit by doing all the procedures we incorporated in **Algorithm 6**. Therefore, we have decided to restrict the maximum number of attempts to correct length feasible in order to avoid an infinite loop. This maximum was defined as a logarithmic function of the number of vertices in the graph, denoted by n.

After selecting a node v^r to be removed in path P^r and reinserting it in the path P^i (P^i and P^r can be the same), there is no guarantee that only these two paths suffer length changes (or, this individual path, if they're the same). Thus, it is important to note that whenever v^r happens to be a black node in P^r, then the path which precedes or succeeds P^r also needs an update on its cardinality and length; mainly because a black vertex determines the beginning and the end of two consecutive path segments.

A feasible 2-exchange move on a path segment $\mathcal{P}_b$ is equivalent to reverse a subpath of $\mathcal{P}_b$ between two white nodes, since both black vertices need to remain in the same position. Let us denote w_i^b as the white node in the i-th position after the first black node b in the path $\mathcal{P}_b$. Reversing a path between w_i^b and w_j^b, with $j \geq i+1$, means to remove arcs $\left(prev(w_i^b), w_i^b\right)$ and $\left(w_j^b, succ(w_j^b)\right)$ in order to insert $\left(prev(w_i^b), w_j^b\right)$ and $\left(w_i^b, succ(w_j^b)\right)$. All the arcs between vertices w_i^b and w_j^b are reversed and the cost of this reversed path is not changed, because we are assuming a symmetric distance matrix.

We detail in **Algorithm 7** how to determine the 2-exchange move with minimum cost on a path segment $\mathcal{P}_b$, for $b \in B$. The pseudocode contains the following notation:

- $MinAddLength \rightarrow$ minimal increased length by reversing a path in $\mathcal{P}_b$. If it is equal to 0, then the current current length of $\mathcal{P}_b$ is minimal; if it is less than 0, then there is a 2-exchange move on $\mathcal{P}_b$ which results in a final lower length;
- $\mathcal{Q}\left(\mathcal{P}_b\right) \rightarrow$ number of white nodes in the path segment $\mathcal{P}_b$;

- $w_i^b \rightarrow$ the i^{th} white node in the path segment $\mathcal{P}_b$;
- $prev(w_i^b), succ(w_i^b) \rightarrow$ vertex which precedes and succeeds, respectively, w_i^b in path P^b.

Algorithm 7 Determining the best 2-exchange move on a path $\mathcal{P}_b$

function 2-EXCHANGE($\mathcal{P}_b$)
  $MinAddLength \leftarrow 0$
  for all $i \in \{1, ..., \mathcal{Q}(\mathcal{P}_b) - 1\}$ **do**
    for all $j \in \{i+1, ..., \mathcal{Q}(\mathcal{P}_b)\}$ **do**
      if $d_{prev(w_i^b),w_j^b} + d_{w_i^b,succ(w_j^b)} - d_{prev(w_i^b),w_i^b} - d_{w_j^b,succ(w_j^b)} < MinAddLength$ **then**
        $MinAddLength \leftarrow d_{prev(w_i^b),w_j^b} + d_{w_i^b,succ(w_j^b)} - d_{prev(w_i^b),w_i^b} - d_{w_j^b,succ(w_j^b)}$
        $i^* \leftarrow i$
        $j^* \leftarrow j$
      end if
    end for
  end for
  if $MinAddLength < 0$ **then**
    Remove arcs $\left(prev(w_{i^*}^b), w_{i^*}^b\right)$ and $\left(w_{j^*}^b, succ(w_{j^*}^b)\right)$
    for all $k \in \{i^*, ..., j^* - 1\}$ **do**
      Replace arc (w_k^b, w_{k+1}^b) with the arc (w_{k+1}^b, w_k^b) in path $\mathcal{P}_b$
    end for
    Insert arcs $\left(prev(w_{i^*}^b), w_{j^*}^b\right)$ and $\left(w_{i^*}^b, succ(w_{j^*}^b)\right)$ in path $\mathcal{P}^b$
  end if
  return $\mathcal{P}_b$
end function

3.5 Final observations

If an infeasible instance of the BWTSP is considered, it is evident that none of the three constructive heuristics we proposed return feasible solutions for it. On the other hand, *even* when a feasible instance of the BWTSP is considered, it is possible that at least one of the three constructive heuristics does not determine a feasible heuristic solution.

Essentially, the procedure we proposed to correct cardinality infeasibility removes a white vertex from a path segment with more than Q white nodes and reinserts it in a path with $Q - 1$ white nodes or even less. This reinsertion is independent from whether the selected paths are length feasible or not. If $|W| \leq Q|B|$, then it is always possible to determine a solution with a maximum of Q white nodes in each path segment, which means that cardinality feasibility is always guaranteed under these circunstances, as mentioned before. When cardinality feasibility of the Hamiltonian circuit is accomplished, the algorithm proceeds to correct length infeasiblity when necessary. But now, a condition needs to be satisfied: the circuit must remain cardinality feasible. This condition leads us to limit the number of possible alterations in the circuit so that cardinality feasibility is not compromised throughout the procedure. To sum up, whenever a feasible instance is considered and one of the constructive heuristics cannot reach a feasible solution, it is mostly due to the restricted variety of moves on the vertices of a circuit when we try to correct length infeasiblity.

The underlying limitation of the correction stage accentuates the importance of having a good construction method to begin with in every constructive heuristic. Three construction methods were proposed by adapting already existing heuristics for the original TSP: Nearest Neighbor, Farthest Insertion and Random Insertion. On further computational tests in this dissertation, the “success rate” of each one of these constructive heuristics should also be evaluated. The “success rate” of a constructive heuristic equates to the proportion of feasible BWTSP instances to which that same heuristic was able to determine a feasible solution.

Chapter 4

Iterated Local Search heuristic

The previous Chapter of this dissertation focused on the development of methods to determine initial feasible solutions for the BWTSP. On the current Chapter, we will start with the premise that such solution is already known, it is denoted by s_0 and its value in the objective function of the BWTSP is $\mathcal{V}(s_0)$. Our goal is to develop an efficient Iterated Local Search (ILS) heuristic in order to return a solution s with value $\mathcal{V}(s)$ close to the global minimum. The development of an ILS procedure as an improvement heuristic is motivated by the possibility of evaluating a wider range of solutions in the feasibility set. This is done by applying perturbations to a solution obtained through a Local Search procedure, which leads us to visit more than one local minimum and thus we increase our chances of achieving the global optimum. We refer to Lourenço et al. [2003] for a detailed explanation of the ILS.

Starting from a feasible solution s_0, a Local Search procedure is applied to it which results in a local minimum s_0^*. A perturbation is applied to s_0^* and a new solution s_1 is obtained. If s_1 happens to be feasible, we apply the same Local Search procedure to s_1 which results in a new local minimum s_1^*, hopefully different from s_0^*. At this point in the algorithm, two local minimums have been determined: s_0^* and s_1^*. For example, let us assume $\mathcal{V}(s_0^*) < \mathcal{V}(s_1^*)$. When we are going to apply a second perturbation, we might wonder whether we should choose the solution with the lowest cost (the incumbent solution s_0^*) or the solution we obtained after the last time we applied the Local Search procedure (the current solution s_1^*). It might be tempting to apply the perturbation to the incumbent solution: since it is the best local minimum we have found so far, it is normal to wonder if a slight change in it does not lead to a new solution with lower cost. If we choose this option, we risk repeated perturbations if many iterations go by and the incumbent solution remains the same. Therefore, it can be interesting to apply perturbations on the current solution, even if it is not the local minimum with the lowest cost so far, because it can bring more diversity to our search. We have decided that every 5 iterations the perturbation is applied to the incumbent solution, whereas on the remaining iterations the perturbation is applied to the current solution.

The proposed ILS procedure is detailed in **Algorithm 8**. The notation on the pseudocode is described as follows:

- $N_{it} \rightarrow$ Total number of iterations of the algorithm;
- $n_{it}^{change} \rightarrow$ Number of iterations since the last update of the incumbent solution (iterations without improvement).
- $s \rightarrow$ The latest solution obtained through Local Search in the current iteration;
- $s^* \rightarrow$ The incumbent solution in the current iteration;

Algorithm 8 Iterated Local Search procedure

Require: An initial feasible solution s_0 for the BWTSP instance

```
N_it ← 0
n_it^change ← 0
s ← LocalSearch(s_0)
s* ← s
while N_it < 2500 do
    N_it ← N_it + 1
    if N_it is divisible by 5 then
        p ← Perturbation(s*)
    else
        p ← Perturbation(s)
    end if
    if p is a feasible solution for the BWTSP instance then
        s ← LocalSearch(p)
        if V(s) < V(s*) then
            s* ← s
            n_it^change ← 0
        else
            n_it^change ← n_it^change + 1
        end if
    else
        n_it^change ← n_it^change + 1
    end if
    if n_it^change > 250 and N_it is divisible by 5 then
        break
    end if
end while
return s* and V(s*)
```

The maximum number of iterations of the algorithm, let it be denoted by $MaxIt$, is set to 2500. Furthermore, if more than 10% of $MaxIt$ iterations go by without any improvement on the incumbent solution we stop the algorithm. In this particular case, if we have more than 250 consecutive iterations without improvement. Additionally, we decided to only allow the algorithm to stop prematurely if the last perturbation was applied to the incumbent solution. Since every 5 iterations the perturbation method is applied to the incumbent solution, we need to verify if the number of the current iteration is divisible by 5.

Let the outcome of a perturbation be a Hamiltonian circuit denoted by p. We must evaluate whether or not it is feasible for the given BWTSP instance. If the answer is affirmative, then it is a valid initial solution for the Local Search procedure. When it happens to be infeasible, the Local Search procedure is not applied to the solution p and we consider one more iteration without improvement. Both the Local Search procedure and the perturbation method are going to be defined on the following sections of this chapter.

4.1 Neighborhoods and Local Search procedure

On this Section, we will define the Local Search procedure we incorporated in the ILS algorithm. First, it is important to define all the neighborhoods for the BWTSP which are going to be needed later on.

As mentioned before, a feasible solution for an instance of the BWTSP contains black and white vertices, and each black vertex determines the beginning of a path segment in the circuit. Let us consider a feasible solution s for a given instance of the BWTSP. Four neighborhoods can be defined as follows:

$$\mathcal{N}_B(s) = \{s' : s' \text{ is a feasible solution which can be obtained by switching a maximum of two black nodes in } s\}$$

$$\mathcal{N}_{interW}(s) = \{s' : s' \text{ is a feasible solution which can be obtained by switching a maximum of two white nodes in different path segments in } s\}$$

$$\mathcal{N}_{intraW}(s) = \{s' : s' \text{ is a feasible solution which can be obtained by switching in each path segment of } s \text{ a maximum of two white vertices } \}$$

$$\mathcal{N}_{paths}(s) = \{s' : s' \text{ is a feasible solution which can be obtained by switching a maximum of two path segments in } s\}$$

Neighborhoods $\mathcal{N}_B(s)$ and $\mathcal{N}_{interW}(s)$ include s and every other feasible circuit which can be obtained by switching the positions of two nodes in s with the same color, on the first case both nodes need to be black and on the second case both nodes need to be white and belong to different path segments.

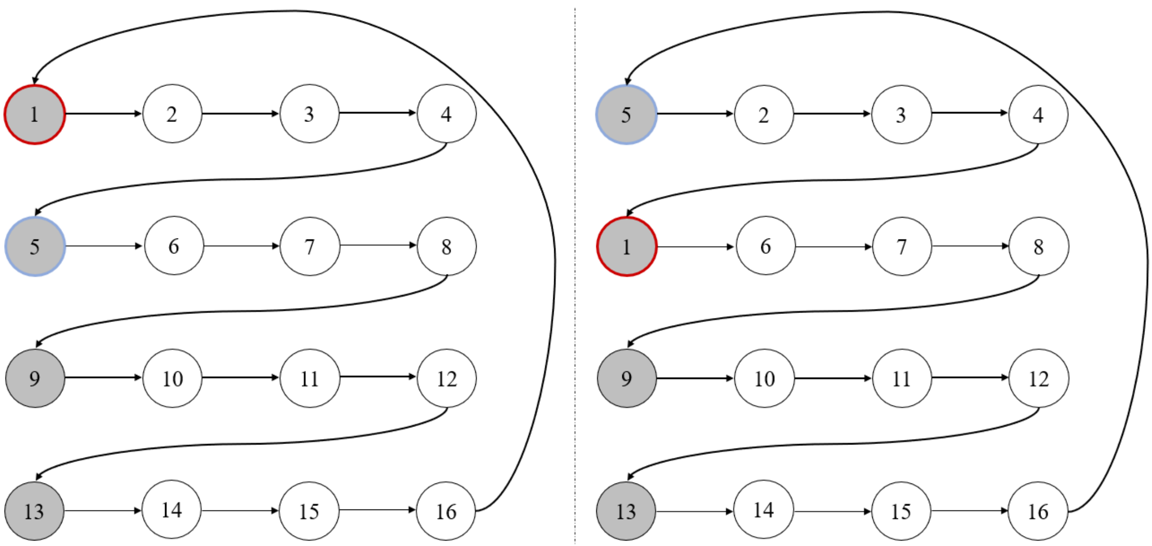

Figure 4.1: On the left, a feasible solution s for an instance of the BWTSP. On the right, an example of a solution in $\mathcal{N}_B(s)$.

Swapping the positions of two white nodes inside the same path segment of s will only be allowed in neighborhood $\mathcal{N}_{intraW}(s)$. This neighborhood includes s and any other feasible solution for the BWTSP instance which can be obtained by switching the position of two white nodes in one or more path segments of s. On figure 4.3 it is possible to see an example of this neighborhood, where we consider a solution s with sixteen vertices distributed along four path segments: $\mathcal{P}_1$, $\mathcal{P}_5$, $\mathcal{P}_9$ and $\mathcal{P}_{13}$.

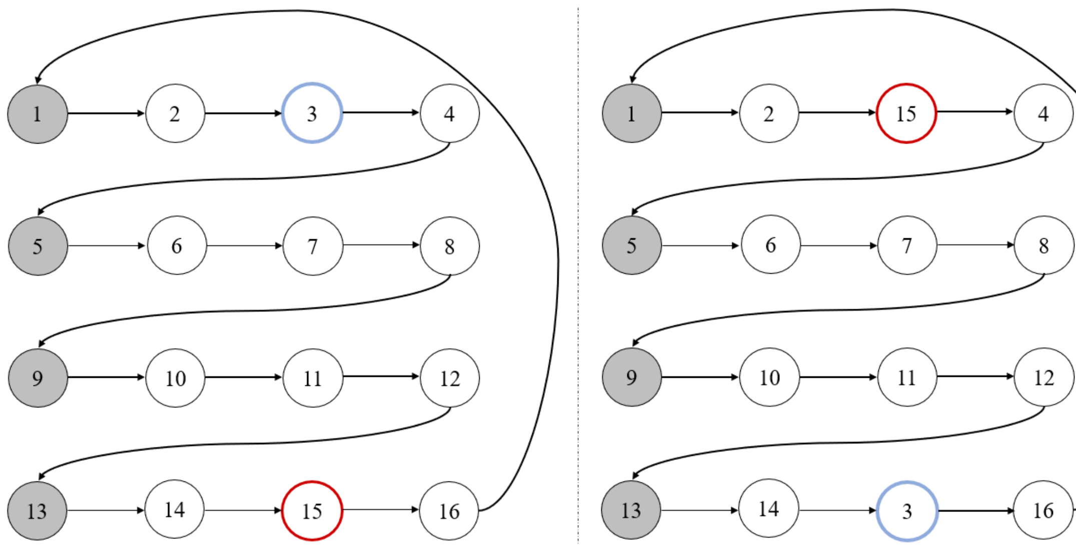

Figure 4.2: On the left, a feasible solution s for an instance of the BWTSP. On the right, an example of a solution in $\mathcal{N}_{interW}(s)$.

A different solution from s, let us denote it by s', was obtained by switching the positions of two white nodes in the paths $\mathcal{P}_5$ and $\mathcal{P}_{13}$. The same thing could occur for both $\mathcal{P}_1$ and $\mathcal{P}_9$ but, once again, it is just an example which aims to illustrate the range of different solutions from s contained in $\mathcal{N}_{intraW}(s)$.

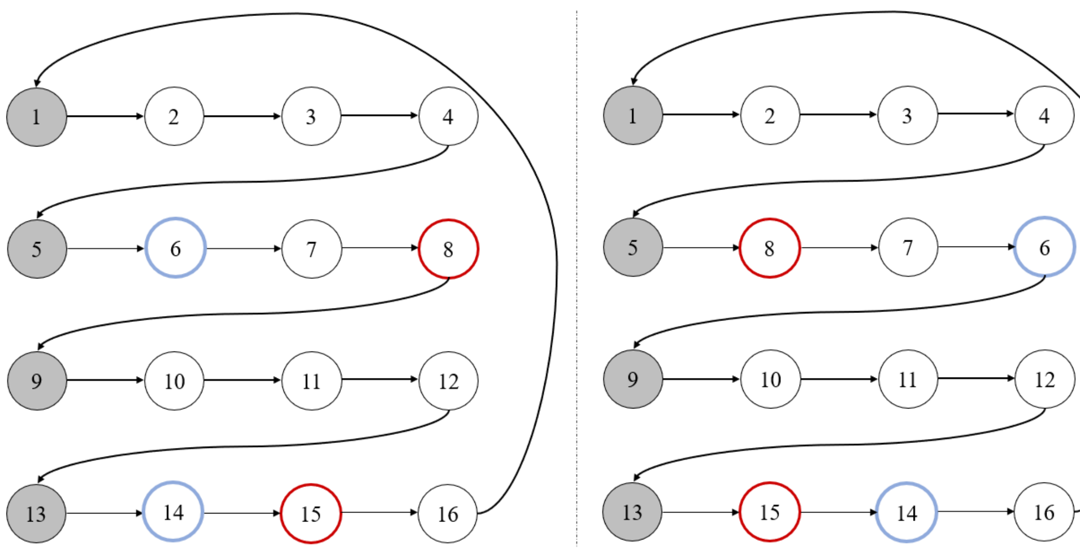

Figure 4.3: On the left, a feasible solution s for an instance of the BWTSP. On the right, an example of a solution in $\mathcal{N}_{intraW}(s)$.

Neighborhood $\mathcal{N}_{paths}(s)$ contains s and any other feasible solution for the BWTSP instance which can be obtained from s by swapping the positions of two path segments in s. This neighborhood is exemplified in figure 4.4, where we consider the same solution s. A different solution from s, let us denote it by s', was obtained by swapping the positions of paths $\mathcal{P}_5$ and $\mathcal{P}_{13}$. It only makes sense to search this neighborhood if s contains more than two path segments, which means it can only be used in instances with more than two black vertices. Note that when s has exactly two black vertices, then s is itself the only solution in $\mathcal{N}_{paths}(s)$.

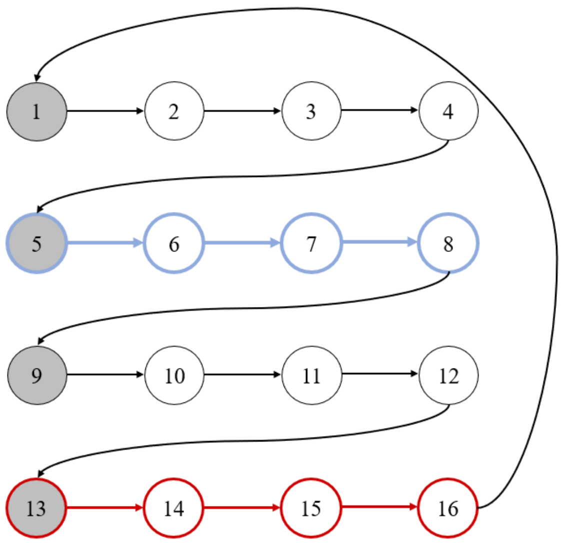

Figure 4.4: On the left, a feasible solution s for an instance of the BWTSP. On the right, an example of a solution in $\mathcal{N}_{paths}(s)$.

A fifth neighborhood will be searched, but only for symmetric instances:

$$\mathcal{N}_{2-exchange}(s) = \{s' : s' \text{ is a feasible solution for the BWTSP which can be obtained by reversing, at most, one subpath in } s\}$$

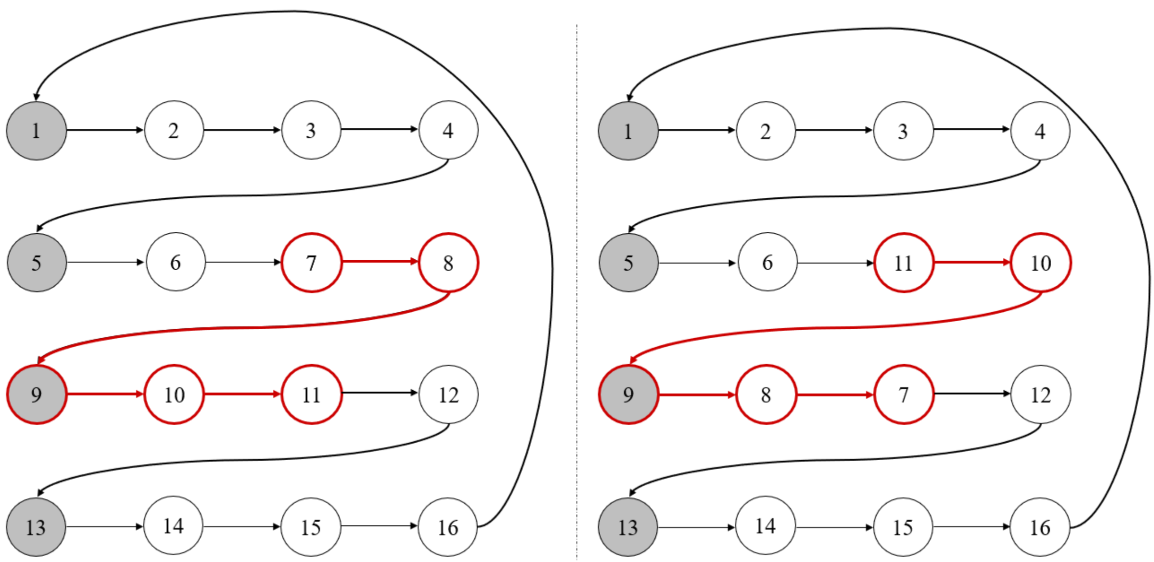

Figure 4.5: On the left, a feasible solution s for an instance of the BWTSP. On the right, an example of a solution in $\mathcal{N}_{2-exchange}(s)$.

This neighborhood is typically used for the original TSP whenever a symmetric distance matrix is considered. It would be possible to apply it on asymmetric instances, however the cost of the reversed subpath would have to be recalculated which would demand a lot of computational effort. The same does not occur for symmetric distance matrices because the cost of every reversed arc is the same, since $d_{i,j} = d_{j,i}, \forall (i,j) \in A$; thus, the cost of the reversed subpath remains unaltered.

Any previously defined neighborhood will be searched following the method which is represented in **Algorithm 9** for a generic neighborhood $\mathcal{N}$ and an initial feasible solution s_0 for the BWTSP instance. On the pseudocode, $\mathcal{V}(*)$ denotes the value of a solution $*$ in the objective function of the problem.

Algorithm 9 Search in neighborhood $\mathcal{N}(s_0)$

Require: A feasible solution s_0 for the BWTSP instance

$s \leftarrow s_0$
while stopping condition is not met **do**
    if there is a solution $s' \in \mathcal{N}(s)$ different from s with $\mathcal{V}(s') < \mathcal{V}(s)$ **then**
        $s \leftarrow s'$
    else
        break
    end if
end while
return s and $\mathcal{V}(s)$

Being $\mathcal{N}$ one of the five previously defined neighborhoods, it is important to note that when we look for a solution $s' \in \mathcal{N}(s)$ which satisfies $\mathcal{V}(s') < \mathcal{V}(s)$ we do not require it to be the solution in $\mathcal{N}(s)$ with the minimal cost. The algorithm stops when a feasible solution s for the BWTSP instance is determined such that $\mathcal{V}(s) = \min\limits_{s' \in \mathcal{N}(s)} \{\mathcal{V}(s')\}$.

The basic tools for the Local Search procedure have been established: neighborhoods have been defined and the adopted search method within each one of these neighborhoods has also been presented. The overall procedure is represented on **Algorithm 10**, which contains the same notation as the previous pseudocode.

Algorithm 10 Local Search procedure

Require: A feasible solution s for the BWTSP instance

function LOCALSEARCH(s)
    while stopping condition is not met **do**
        $InitialCost \leftarrow \mathcal{V}(s)$
        Search neighborhood $\mathcal{N}_B(s)$ and obtain s'
        if $s' \neq s$ **then**
            $s \leftarrow s'$
        end if
        Search neighborhood $\mathcal{N}_{intraW}(s)$ and obtain s'
        if $s' \neq s$ **then**
            $s \leftarrow s'$
        end if
        Search neighborhood $\mathcal{N}_{interW}(s)$ and obtain s'
        if $s' \neq s$ **then**
            $s \leftarrow s'$
        end if

if $|B| > 2$ **then**
Search neighborhood $\mathcal{N}_{paths}(s)$ and obtain s'
if $s' \neq s$ **then**
$s \leftarrow s'$
end if
end if
if the distance matrix is symmetric **then**
Search neighborhood $\mathcal{N}_{2-exchange}(s)$ and obtain s'
if $s' \neq s$ **then**
$s \leftarrow s'$
end if
end if
$NewCost \leftarrow \mathcal{V}(s)$
if $NewCost = InitialCost$ **then**
break
end if
end while
return s and $\mathcal{V}(s)$
end function

4.2 Perturbation method

Choosing the appropriate perturbation method for an ILS algorithm is not an easy task. On one hand, the perturbation should be impactful enough to escape from a local optimum; on the other hand, if it changes entirely the solution we risk a random restart in every iteration of the ILS. We should take into consideration the Local Search procedure we decide to use, because a perturbation applied upon a solution s must not be easily undone by the Local Search procedure and lead to s again.

We have decided to randomize our perturbation method. Let us consider a solution s for an instance of the BWTSP with n vertices. A total of $\lceil \omega n \rceil$ vertices out of n, with $0 < \omega < 1$, will be randomly chosen to be removed from the Hamiltonian circuit s. If a node k, $1 \leq k \leq n$, is selected to be removed from s, then we remove arcs $(prev(k), k)$ and $(k, succ(k))$ and insert $(prev(k), succ(k))$, where $prev(k)$ and $succ(k)$ denote the vertex which precedes and succeeds k, respectively, in the circuit s. We apply a permutation on the set of removed vertices and we will denote it by $\mathcal{R}^p$. Successively, we insert each vertex of the permutation $\mathcal{R}^p$ (by order) in the position of s which leads to the minimal cost of insertion. Once this process is over we attempt to establish the feasiblity of the resulting circuit, following the detailed procedure in Section 3.4. The resultant circuit is considered to be the result of the perturbation and is denoted by p.

The parameter ω is the strength of the permutation since it controls the portion of vertices in a Hamiltonian circuit s which are going to be removed and reinserted in s. We are going to set $\omega = 1/2$. The pseudocode of the perturbation method is present in **Algorithm 11** and it contains the notation:

- $\omega \rightarrow$ strength of the permutation (portion of vertices from the initial solution which are going to be removed and reinserted);
- $\mathcal{R} \rightarrow$ Set of removed vertices from the initial solution
- $\mathcal{R}^p \rightarrow$ Permutation of $\mathcal{R}$
- $\mathcal{R}^p(i) \rightarrow$ The i-th element of $\mathcal{R}^p$

Algorithm 11 Perturbation method

Require: A feasible solution s for the BWTSP instance with n vertices

function PERTURBATION(s)
  $\mathcal{R} \leftarrow \{\}$
  $\omega \leftarrow 1/2$
  while $|\mathcal{R}| < \lceil \omega n \rceil$ **do**
    Choose a random vertex k^* in s
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{k^*\}$
    Replace arcs $(prev(k^*), k^*)$ and $(k^*, succ(k^*))$ with $(prev(k^*), succ(k^*))$ in s
  end while
  Apply a permutation to $\mathcal{R}$ and denote it by $\mathcal{R}^p$
  for $i \in \{1, ..., \lceil wn \rceil\}$ **do**
    $MinInsCost \leftarrow +\infty$
    for $(x, y) \in s$ **do**
      if $d_{x,\mathcal{R}^p(i)} + d_{\mathcal{R}^p(i),y} - d_{x,y} < MinInsCost$ **then**
        $MinInsCost \leftarrow d_{x,\mathcal{R}^p(i)} + d_{\mathcal{R}^p(i),y} - d_{x,y}$
        $(x^*, y^*) \leftarrow (x, y)$
      end if
    end for
    Replace the arc (x^*, y^*) with arcs $(x^*, \mathcal{R}^p(i))$ and $(\mathcal{R}^p(i), y^*)$ in s
  end for
  $p \leftarrow$ ESTABLISHFEASIBILITY(s) ▷ detailed in Section 3.4
  return p
end function

4.3 Final observations

Our computational study is detailed on Chapter 5. Two parameters were tested for the ILS heuristic: the strength of the perturbation, denoted by ω, and the maximum number of iterations of the algorithm, $MaxIt$. On one hand, the parameter ω controls the proportion of nodes to remove randomly from a given Hamiltonian circuit s, in order to reinsert them in the position which leads to the minimal increase of cost for the circuit s (detailed in Section 4.2). On the other hand, the parameter $MaxIt$ represents the maximum number of iterations of the ILS heuristic and it also regulates when to prematurely terminate the algorithm.

We tested three options for $MaxIt$, 1000, 2500 and 5000, meaning that the ILS algorithm stops after 100, 250 and 500 iterations, relatively, without improvement of the incumbent value. It is more complex to decide which values for ω are reasonable, but we decided to consider ω as either $1/3$ or $1/2$. Reorganizing less than 33% of the nodes in a circuit when a perturbation is being applied seemed to be useless because we are not changing many componentes in the solution. If we decide to randomly remove more than 50% of the nodes just to reinsert them later on in the best position, then we risk a random restart of the full solution.

Therefore, we tested 6 different combinations of the parameters $(\omega, MaxIt)$: for both $\omega = 1/3$ and $\omega = 1/2$ we used the three different values for the maximum number of iterations. Our goal is to justify why we chose to set the parameter ω as $1/2$ and the maximum number of iterations to 2500. Furthermore, we will display a summary of the computational results we obtained for this combination of parameters.

Chapter 5

Computational study

Tests were conducted to evaluate the performance of the proposed heuristics. This chapter begins with a detailed explanation on how to determine all the test instances used within the scope of this dissertation and their respective optimal values. Then we compare all the previously developed constructive heuristics regarding their success on obtaining feasible solutions for BWTSP instances, their computational times and we discuss advantages and disadvantages of each procedure. Finally, we test different combinations of parameters for the ILS heuristic to justify our choices in the previous chapter and we proceed to summarize the obtained results for our choice of parameters.

The experiments were conducted on a laptop with a *AMD Ryzen 7 5700U* processor, with a clock-speed of 1.80 GHz, and 16 GB RAM. All of the developed heuristics were implemented using Python 3.9.

5.1 Test instances

Let us consider a directed graph $G = (V, A)$ with a distance matrix $D = \{d_{(i,j)} : (i, j) \in A\}$ which contains the distance of every arc in the set A. We will create test instances from D by adopting a similar approach to Ghiani et al. [2006]. Using three parameters denoted by α, β and γ, the number of black vertices as well as the value of Q and L will be determined as follows:

1. If $|V|$ is equal to n, then Q is set to be $\lceil \alpha n \rceil$;

2. Knowing the value of Q, determine b_{min} as the smallest integer satisfying $b_{min} \geq \lceil (n - b_{min})/Q \rceil$. Set the number of black vertices in the instance to be $\lceil \beta b_{min} \rceil$;

3. Denote l_{max} as the length of the longest path segment between two consecutive black vertices when the current instance of the BWTSP is solved heuristically for $L = +\infty$. Set $L = \gamma l_{max}$.

We will consider $\alpha = \{0.2, 0.35, 0.5\}$, $\beta = \{1, 1.33, 1.67\}$ and $\gamma = \{0.95, \infty\}$, which means that 18 different instances can be obtained from the same distance matrix.

As it was mentioned before, a value b_{min} is calculated in order to determine $|B|$. Formally b_{min} can be defined as the minimal number of required black vertices in the graph to assure cardinality feasibility. On Section 3.4.1 we stated that a cardinality feasible solution for a BWTSP instance can be obtained as long as the inequality $Q|B| \geq |W|$ applies. In other words, b_{min} is the smallest value $|B|$ can take to guarantee that the previous inequality is satisfied. Once the number of black vertices in the graph is accordingly determined, it is important to select which nodes are going to be coloured as black. For simplicity's sake, most of the referenced articles select the first $\lceil \beta b_{min} \rceil$ vertices in the graph to be black. However, we have decided to select node 1 as the initial black node and we iteratively select the next black node as the vertex which maximizes the minimal distance to the nodes already coloured as black - this process stops once $\lceil \beta b_{min} \rceil$ black vertices have been selected. As mentioned on the literature review of this dissertations (see Section 1.2), Bourgeois et al. [2003] tested heuristics for the BWTSP on instances with different levels of dispersion of the black vertices and Gouveia et al. [2017] tested exact methods for the BWTSP on randomly generated instances, with different levels of dispersion of the black vertices as well, and both articles emphasize that on instances with tight values of the parameters $|B|$, Q and L, a small dispersion of the black nodes tends to hinder solving it to optimality (in regards to exact methods) or find feasible solutions (in regards to heuristic methods). We refer to these conclusions in order to justify our selection method for the black nodes in a graph.

It is also worth mentioning that we consider the distance between two different nodes $i \in V$ and $j \in V \setminus \{i\}$ as $\max\{d_{(i,j)}, d_{(j,i)}\}$, since the distance matrix can be asymmetric. The following pseudocode contains the algorithm we used to select the black vertices in a graph.

Algorithm 12 Determining set B

Require: A set of vertices V, a distance matrix D and the values of parameters β and b_{min}
1: $B \leftarrow \{1\}$
2: **while** $|B| < \lceil \beta b_{min} \rceil$ **do**
3: $\quad next^b \leftarrow \arg\max_{i \in V \setminus B} \left\{ \min_{b \in B} \left\{ \max\{d_{(i,b)}, d_{(b,i)}\} \right\} \right\}$
4: $\quad B \leftarrow B \cup \{next^b\}$
5: **end while**
6: **return** B

Furthermore, it is worth mentioning that the original article ([Ghiani et al., 2006]) considered l_{max} to be the length of the longest path segment when the BWTSP instance is solved to optimality for $L = +\infty$. Instead, we chose to use a heuristic solution to define l_{max}, similarly to İbrahim Muter [2015], because determining the optimal solution when $L = +\infty$ would demand a lot of computational effort for larger instances. Therefore, the following approach is considered:

1. Knowing the set B and the value of Q, consider $L = +\infty$ and determine an initial feasible solution through the ANN heuristic (Section 3.1) and then apply the Local Search procedure (detailed on Section 4.1). Save the resultant solution as $s_{(ANN+LS)}$.

2. Knowing the set B and the value of Q, consider $L = +\infty$ and determine an initial feasible solution through the AFI heuristic (Section 3.2) and then apply the Local Search procedure (detailed on Section 4.1). Save the resultant solution as $s_{(AFI+LS)}$.

3. If $\mathcal{V}\left(s_{(ANN+LS)}\right) < \mathcal{V}\left(s_{(AFI+LS)}\right)$, then $s^h = s_{(ANN+LS)}$; otherwise, $s^h = s_{(AFI+LS)}$.

4. Determine l_{max} as the length of the longest path segment between two consecutive black vertices in s^h.

Note that we are not using a third heuristic solution, which is the output of the Local Search procedure applied to an initial solution provided by the Adapted Random Insertion (ARI) heuristic (Section 3.3), mainly because this constructive heuristic is not deterministic, unlike the ANN and the AFI heuristics. This means that these two last procedures return the same initial solutions for any BWTSP instance; the same does not apply to the ARI heuristic because it has a random component to it. A similar line of thought justifies why we are using the Local Search procedure as an improvement mechanism at this stage instead of the Iterated Local Search heuristic: the latter incorporates a random perturbation method whereas the former searches for feasible solutions within pre-determined neighborhoods. Therefore, we decided to use two **deterministic** heuristic solutions to guarantee that the same instance of the BWTSP has the same value for l_{max}, for a matter of consistency.

TSPLIB matrices, both symmetric and asymmetric, were used for all the conducted tests with a number of vertices ranging from 50 to 250. The list of considered matrices is as follows:

- **Symmetric instances:** berlin52, pr76, kroA100, pr124, pr152, rat195, pr226;
- **Asymmetric instances:** ft53, ftv64, ft70, kro124p, ftv170.

Since 18 different combinations of parameters are applied on each distance matrix, then 216 instances are considered during the tests, 126 are symmetric and 90 are asymmetric. The set of black nodes B and the values of Q and L associated to each instance can be consulted in Appendix A.

5.1.1 Optimal values

In order to evaluate the performance of the heuristics, it is important to compare all the results obtained heuristically with the corresponding optimal value. In particular, we will use the gap between a heuristic solution, denoted by x^h, and the optimal solution, denoted by x^*, calculated as follows:

$$\text{Gap} = \left(\frac{\mathcal{V}(x^h) - \mathcal{V}(x^*)}{\mathcal{V}(x^*)}\right) \times 100\% \tag{5.1}$$

To obtain the value of the optimal solution for a given instance, we used IBM® ILOG® CPLEX® *Optimization Studio* 20.1.0. It was previously mentioned in this dissertation that the BWTSP is a NP-hard problem in the theory of computational complexity. As the number of arcs in the considered graph increases, it becomes more and more time-consuming to determine the optimal solution of the corresponding instance. Therefore, we have decided to impose a time limit of 5 hours (18000 seconds) to the resolution of every instance in the solver. This limitation did not allow us to determine the optimal value of some instances, so the corresponding lower and upper bounds of the Branch & Cut algorithm embedded on the solver were considered as references for the optimal value of every instance where the time limit was exceeded. Whenever the optimal value was not obtained within 5 hours of resolution, we will use the knowledge of its respective lower bound, let it be denoted by $\underline{\mathcal{V}(x^*)}$, and replace $\mathcal{V}(x^*)$ in the equation (5.1) with $\underline{\mathcal{V}(x^*)}$ to determine an overestimation of the gap between the value of a heuristic solution and the optimal one.

All of the considered TSPLIB matrices resulted in 18 different instances, knowing that each one of these 18 instances is uniquely characterized by a combination of the parameters (α, β, γ). Remember that $\alpha \in \{0.2, 0.35, 0.5\}$, $\beta \in \{1, 1.33, 1.67\}$ and $\gamma \in \{0.95, \infty\}$. The Integer Linear Programming model we presented in Section 2.1 was used to obtain the following results:

Table 5.1: Optimal values of the symmetric instances

Matrix name	α	β	γ	**Optimal value**
berlin52	0.2	1	0.95	$[7555, +\infty[$

Matrix name	α	β	γ	**Optimal value**
berlin52	0.2	1	∞	$[7683, 8742]$
		1.33	0.95	$[7643, 8111]$
			∞	$[7572, 8064]$
		1.67	0.95	7731
			∞	7731
	0.35	1	0.95	7929
			∞	7929
		1.33	0.95	7657
			∞	7657
		1.67	0.95	7657
			∞	7657
	0.5	1	0.95	7779
			∞	7696
		1.33	0.95	7834
			∞	7674
		1.67	0.95	7657
			∞	7657
pr76	0.2	1	0.95	$[106622, +\infty[$
			∞	$[107962, 109454]$
		1.33	0.95	$[107867, 108983]$
			∞	$[107863, 108983]$
		1.67	0.95	$[108440, 108863]$
			∞	$[108505, 108863]$
	0.35	1	0.95	$[107552, 112954]$
			∞	$[107652, 112095]$
		1.33	0.95	109021
			∞	109021
		1.67	0.95	108137
			∞	108137
	0.5	1	0.95	109021
			∞	109021
		1.33	0.95	109021
			∞	109021
		1.67	0.95	108422
			∞	108137
kroA100	0.2	1	0.95	$[20989, +\infty[$
			∞	$[21006, 26766]$
		1.33	0.95	21247
			∞	21247
		1.67	0.95	21247
			∞	21247

Matrix name	α	β	γ	**Optimal value**
kroA100	0.35	1	0.95	$[21118, +\infty[$
			∞	$[20984, 23825]$
		1.33	0.95	$[21009, +\infty[$
			∞	$[21191, 21522]$
		1.67	0.95	21247
			∞	21247
	0.5	1	0.95	$[20876, +\infty[$
			∞	$[21266, 21778]$
		1.33	0.95	$[21232, 21690]$
			∞	$[21228, 21601]$
		1.67	0.95	21247
			∞	21247
pr124	0.2	1	0.95	$[57095, +\infty[$
			∞	$[57513, 60724]$
		1.33	0.95	$[54394, +\infty[$
			∞	$[54463, 65895]$
		1.67	0.95	$[53664, 72768]$
			∞	$[53633, +\infty[$
	0.35	1	0.95	$[55685, +\infty[$
			∞	$[55852, 61931]$
		1.33	0.95	59011
			∞	59011
		1.67	0.95	$[55354, +\infty[$
			∞	$[55713, 60998]$
	0.5	1	0.95	$[56120, +\infty[$
			∞	$[56938, 59397]$
		1.33	0.95	$[57868, 59011]$
			∞	$[56564, 59011]$
		1.67	0.95	59011
			∞	59011
pr152	0.2	1	0.95	$[65806, +\infty[$
			∞	$[65491, +\infty[$
		1.33	0.95	$[65917, +\infty[$
			∞	$[65910, +\infty[$
		1.67	0.95	$[65576, +\infty[$
			∞	$[65647, +\infty[$
	0.35	1	0.95	$[67995, +\infty[$
			∞	$[67856, 109962]$
		1.33	0.95	$[62309, +\infty[$
			∞	$[61545, +\infty[$
		1.67	0.95	$[64715, +\infty[$
			∞	$[64644, 276764]$

Matrix name	α	β	γ	**Optimal value**
pr152	0.5	1	0.95	$[66714, +\infty[$
			∞	$[67888, 83637]$
		1.33	0.95	$[67891, +\infty[$
			∞	$[67898, 79989]$
		1.67	0.95	$[61666, +\infty[$
			∞	$[60341, +\infty[$
rat195	0.2	1	0.95	$[2227, +\infty[$
			∞	$[2230, +\infty[$
		1.33	0.95	$[2231, +\infty[$
			∞	$[2234, +\infty[$
		1.67	0.95	$[2231, +\infty[$
			∞	$[2233, +\infty[$
	0.35	1	0.95	$[2253, 2421]$
			∞	$[2256, 2577]$
		1.33	0.95	$[2247, +\infty[$
			∞	$[2243, 4490]$
		1.67	0.95	$[2226, +\infty[$
			∞	$[2230, +\infty[$
	0.5	1	0.95	2275
			∞	$[2273, 2279]$
		1.33	0.95	$[2270, 2272]$
			∞	$[2253, 2332]$
		1.67	0.95	$[2245, +\infty[$
			∞	2248
pr226	0.2	1	0.95	$[72587, +\infty[$
			∞	$[72534, +\infty[$
		1.33	0.95	$[74300, +\infty[$
			∞	$[74334, +\infty[$
		1.67	0.95	$[75306, +\infty[$
			∞	$[75219, +\infty[$
	0.35	1	0.95	$[72888, +\infty[$
			∞	$[72589, 112721]$
		1.33	0.95	$[72187, +\infty[$
			∞	$[72743, 85851]$
		1.67	0.95	$[73744, +\infty[$
			∞	$[73890, 88008]$
	0.5	1	0.95	$[71692, +\infty[$
			∞	$[71695, 126677]$
		1.33	0.95	$[72753, +\infty[$
			∞	$[72714, 130171]$
		1.67	0.95	$[72419, +\infty[$
			∞	$[72714, +\infty[$

Table 5.2: Optimal values of the asymmetric instances

Matrix name	α	β	γ	Optimal value
ft53	0.2	1	0.95	$[6986, 8181]$
			∞	$[7018, 8292]$
		1.33	0.95	7131
			∞	7131
		1.67	0.95	7023
			∞	7023
	0.35	1	0.95	7086
			∞	7086
		1.33	0.95	7086
			∞	7086
		1.67	0.95	6905
			∞	6905
	0.5	1	0.95	6905
			∞	6905
		1.33	0.95	6905
			∞	6905
		1.67	0.95	6905
			∞	6905
ftv64	0.2	1	0.95	1883
			∞	1883
		1.33	0.95	1855
			∞	1855
		1.67	0.95	1839
			∞	1839
	0.35	1	0.95	1846
			∞	1846
		1.33	0.95	1846
			∞	1846
		1.67	0.95	1839
			∞	1839
	0.5	1	0.95	1850
			∞	1850
		1.33	0.95	1842
			∞	1842
		1.67	0.95	1842
			∞	1842
ft70	0.2	1	0.95	$[38785, +\infty[$
			∞	$[38776, 39896]$
		1.33	0.95	38803
			∞	38803
		1.67	0.95	38803

Matrix name	α	β	γ	**Optimal value**
ft70	0.2	1.67	∞	38803
	0.35	1	0.95	38719
			∞	38719
		1.33	0.95	38712
			∞	38712
		1.67	0.95	38712
			∞	38712
	0.5	1	0.95	38707
			∞	38707
		1.33	0.95	38673
			∞	38673
		1.67	0.95	38673
			∞	38673
kro124p	0.2	1	0.95	$[36396, +\infty[$
			∞	$[36447, 43634]$
		1.33	0.95	36230
			∞	36230
		1.67	0.95	36230
			∞	36230
	0.35	1	0.95	$[36365, +\infty[$
			∞	$[36355, 39751]$
		1.33	0.95	36612
			∞	36612
		1.67	0.95	36230
			∞	36230
	0.5	1	0.95	$[36624, 36873]$
			∞	$[36640, 36873]$
		1.33	0.95	36650
			∞	36650
		1.67	0.95	36230
			∞	36230
ftv170	0.2	1	0.95	$[2717, 3879]$
			∞	$[2742, 4299]$
		1.33	0.95	2758
			∞	2758
		1.67	0.95	2758
			∞	2758
	0.35	1	0.95	$[2742, 2865]$
			∞	$[2754, 2831]$
		1.33	0.95	2755
			∞	2755
		1.67	0.95	2755
			∞	2755

Matrix name	α	β	γ	Optimal value
ftv170	0.5	1	0.95	$[2757, 2789]$
			∞	$[2747, 2946]$
		1.33	0.95	2766
			∞	2766
		1.67	0.95	2755
			∞	2755

None of the considered instances were proven to be infeasible. Whenever the optimal integer solution was not obtained within the established time limit, the solver returns the lower and upper bounds for the global minimum. In particular, the upper bound corresponds to the value in the objective function of the feasible integer solution with the lowest cost when the Branch & Cut algorithm stopped. Thus, it is possible to see for which BWTSP instances no feasible integer solution was found within the established time limit because their upper bounds appear as $+\infty$.

As the number of vertices increases, so does the number of arcs. Therefore, the number of variables and constraints in the corresponding Integer Linear Programming model also increases, making it less likely that the optimal integer solution of the BWTSP instance is determined within the time period of 5 hours, which is clear in Tables 5.1 and 5.2. It is also possible to see for symmetric and asymmetric instances of similar dimensions, more global optimums were obtained for asymmetric cases.

It is interesting to see that, for a given matrix, it becomes harder to determine the global optimum within the time limit when we consider the minimum number of black nodes possible (when $\beta = 1$). It also appears to be more time consuming to determine the optimal integer solution whenever $\alpha = 0.2$. Remember that Q, the superior limit to the number of white nodes in each path segment of a feasible circuit of the BWTSP, is equal to $\lceil \alpha n \rceil$; thus, a reduced value for the parameter α will contribute to a relatively tight Q.

Finally, it is important to note that some of these instances present a large deviation between their upper and lower bounds. In the following sections, whenever the global optimum is unknown, we will compare heuristic results with the lower bounds we obtained, which means that we will calculate overestimations of the respective real gaps in these instances and it is expected that these overestimations are very loose.

5.2 Comparative study of the constructive heuristics

On Chapter 3 of this dissertation, three constructive heuristics were proposed: the Adapted Nearest Neighbor (ANN), the Adapted Farthest Insertion (AFI) and the Adapted Random Insertion (ARI). The first two methods are deterministic and consist of slight adaptations on the selection methods of the Nearest Neighbor and the Farthest Insertion heuristics, respectively, for the classic TSP. The third heuristic, on the other hand, contains a random component, which means that if a given instance is considered and we try multiple times to determine an initial solution for it through the ARI heuristic, we might obtain different solutions each time we run this algorithm. Remember that each of these three heuristics can be divided in two different stages: a construction stage, which is unique for all of them, and a correction stage, which is common for the three. The correction stage aims to correct cardinality and length infeasibility. When we try to establish length feasibility, it is forbidden to return the solution to a cardinality infeasible state, which restricts the diversity of changes we can attempt to perform onto the considered

Hamiltonian circuit. Thus, we risk being unsuccessful on the task of transforming a circuit into a feasible solution for a BWTSP instance.

In order to be able to compare the performance of the three constructive heuristics, we ran 10 times each of the 216 considered instances to obtain three key informations:

- Regarding the ARI heuristic, out of the 10 runs for each instance, how many of them were successful?
- How far from the global minimum of each instance (or its lower bound) are the values of the initial feasible solutions provided by each of the three constructive heuristics?
- How much CPU time is required to obtain a feasible solution through each of the three constructive algorithms?

The following table contains the gap (or an overestimation of it, if it is followed by the symbol *) between the corresponding heuristic solution and its global optimum for both the ANN and the AFI heuristics. For the ARI heuristic, it contains statistics concerning the number of successful runs (out of 10) for each instance. By "successful" we mean a run where a feasible solution was obtained for the given instance. Furthermore, the minimum, the average and the maximum gaps of the successful runs of the ARI heuristic are also displayed.

Table 5.3: Statistics for the comparative study of the ANN, AFI and ARI heuristics

Symmetric instances									
Instance				**ANN**	**AFI**	**ARI**			
Matrix	α	β	γ	**Gap**	**Gap**	**Feasible solutions**	**Gap Min**	**Gap Average**	**Gap Max**
berlin52	0.2	1	0.95	———	———	0/10	———	———	———
			∞	43.41% *	23.43% *	10/10	17.4% *	28.94% *	44.4% *
		1.33	0.95	83.68% *	20.02% *	10/10	13.23% *	21.08% *	43.05% *
			∞	30.72% *	21.14% *	10/10	19.16% *	22.82% *	28.04% *
		1.67	0.95	———	10.93%	7/10	6.88%	23.2%	63.38%
			∞	20.39%	10.93%	10/10	5.98%	14.54%	24.42%
	0.35	1	0.95	30.68%	27.3%	8/10	15.53%	27.22%	47.27%
			∞	43.09%	31.68%	10/10	22.08%	31.52%	48.28%
		1.33	0.95	18.9%	38.91%	5/10	17.16%	26.87%	34.16%
			∞	23.81%	21.61%	10/10	5.97%	14.17%	22.2%
		1.67	0.95	28.68%	8.29%	10/10	5.86%	11.98%	21.0%
			∞	28.68%	8.29%	10/10	7.76%	10.44%	15.82%
	0.5	1	0.95	———	———	3/10	6.52%	10.79%	15.17%
			∞	44.76%	20.17%	10/10	1.83%	18.05%	28.78%
		1.33	0.95	———	18.05%	7/10	11.78%	18.74%	25.99%
			∞	29.24%	20.51%	10/10	4.74%	12.46%	20.95%
		1.67	0.95	32.38%	33.03%	10/10	7.65%	14.61%	50.01%
			∞	34.3%	8.29%	10/10	3.8%	9.54%	16.42%

Instance				ANN	AFI	ARI			
Matrix	α	β	γ	Gap	Gap	Feasible solutions	Gap		
							Min	Average	Max
pr76	0.2	1	0.95	———	———	0/10	———	———	———
			∞	36.3% *	46.11% *	10/10	6.93% *	14.28% *	24.92% *
		1.33	0.95	52.3% *	29.61% *	5/10	8.46% *	33.69% *	55.84% *
			∞	48.16% *	16.93% *	10/10	6.4% *	13.6% *	22.38% *
		1.67	0.95	54.57% *	6.67% *	10/10	2.92% *	7.09% *	10.19% *
			∞	42.83% *	6.6% *	10/10	2.66% *	6.79% *	13.79% *
	0.35	1	0.95	———	———	1/10	36.53% *	36.53% *	36.53% *
			∞	39.68% *	48.29% *	10/10	17.27% *	31.41% *	51.79% *
		1.33	0.95	42.49%	24.45%	10/10	7.54%	20.47%	57.16%
			∞	43.97%	16.97%	10/10	9.56%	14.56%	24.89%
		1.67	0.95	72.49%	10.67%	10/10	2.7%	11.44%	40.33%
			∞	34.46%	10.67%	10/10	4.36%	9.6%	12.24%
	0.5	1	0.95	———	———	3/10	4.09%	11.85%	16.64%
			∞	45.57%	23.81%	10/10	5.4%	19.44%	25.67%
		1.33	0.95	50.66%	23.81%	10/10	21.89%	32.24%	64.54%
			∞	45.57%	23.81%	10/10	5.65%	19.89%	32.85%
		1.67	0.95	41.12%	———	6/10	2.15%	13.9%	23.47%
			∞	36.89%	10.67%	10/10	4.9%	11.33%	20.74%
kroA100	0.2	1	0.95	———	———	1/10	23.4% *	23.4% *	23.4% *
			∞	29.91% *	30.46% *	10/10	28.95% *	43.21% *	59.3% *
		1.33	0.95	———	5.01%	6/10	2.52%	5.58%	10.15%
			∞	38.03%	5.01%	10/10	3.17%	6.11%	11.87%
		1.67	0.95	———	5.01%	10/10	4.17%	16.69%	46.62%
			∞	30.71%	5.01%	10/10	1.16%	7.97%	16.0%
	0.35	1	0.95	———	———	0/10	———	———	———
			∞	49.91% *	55.82% *	10/10	33.81% *	55.07% *	69.09% *
		1.33	0.95	———	10.97% *	10/10	5.65% *	23.73% *	48.88% *
			∞	35.42% *	10.02% *	10/10	4.77% *	14.43% *	34.44% *
		1.67	0.95	54.52%	63.4%	10/10	4.41%	25.39%	74.42%
			∞	27.27%	9.73%	10/10	4.31%	7.32%	11.6%
	0.5	1	0.95	———	———	0/10	———	———	———
			∞	30.54% *	31.4% *	10/10	21.62% *	28.47% *	37.27% *
		1.33	0.95	28.13% *	60.53% *	7/10	12.85% *	47.45% *	56.44% *
			∞	30.83% *	29.42% *	10/10	7.84% *	22.93% *	36.68% *
		1.67	0.95	47.77%	9.73%	10/10	6.28%	13.85%	57.08%
			∞	30.71%	9.73%	10/10	5.48%	9.78%	18.13%
pr124	0.2	1	0.95	———	19.92% *	0/10	———	———	———
			∞	55.05% *	19.05% *	10/10	14.84% *	25.08% *	41.49% *
		1.33	0.95	53.58% *	26.4% *	9/10	12.73% *	15.83% *	21.01% *
			∞	35.65% *	20.65% *	10/10	14.1% *	21.5% *	26.49% *
		1.67	0.95	38.88% *	74.27% *	8/10	15.53% *	18.71% *	23.42% *

Instance				ANN	AFI	ARI			
Matrix	α	β	γ	Gap	Gap	Feasible solutions	Gap Min	Gap Average	Gap Max
pr124	0.2	1.67	∞	29.18% *	19.91% *	10/10	13.42% *	17.88% *	29.67% *
	0.35	1	0.95	———	———	2/10	28.75% *	30.13% *	31.51% *
			∞	40.81% *	31.54% *	10/10	27.13% *	37.31% *	48.68% *
		1.33	0.95	———	———	1/10	12.15%	12.15%	12.15%
			∞	21.94%	17.58%	10/10	7.26%	13.95%	18.32%
		1.67	0.95	———	18.71% *	10/10	10.96% *	17.56% *	44.75% *
			∞	29.15% *	17.94% *	10/10	9.77% *	15.69% *	19.44% *
	0.5	1	0.95	———	———	0/10	———	———	———
			∞	31.11% *	27.57% *	10/10	23.27% *	29.07% *	35.7% *
		1.33	0.95	———	13.55% *	8/10	6.59% *	10.49% *	20.2% *
			∞	23.4% *	16.17% *	10/10	6.13% *	11.59% *	17.74% *
		1.67	0.95	———	24.28%	5/10	5.3%	9.15%	13.67%
			∞	17.41%	11.35%	10/10	3.45%	7.1%	11.73%
pr152	0.2	1	0.95	33.01% *	———	1/10	28.46% *	28.46% *	28.46% *
			∞	33.65% *	53.4% *	10/10	33.11% *	45.28% *	59.65% *
		1.33	0.95	35.68% *	36.31% *	1/10	35.97% *	35.97% *	35.97% *
			∞	35.69% *	36.32% *	10/10	23.73% *	35.96% *	49.39% *
		1.67	0.95	32.95% *	20.92% *	9/10	17.11% *	21.59% *	28.51% *
			∞	32.81% *	20.79% *	10/10	17.84% *	23.74% *	34.83% *
	0.35	1	0.95	51.61% *	———	2/10	41.19% *	43.64% *	46.09% *
			∞	66.93% *	65.06% *	10/10	45.6% *	71.94% *	97.2% *
		1.33	0.95	36.7% *	22.7% *	9/10	20.85% *	23.43% *	29.86% *
			∞	38.84% *	24.23% *	10/10	23.3% *	29.94% *	41.75% *
		1.67	0.95	32.33% *	———	2/10	16.74% *	28.29% *	39.84% *
			∞	32.48% *	18.27% *	10/10	17.22% *	23.86% *	35.63% *
	0.5	1	0.95	———	———	0/10	———	———	———
			∞	27.57% *	24.61% *	10/10	11.09% *	26.3% *	44.85% *
		1.33	0.95	39.51% *	12.61% *	8/10	12.26% *	19.08% *	29.6% *
			∞	28.18% *	12.6% *	10/10	12.97% *	25.35% *	44.63% *
		1.67	0.95	———	———	0/10	———	———	———
			∞	41.44% *	26.7% *	10/10	25.36% *	27.12% *	30.0% *
rat195	0.2	1	0.95	———	———	0/10	———	———	———
			∞	27.26% *	35.43% *	10/10	32.96% *	44.86% *	55.74% *
		1.33	0.95	———	———	1/10	13.18% *	13.18% *	13.18% *
			∞	18.04% *	12.09% *	10/10	13.61% *	20.17% *	27.04% *
		1.67	0.95	20.22% *	12.24% *	10/10	15.15% *	16.76% *	19.72% *
			∞	26.47% *	12.14% *	10/10	13.3% *	17.49% *	21.32% *
	0.35	1	0.95	———	———	1/10	19.17% *	19.17% *	19.17% *
			∞	30.27% *	20.97% *	10/10	21.45% *	32.27% *	43.44% *
		1.33	0.95	18.42% *	11.44% *	8/10	13.13% *	16.04% *	18.51% *
			∞	18.64% *	11.64% *	10/10	12.8% *	19.11% *	35.18% *

Instance				ANN	AFI	ARI			
Matrix	α	β	γ	**Gap**	**Gap**	**Feasible solutions**	**Gap Min**	**Gap Average**	**Gap Max**
rat195	0.35	1.67	0.95	—	—	1/10	29.38% *	29.38% *	29.38% *
			∞	19.33% *	12.29% *	10/10	12.24% *	15.78% *	17.85% *
	0.5	1	0.95	—	—	0/10	—	—	—
			∞	27.76% *	9.11% *	10/10	11.39% *	21.44% *	28.02% *
		1.33	0.95	19.16% *	32.51% *	1/10	11.41% *	11.41% *	11.41% *
			∞	17.49% *	11.14% *	10/10	14.87% *	20.37% *	26.94% *
		1.67	0.95	18.53% *	11.54% *	10/10	12.69% *	15.88% *	20.94% *
			∞	18.37%	11.39%	10/10	13.48%	15.22%	18.77%
pr226	0.2	1	0.95	55.83% *	—	7/10	63.76% *	87.39% *	96.82% *
			∞	55.94% *	55.21% *	10/10	51.76% *	76.35% *	99.06% *
		1.33	0.95	—	24.85% *	8/10	13.44% *	20.13% *	24.35% *
			∞	31.07% *	24.79% *	10/10	14.72% *	27.19% *	46.91% *
		1.67	0.95	57.86% *	11.76% *	10/10	10.36% *	23.0% *	91.32% *
			∞	25.83% *	11.89% *	10/10	9.12% *	10.47% *	12.73% *
	0.35	1	0.95	51.18% *	—	3/10	36.44% *	46.19% *	56.65% *
			∞	52.89% *	33.97% *	10/10	35.24% *	54.25% *	69.84% *
		1.33	0.95	—	16.59% *	10/10	13.24% *	14.97% *	17.67% *
			∞	30.15% *	15.7% *	10/10	13.35% *	15.0% *	17.95% *
		1.67	0.95	—	14.13% *	6/10	11.97% *	13.13% *	14.88% *
			∞	28.13% *	13.9% *	10/10	11.37% *	13.67% *	16.27% *
	0.5	1	0.95	50.04% *	—	0/10	—	—	—
			∞	48.55% *	41.28% *	10/10	24.92% *	38.49% *	60.27% *
		1.33	0.95	39.57% *	59.78% *	10/10	18.07% *	69.8% *	98.71% *
			∞	46.46% *	15.75% *	10/10	12.77% *	14.08% *	15.24% *
		1.67	0.95	—	16.22% *	10/10	13.12% *	14.86% *	16.81% *
			∞	30.2% *	15.75% *	10/10	12.29% *	14.76% *	17.38% *

Asymmetric instances

Instance				ANN	AFI	ARI			
Matrix	α	β	γ	**Gap**	**Gap**	**Feasible solutions**	**Gap Min**	**Gap Average**	**Gap Max**
ft53	0.2	1	0.95	40.9% *	—	10/10	29.73% *	51.98% *	74.59% *
			∞	40.25% *	54.49% *	10/10	31.73% *	50.77% *	74.41% *
		1.33	0.95	40.15%	57.96%	10/10	19.17%	30.46%	45.1%
			∞	40.15%	26.36%	10/10	15.13%	26.6%	39.45%
		1.67	0.95	44.55%	24.26%	10/10	14.04%	19.56%	25.72%
			∞	33.28%	24.26%	10/10	10.39%	18.0%	32.44%
	0.35	1	0.95	51.02%	—	8/10	12.42%	24.34%	32.44%
			∞	41.8%	39.18%	10/10	20.89%	32.51%	47.05%
		1.33	0.95	71.92%	32.49%	10/10	6.66%	32.43%	87.58%
			∞	39.88%	32.49%	10/10	12.15%	23.62%	39.27%

Instance				ANN	AFI	ARI			
Matrix	α	β	γ	Gap	Gap	Feasible solutions	Gap Min	Gap Average	Gap Max
ft53	0.35	1.67	0.95	61.13%	21.13%	10/10	14.95%	24.88%	80.48%
			∞	42.65%	21.13%	10/10	12.12%	21.46%	28.81%
	0.5	1	0.95	49.47%	18.57%	6/10	22.42%	29.8%	42.26%
			∞	49.47%	18.57%	10/10	14.76%	24.24%	33.83%
		1.33	0.95	67.69%	18.57%	10/10	9.41%	21.54%	31.99%
			∞	43.33%	18.57%	10/10	11.53%	22.16%	27.6%
		1.67	0.95	67.69%	7.63%	10/10	14.09%	19.26%	26.75%
			∞	43.33%	7.63%	10/10	15.9%	21.07%	34.77%
ftv64	0.2	1	0.95	———	37.92%	9/10	19.92%	39.45%	69.52%
			∞	56.88%	37.92%	10/10	20.29%	38.8%	66.44%
		1.33	0.95	———	18.27%	10/10	10.73%	18.09%	24.26%
			∞	48.03%	18.27%	10/10	10.51%	20.08%	26.42%
		1.67	0.95	60.47%	———	7/10	13.27%	41.41%	74.5%
			∞	45.08%	15.39%	10/10	12.29%	19.05%	31.32%
	0.35	1	0.95	53.41%	45.77%	4/10	31.42%	44.35%	58.94%
			∞	44.2%	45.77%	10/10	20.96%	38.65%	50.81%
		1.33	0.95	33.21%	30.5%	10/10	10.83%	28.81%	48.37%
			∞	47.83%	30.5%	10/10	15.44%	22.95%	32.45%
		1.67	0.95	69.44%	27.51%	10/10	8.43%	17.43%	27.24%
			∞	43.5%	27.51%	10/10	9.73%	16.65%	27.46%
	0.5	1	0.95	44.7%	———	6/10	11.57%	18.26%	22.49%
			∞	48.38%	43.08%	10/10	22.16%	32.15%	38.65%
		1.33	0.95	71.88%	43.7%	10/10	15.15%	21.79%	29.15%
			∞	45.49%	43.7%	10/10	15.04%	23.94%	40.88%
		1.67	0.95	65.31%	———	10/10	13.03%	25.59%	54.23%
			∞	43.27%	28.66%	10/10	9.39%	16.78%	24.92%
ft70	0.2	1	0.95	21.28% *	17.87% *	2/10	17.29% *	18.35% *	19.41% *
			∞	16.83% *	14.56% *	10/10	12.54% *	17.34% *	20.76% *
		1.33	0.95	20.71%	16.36%	10/10	14.83%	20.27%	29.08%
			∞	10.38%	10.76%	10/10	8.75%	11.68%	13.87%
		1.67	0.95	18.99%	23.4%	10/10	13.37%	20.68%	25.73%
			∞	10.38%	10.86%	10/10	8.38%	10.95%	13.68%
	0.35	1	0.95	17.81%	12.24%	10/10	8.34%	17.04%	28.58%
			∞	13.2%	12.24%	10/10	8.41%	11.65%	14.94%
		1.33	0.95	11.97%	18.75%	10/10	7.21%	12.79%	21.79%
			∞	11.97%	11.83%	10/10	6.49%	9.63%	12.12%
		1.67	0.95	———	17.66%	7/10	9.34%	12.9%	17.18%
			∞	11.97%	8.79%	10/10	7.52%	9.54%	11.46%
	0.5	1	0.95	———	9.32%	6/10	6.94%	8.66%	10.46%
			∞	14.95%	9.32%	10/10	8.51%	10.41%	12.51%

Instance				ANN	AFI	ARI			
Matrix	α	β	γ	Gap	Gap	Feasible solutions	Gap Min	Gap Average	Gap Max
ft70	0.5	1.33	0.95	———	7.44%	4/10	8.02%	13.12%	25.26%
			∞	11.47%	7.44%	10/10	7.55%	9.49%	12.37%
		1.67	0.95	———	8.9%	6/10	6.78%	10.96%	14.9%
			∞	11.47%	8.9%	10/10	7.46%	9.49%	12.2%
kro124p	0.2	1	0.95	———	19.08% *	8/10	18.85% *	29.76% *	37.72% *
			∞	42.19% *	18.92% *	10/10	18.7% *	33.83% *	47.13% *
		1.33	0.95	37.34%	13.15%	7/10	9.53%	18.67%	42.61%
			∞	32.19%	13.15%	10/10	10.04%	14.18%	19.68%
		1.67	0.95	31.36%	14.15%	10/10	11.63%	15.21%	19.48%
			∞	31.2%	14.15%	10/10	10.98%	13.63%	18.03%
	0.35	1	0.95	———	———	0/10	———	———	———
			∞	28.93% *	37.71% *	10/10	26.93% *	35.32% *	44.22% *
		1.33	0.95	28.02%	15.22%	9/10	8.72%	13.3%	19.79%
			∞	28.02%	15.22%	10/10	10.23%	16.13%	31.44%
		1.67	0.95	60.08%	10.29%	10/10	7.57%	11.73%	16.19%
			∞	32.07%	10.29%	10/10	8.77%	13.67%	19.33%
	0.5	1	0.95	———	23.36% *	8/10	20.26% *	23.49% *	27.7% *
			∞	29.66% *	23.31% *	10/10	20.07% *	25.89% *	32.14% *
		1.33	0.95	43.07%	21.31%	10/10	9.72%	20.83%	30.54%
			∞	29.62%	21.31%	10/10	9.13%	19.43%	29.9%
		1.67	0.95	71.88%	10.29%	10/10	9.62%	20.62%	51.73%
			∞	31.12%	10.29%	10/10	9.96%	13.49%	18.85%
ftv170	0.2	1	0.95	———	———	4/10	40.93% *	46.12% *	50.24% *
			∞	45.48% *	40.41% *	10/10	31.4% *	46.61% *	59.88% *
		1.33	0.95	50.4%	34.74%	10/10	11.64%	35.84%	77.92%
			∞	49.17%	34.74%	10/10	25.09%	32.44%	44.67%
		1.67	0.95	64.72%	34.74%	9/10	19.07%	27.72%	38.18%
			∞	43.47%	34.74%	10/10	15.81%	21.92%	31.62%
	0.35	1	0.95	———	34.87% *	6/10	33.84% *	40.98% *	48.83% *
			∞	46.95% *	34.28% *	10/10	32.64% *	38.4% *	46.91% *
		1.33	0.95	———	28.28%	10/10	13.83%	31.47%	41.85%
			∞	45.15%	28.28%	10/10	19.24%	28.87%	36.99%
		1.67	0.95	64.07%	18.91%	10/10	15.28%	22.68%	33.18%
			∞	43.59%	18.91%	10/10	9.98%	24.18%	37.64%
	0.5	1	0.95	———	32.64% *	3/10	20.53% *	25.39% *	29.38% *
			∞	36.55% *	33.13% *	10/10	21.15% *	39.03% *	51.8% *
		1.33	0.95	———	18.33%	10/10	18.08%	29.52%	41.61%
			∞	46.49%	18.33%	10/10	17.97%	28.05%	39.52%
		1.67	0.95	52.63%	18.8%	10/10	13.58%	24.88%	31.22%
			∞	42.4%	18.8%	10/10	19.85%	24.71%	29.33%

Table 5.4: Average CPU time, in seconds, to obtain feasible solutions through the ANN, AFI and ARI heuristics.

Type of instance	**Constructive heuristic**		
	ANN	AFI	ARI
Symmetric	0,0239	0,6184	0,0343
Asymmetric	0,0094	0,2426	0,0173

In average, all of the three methods require less than 1 second to determine their respective feasible solutions. However, it is possible to conclude from Table 5.4 that the AFI heuristic consumes, in average, 25 and 18 times more CPU time to obtain a feasible solution when compared to the ANN and ARI heuristics, relatively. This significant difference can be explained by the implementation of the construction stage of the AFI heuristic, since we determine in every iteration of the corresponding algorithm the minimal distance of every non-inserted node to the determined circuit in order to select the vertex with maximal distance as the next node to be inserted, which demands additional CPU time. If we prioritize reduced computational time over solution quality and consistency, the ANN and ARI heuristics are preferable. When we compare the required computational time to determine feasible solutions between symmetric and asymmetric instances, we might feel tempted to conclude that the constructive algorithms are three times faster in asymmetric cases. However, it is important to bear in mind that the symmetric instances we used for this analysis tend to have a larger size relatively to the asymmetric instances, which might influence some of the disparity of average CPU time.

If we are particularly interested in comparing the performance of both deterministic methods (ANN and AFI), our sample of instances suggests that the AFI heuristic has a higher success rate for determining feasible solutions: regarding symmetric instances, it determined feasible solutions for 80% of the cases, whereas the ANN heuristic obtained feasible solutions only for 75% of them; this is still valid for asymmetric instances because these two algorithms determined feasible solutions for 92% and 84% of the cases, respectively. Furthermore, for the vast majority of instances where both methods found feasible solutions, the solution provided by the AFI has a significantly lower cost. It is true that the ANN heuristic determines feasible solutions quicker, but the trade-off of this is that the AFI method ensures solutions with a value in the objective function closer to the global minimum.

Since the ARI method is not deterministic, it obviously does not provide us the same solution in all of the 10 runs we performed for every instance. On Table 5.3 it is possible to see for the instance which is defined on the asymmetric matrix “ft53”, for $\alpha = 0.35$, $\beta = 1.33$ and $\gamma = 0.95$, that out of the 10 successful runs of the ARI heuristic, the obtained solution with a lower cost has a gap of 6.66% from the optimal value, while the solution with the highest cost has a gap of 87.58% from the same value. This difference of over 80% between the maximum and the minimum gap, out of 10 runs, for the same instance is a great example of how much the quality of the provided solutions might vary. In spite of this deviation, it seems to be common, based on our sample, that after 10 runs of the ARI heuristic for a feasible BWTSP instance at least one of them returns a feasible solution. For symmetric instances, the ARI heuristic determined at least one feasible solution, out of the 10 runs, for 91% of the instances. Regarding asymmetric instances, this rate increases to 98%. Out of the two deterministic methods we proposed in this dissertation, we previously concluded that the AFI heuristic has a higher success rate at obtaining feasible solutions for the considered instances. On one hand, the AFI algorithm has the advantage of being more consistent in regards to solution quality relatively to the ARI, which is not a surprise since it is a deterministic method. On the other hand, it is not negligible that the AFI heuristic could not determine feasible solutions for 14% of our entire sample of instances, whereas the ARI heuristic returned 0 feasible solutions, out of the 10 runs, only for 5.5% of cases. This seems to suggest that if we prioritize obtaining an initial feasible solution for a BWTSP instance, and we find negligible its value in the objective function, then the ARI heuristic is an appropriate constructive method. In fact, one can argue that due to the disparity in CPU time of both algorithms it is possible to replace an unsuccessful run of the AFI heuristic with 18 runs of the ARI heuristics with the premise that, if the overall BWTSP instance is feasible, there is a high chance that at least one of those 18 runs will result in a feasible solution.

At last, the results in Table 5.3 show that whenever we eliminate the length constraint of the BWTSP, which happens everytime we set L to infinity, our constructive heuristics are always able to determine feasible solutions for the BWTSP instance we are considering. This confirmations that the limitations of the methods we developed to determine feasible solutions are based on the procedure we used to correct length infeasibility, just like we stated in Section 3.5.

5.3 Testing parameters for the ILS algorithm

An Iterated Local Search (ILS) algorithm was proposed on Chapter 4. This algorithm is an improvement heuristic, which means that a feasible solution for the BWTSP is required as a starting point for the heuristic. For this purpose we have decided to use the solutions provided by the three constructive heuristics we developed in this dissertation. Since each of these constructive algorithms return solutions with different values in the objective function of the problem, it is fair to analyse separately the gaps we obtained through the three different combinations of heuristics, ANN+ILS, AFI+ILS and ARI+ILS. Furthermore, this study could be an additional contribution for the comparative analysis of the constructive heuristics ANN, AFI and ARI, because it will allow us to take conclusions regarding the impact their initial solutions can have on the performance of the ILS algorithm.

Our goal in the current section is to test two parameters for the ILS heuristic: the strength of the perturbation, denoted by ω, and the maximum number of iterations of the algorithm, $MaxIt$. On one hand, the parameter ω controls the proportion of nodes to randomly remove from a given Hamiltonian circuit s, in order to reinsert them in the position which leads to the minimal increase of cost on the circuit s (detailed in Section 4.2). On the other hand, the parameter $MaxIt$ represents the maximum number of iterations of the ILS heuristic. It is important to remember that our heuristic stops prematurely if after $0.1 \times MaxIt$ iterations the incumbent solution has remained unaltered, which means that $MaxIt$ also regulates when to prematurely terminate the algorithm. Six different combinations of the parameters $(\omega, MaxIt)$ were considered for the tests:

Table 5.5: Different combinations of parameters $(\omega, MaxIt)$ to test for the ILS heuristic.

Parameters	**Combinations**					
ω	1/3	1/3	1/3	1/2	1/2	1/2
$MaxIt$	1000	2500	5000	1000	2500	5000

These tests were conducted on the 216 instances we previously defined, considering the different solutions the three constructive heuristics determined for each one of these instances. For each combination of parameters $(\omega, MaxIt)$, for every instance and for each constructive heuristic, 5 different runs were performed. The results we obtained are very extensive and they can be consulted in Appendix B.

We aim to analyse through these results not only the computational times of the different combinations of parameters, but also if there are any alterations in the dispersion of the gaps for each of the 6 combinations we tested. It is possible to see on Table 5.6 the average computational time required, in seconds, to run the ILS algorithm for the different values of ω and $MaxIt$, considering the three constructive heuristics. We will display the boxplots of the resultant gaps for each combination of parameters, for symmetric and asymmetric instances, respectively. On Figure 5.1 we have all the boxplots associated to the conducted tests on symmetric instances and on Figure 5.2 we present the boxplots associated to the asymmetric cases.

Table 5.6: Average CPU time, in seconds, to run the ILS heuristic for different combinations of parameters.

Parameters		**Symmetric instances**			**Asymmetric instances**		
ω	$MaxIt$	ANN+ILS	AFI+ILS	ARI+ILS	ANN+ILS	AFI+ILS	ARI+ILS
$1/3$	1000	22,09	23,86	42,28	9,97	11,63	20,87
	2500	48,53	53,63	88,43	19,53	24,25	38,60
	5000	88,38	92,61	150,79	35,66	44,73	71,48
$1/2$	1000	25,70	26,58	57,63	9,99	11,81	44,07
	2500	56,22	56,55	112,98	22,55	28,55	50,19
	5000	110,72	112,59	225,57	39,38	46,67	94,57

For symmetric instances, most of the gaps which were greater than 5% were in reality overestimations of the real corresponding gaps, because they are comparisons between an heuristic solution and the lower bound of the global optimum, which was not determined within the time limit we imposed (see Section 5.1.1). The same applies for most of the gaps which were greater than 10% in the asymmetric case. Due to these limitations, the purpose of the boxplots we previously displayed is not to individually analyse them because they are not accurate visual representations of the real quality of the methods we tested. Instead, our goal is to compare the dispersion of the resultant gaps we obtained through the ILS algorithm as a consequence of the factors we altered in every test; those factors can either be the constructive heuristic we used before the ILS, the strength of the perturbation or the maximum number of iterations. In order to do a comparative analysis of the dispersion of the resultant gaps we will study the differences regarding three different statistics for each combination of parameters for the ILS: the first quartile of the gaps, the median values and the third quartile.

It becomes clear from Table 5.6 that when we apply the ILS algorithm on initial solutions provided by the ARI heuristic it consumes, at least, twice the computational time of the combinations ANN+ILS and AFI+ILS. In fact, this conclusion is valid for the six different combinations of parameters $(\omega, MaxIt)$ and for both symmetric and asymmetric instances. From Figures 5.1 and 5.2 it is possible to see that, for each type of instance symmetry and for the same combination of parameters on the ILS heuristic, the additional CPU time for the ARI+ILS combination did not result in a significantly improved solution quality. In fact, Lourenço et al. [2003] mentioned that one key advantage of using greedy constructive heuristics before the ILS algorithm is that it requires less improvement stages during the procedure, in comparison to initial solutions obtained randomly, which helps us explain the disparity in CPU time between starting the ILS from a solution we determined through the ARI and the other two constructive algorithms.

For each combination of parameters $(\omega, MaxIt)$ of the ILS heuristic, it is interesting to see that when we considered initial solutions which had been determined by the AFI algorithm, the interquartile

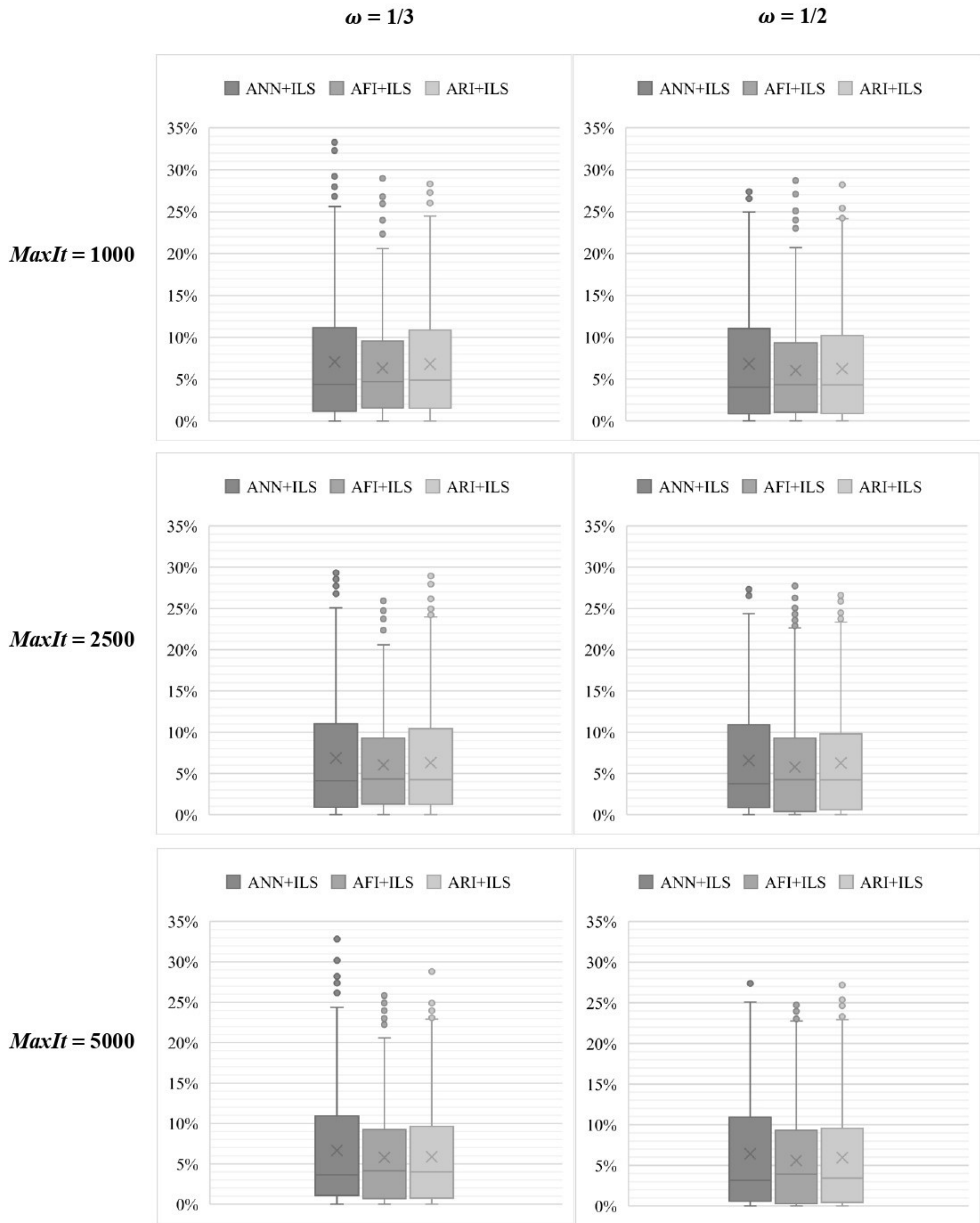

Figure 5.1: Boxplots of the obtained gaps for symmetric instances.

range (which means, the difference between the third and the first quartile) of the resultant gaps is less than the range we observed for gaps which were obtained from the combinations ANN+ILS or ARI+ILS. Such difference is even more prominent in asymmetric instances.

We can also see from Figures 5.1 and 5.2 that increasing the value of both parameters ω and $MaxIt$ enhances our chances of determining new local minimums for the BWTSP instances we considered. This is generalized for symmetric and asymmetric instances because the first quartile, the median value and

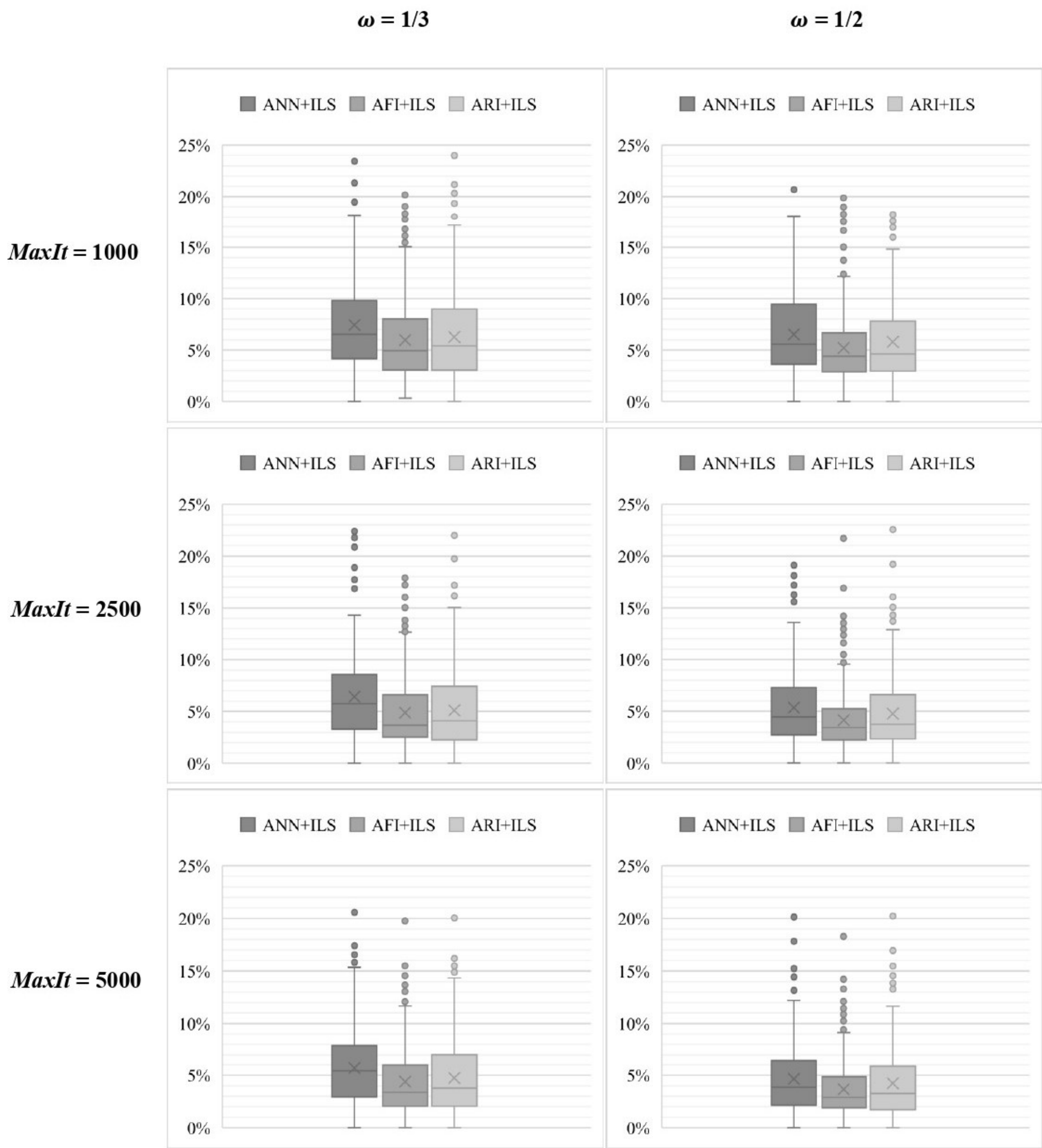

Figure 5.2: Boxplots of the obtained gaps for asymmetric instances.

the third quartile of the gaps we obtained tend to decrease when $\omega = 1/2$, as well as when the maximum number of iterations increases. It is also interesting to see that this decrease is more accentuated in the asymmetric case. For example, between the combination of parameters $(\omega, MaxIt) = (1/3, 1000)$ and $(\omega, MaxIt) = (1/2, 5000)$, the difference between the two sets of boxplots is very residual for symmetric instances: the median, the first and third quartiles decreased, at most, 1% in these cases, regardless of the constructive heuristic we used for the ILS. In contrast, when we compare the two sets of boxplots associated to the previous combinations of parameters $(\omega, MaxIt)$ on asymmetric instances, the differences between the values of the median, the first and third quartiles are much more abrupt. This seems to suggest that when we apply the ILS heuristic on asymmetric instances of the BWTSP, they are more vulnerable not only to the strength of the perturbation method we consider but also to the maximum number of iterations we allow. Note that when we allow more iterations of the ILS heuristic, we are applying more perturbations during the process which might lead us to new local minimums. This constitutes a confirmation that our Local Search procedure is more impactful on symmetric instances, because we are not too dependent on the power of our perturbations in these cases to determine feasible solutions with a good value in the objective function of the BWTSP throughout the ILS algorithm. For context, it is important to remember that we search an additional neighborhood, $\mathcal{N}_{2-exchange}$, during the Local Search procedure we incorporated in the ILS whenever we consider a symmetric BWTSP instance, which can explain the difference in regards to the performance of the Local Search procedure between symmetric and asymmetric instances.

Due to the fact that the asymmetric instances of the BWTSP appear to be relatively sensitive to the quality and quantity of the perturbations we apply during the ILS procedure, it is utterly important to pay attention to the differences regarding the dispersion of the gaps we obtained for the six combinations of parameters $(\omega, MaxIt)$ in the asymmetric case (see Figure 5.2) in order to, finally, select the most appropriate combination of parameters. We can start by eliminating the possibility of considering $MaxIt$ as 1000. The conducted tests, regardless of the values of ω, prove that it consumes an impressively low average of CPU time to run the ILS algorithm when we set the maximum number of iterations to 1000 and we consequently decide to stop the procedure after 100 iterations without improvement. However, the values of the three main statistics we considered (first and third quartiles, and the median) for the resultant gaps were the highest, specially on the asymmetric case.

We have four different combinations of parameters of the ILS algorithm left to compare. If we focus on Figure 5.1, we see that the first quartiles of the gaps we obtained for $\omega = 1/2$, considering $MaxIt$ as 2500 or 5000, are closer to zero relatively to the first quartiles of the gaps we obtained for $\omega = 1/3$ on symmetric instances. Additional differences can be seen for the asymmetric instances we considered, where the medians of the gaps we obtained for $\omega = 1/2$, for $MaxIt$ set to 2500 or to 5000, are lower relatively to the respective medians we obtained when $\omega = 1/3$.

It seems that we only have two interesting combinations of parameters for the ILS heuristic at this point, both of them consider ω as $1/2$ but one sets $MaxIt$ as 2500 while the other sets this value to 5000. For both symmetric and asymmetric instances (Figures 5.1 and 5.2, relatively), we can see that there is not a significant difference amongst the three boxplots we displayed when $MaxIt = 2500$ or when $MaxIt = 5000$. However, when we consider a maximum of 5000 iterations, the average CPU time doubles relatively to the scenario where we used 2500 as the maximum number of iterations. Therefore, we can conclude that, for $\omega = 1/2$, increasing $MaxIt$ from 2500 to 5000 lead to a very modest improvement of the general gaps we obtained while the required computational time, in comparison, increased massively. To sum up, we have decided to define $\omega = 1/2$ and $MaxIt = 2500$ on our ILS algorithm because it accomplishes our purpose of devoloping an improvement heuristic that determines solutions with a good value in the objective function within a reasonable amount of computational time, in comparison to the other five combinations of parameters we analysed.

5.4 Performance of the ILS heuristic

In this section we will summarize the results we obtained for the ILS heuristic we developed. As we explained in the previous section, we ran each of the 216 instances, for each constructive method, 5 times. We display in Table 5.7 the average gap (or an overestimation of it, if it is followed by the symbol *) and the average CPU time, in seconds, we obtained for every instance and for every constructive heuristic we considered. Whenever the corresponding constructive method did not determine an initial feasible solution for a BWTSP instance, we symbolized it with "—-". In order to see the minimum and the maximum gap we obtained for each instance, for the different constructive heuristics we used, see Appendix B.2.2.

Table 5.7: Computational results of the ILS algorithm, considering the determined solutions by the three different constructive heuristics.

Symmetric instances									
Instance				**ANN+ILS**		**AFI+ILS**		**ARI+ILS**	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
berlin52	0.2	1	0.95	——	——	——	——	——	——
			∞	4.49% *	7.62	4.49% *	8.41	4.65% *	17.03
		1.33	0.95	4.43% *	9.95	4.49% *	8.02	4.41% *	12.73
			∞	5.45% *	6.71	5.47% *	5.11	5.38% *	10.72
		1.67	0.95	——	——	1.04%	9.48	0.52%	18.08
			∞	1.77%	7.44	1.77%	7.05	2.02%	11.91
	0.35	1	0.95	0.53%	7.55	0.33%	6.44	0.5%	11.4
			∞	0.41%	7.59	0.18%	6.84	0.08%	15.49
		1.33	0.95	0.47%	7.71	0.47%	7.8	1.91%	11.51
			∞	0.75%	5.71	0.0%	6.26	1.12%	12.04
		1.67	0.95	0.0%	6.18	0.47%	5.45	0.73%	10.05
			∞	0.0%	6.17	0.47%	5.57	1.06%	10.32
	0.5	1	0.95	——	——	——	——	——	——
			∞	0.08%	5.47	0.0%	5.01	0.53%	10.15
		1.33	0.95	——	——	1.07%	7.3	1.03%	19.76
			∞	0.94%	5.68	0.29%	4.13	0.23%	8.49
		1.67	0.95	1.67%	6.75	0.47%	6.16	0.47%	8.09
			∞	0.96%	5.02	0.53%	5.03	0.1%	10.64
pr76	0.2	1	0.95	——	——	——	——	——	——
			∞	1.45% *	13.22	1.81% *	14.8	1.75% *	36.52
		1.33	0.95	1.54% *	15.84	3.16% *	19.66	3.86% *	23.5
			∞	1.45% *	15.56	2.14% *	12.92	1.43% *	23.24
		1.67	0.95	0.78% *	13.01	0.46% *	14.52	0.63% *	28.86
			∞	0.62% *	12.04	0.37% *	14.38	0.52% *	27.25
	0.35	1	0.95	——	——	——	——	4.22% *	32.67
			∞	4.36% *	17.03	4.28% *	15.44	4.36% *	28.49
		1.33	0.95	0.02%	11.57	0.02%	9.75	0.08%	21.19
			∞	0.05%	8.44	0.14%	11.79	0.03%	20.39
		1.67	0.95	0.72%	13.56	0.03%	12.23	0.47%	29.48
			∞	0.54%	11.44	0.09%	10.31	0.2%	22.6
	0.5	1	0.95	——	——	——	——	——	——
			∞	0.0%	13.74	2.7%	15.09	0.02%	34.97
		1.33	0.95	——	——	1.53%	11.5	1.16%	25.15
			∞	1.05%	8.99	1.65%	12.02	0.55%	29.56

Instance				ANN+ILS		AFI+ILS		ARI+ILS	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
pr76	0.5	1.67	0.95	0.55%	7.28	———	———	0.57%	20.75
			∞	0.61%	9.74	0.09%	10.28	0.68%	18.25
kroA100	0.2	1	0.95	———	———	———	———	———	———
			∞	9.05% *	70.24	9.46% *	67.07	8.76% *	184.62
		1.33	0.95	———	———	0.0%	20.35	0.0%	27.06
			∞	0.02%	16.09	0.0%	15.47	0.01%	34.19
		1.67	0.95	———	———	0.0%	17.54	0.01%	31.41
			∞	0.13%	18.03	0.0%	16.21	0.02%	32.87
	0.35	1	0.95	———	———	———	———	———	———
			∞	6.48% *	55.48	6.33% *	53.12	6.47% *	125.12
		1.33	0.95	———	———	4.23% *	26.07	3.59% *	81.35
			∞	1.6% *	40.3	1.57% *	57.97	2.55% *	85.45
		1.67	0.95	0.01%	14.08	0.0%	16.0	0.02%	33.47
			∞	0.02%	18.84	0.02%	20.44	0.02%	35.63
	0.5	1	0.95	———	———	———	———	———	———
			∞	3.06% *	69.86	3.21% *	53.2	3.2% *	104.57
		1.33	0.95	2.04% *	56.0	1.92% *	76.34	3.4% *	72.75
			∞	1.79% *	37.56	2.0% *	33.82	1.79% *	63.69
		1.67	0.95	0.15%	16.16	0.0%	18.86	0.0%	38.74
			∞	0.0%	18.98	0.0%	21.25	0.0%	35.25
pr124	0.2	1	0.95	———	———	6.93% *	46.74	———	———
			∞	6.06% *	43.69	5.95% *	32.52	6.52% *	97.4
		1.33	0.95	10.06% *	20.79	9.44% *	20.89	9.44% *	42.23
			∞	9.61% *	20.65	9.27% *	28.99	9.61% *	45.28
		1.67	0.95	10.93% *	17.66	10.93% *	20.13	11.4% *	37.46
			∞	11.0% *	16.08	11.0% *	18.03	11.1% *	37.11
	0.35	1	0.95	———	———	———	———	———	———
			∞	9.67% *	23.6	8.4% *	20.94	8.4% *	31.37
		1.33	0.95	———	———	———	———	1.32%	39.92
			∞	0.35%	23.74	0.42%	20.18	0.24%	50.29
		1.67	0.95	———	———	7.27% *	31.5	6.99% *	70.67
			∞	6.53% *	17.9	6.61% *	22.25	5.92% *	47.08
	0.5	1	0.95	———	———	———	———	———	———
			∞	5.31% *	50.89	4.7% *	36.48	4.9% *	83.85
		1.33	0.95	———	———	2.15% *	30.89	2.95% *	49.85
			∞	4.58% *	27.22	4.58% *	27.31	4.34% *	42.9
		1.67	0.95	———	———	0.29%	18.54	0.3%	44.65
			∞	0.0%	16.2	0.36%	27.35	0.59%	40.12
pr152	0.2	1	0.95	14.18% *	95.32	———	———	13.99% *	133.75
			∞	14.38% *	83.19	14.54% *	107.5	14.17% *	152.64
		1.33	0.95	14.0% *	70.05	14.31% *	86.9	14.43% *	210.35
			∞	13.27% *	90.94	13.58% *	82.15	13.72% *	161.84

Instance				ANN+ILS		AFI+ILS		ARI+ILS	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
pr152	0.2	1.67	0.95	13.14% *	60.19	12.97% *	34.82	13.61% *	99.38
			∞	12.95% *	46.68	12.99% *	44.15	12.68% *	101.39
	0.35	1	0.95	24.24% *	88.97	———	———	24.21% *	96.48
			∞	24.67% *	94.67	24.07% *	74.12	23.71% *	150.04
		1.33	0.95	19.63% *	34.82	18.82% *	50.19	18.71% *	86.32
			∞	21.19% *	39.09	20.32% *	42.69	20.55% *	62.26
		1.67	0.95	14.68% *	49.3	———	———	14.07% *	81.3
			∞	14.77% *	31.16	14.64% *	39.66	14.25% *	86.62
	0.5	1	0.95	———	———	———	———	———	———
			∞	9.56% *	42.01	9.14% *	47.87	12.35% *	113.56
		1.33	0.95	9.67% *	40.81	9.21% *	37.15	9.08% *	93.77
			∞	9.4% *	34.46	8.89% *	52.69	9.01% *	73.27
		1.67	0.95	———	———	———	———	———	———
			∞	23.61% *	36.07	22.93% *	47.79	22.82% *	77.01
rat195	0.2	1	0.95	———	———	———	———	———	———
			∞	5.57% *	259.63	7.04% *	312.71	6.72% *	579.07
		1.33	0.95	———	———	———	———	———	———
			∞	3.55% *	150.65	4.54% *	157.73	3.8% *	350.28
		1.67	0.95	3.56% *	120.69	4.2% *	134.16	3.26% *	327.07
			∞	3.32% *	114.44	4.42% *	150.29	4.37% *	280.9
	0.35	1	0.95	———	———	———	———	———	———
			∞	3.62% *	157.75	3.75% *	202.63	3.7% *	400.48
		1.33	0.95	2.46% *	108.91	4.44% *	135.64	3.43% *	315.57
			∞	2.77% *	119.97	4.06% *	155.09	3.76% *	318.29
		1.67	0.95	———	———	———	———	———	———
			∞	3.09% *	119.07	4.03% *	146.68	3.52% *	296.32
	0.5	1	0.95	———	———	———	———	———	———
			∞	2.51% *	188.67	4.0% *	141.34	3.49% *	334.12
		1.33	0.95	1.96% *	181.53	2.21% *	266.81	3.61% *	299.24
			∞	2.21% *	115.46	3.92% *	165.56	3.12% *	268.23
		1.67	0.95	2.44% *	155.32	3.48% *	134.59	3.89% *	426.2
			∞	2.36%	98.26	3.19%	160.67	3.31%	269.86
pr226	0.2	1	0.95	24.52% *	213.85	———	———	23.4% *	1018.22
			∞	22.76% *	273.37	25.08% *	310.7	25.16% *	715.39
		1.33	0.95	———	———	10.44% *	222.35	10.72% *	426.14
			∞	10.74% *	204.51	10.47% *	182.47	9.48% *	714.83
		1.67	0.95	7.2% *	161.58	6.78% *	162.55	7.01% *	239.59
			∞	7.09% *	149.36	6.94% *	134.94	6.98% *	239.21
	0.35	1	0.95	19.4% *	289.47	———	———	19.5% *	639.84
			∞	17.41% *	201.95	14.61% *	252.52	15.26% *	523.0
		1.33	0.95	———	———	11.4% *	166.1	11.46% *	242.18
			∞	11.02% *	140.79	10.63% *	174.28	10.59% *	301.08

Instance				ANN+ILS		AFI+ILS		ARI+ILS	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
pr226	0.35	1.67	0.95	———	———	9.18% *	201.06	9.54% *	185.3
			∞	9.07% *	157.68	8.85% *	150.8	9.68% *	232.86
	0.5	1	0.95	15.33% *	628.23	———	———	———	———
			∞	19.47% *	264.38	19.43% *	383.71	19.75% *	510.61
		1.33	0.95	10.86% *	246.07	11.58% *	186.87	14.37% *	328.52
			∞	12.11% *	207.03	10.6% *	131.32	10.95% *	212.98
		1.67	0.95	———	———	11.23% *	153.58	11.31% *	230.0
			∞	10.65% *	132.26	10.6% *	95.24	11.0% *	207.82

Asymmetric instances

Instance				ANN+ILS		AFI+ILS		ARI+ILS	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
ft53	0.2	1	0.95	17.39% *	14.1	———	———	14.01% *	35.44
			∞	16.52% *	18.96	14.44% *	15.23	14.97% *	39.62
		1.33	0.95	3.91%	11.01	4.27%	8.59	5.92%	17.4
			∞	5.18%	6.55	5.77%	7.18	4.43%	21.4
		1.67	0.95	2.03%	5.1	2.44%	6.23	3.09%	13.32
			∞	1.54%	6.32	2.46%	5.4	2.3%	12.57
	0.35	1	0.95	6.16%	12.45	———	———	6.17%	15.16
			∞	2.14%	15.35	3.56%	13.31	2.97%	20.22
		1.33	0.95	5.12%	10.43	3.7%	9.38	3.36%	19.86
			∞	4.01%	10.49	2.92%	13.94	4.79%	14.32
		1.67	0.95	2.25%	4.98	1.12%	5.27	2.62%	10.71
			∞	2.09%	4.68	1.26%	4.83	1.95%	12.38
	0.5	1	0.95	2.34%	9.17	3.62%	5.34	1.13%	17.83
			∞	4.72%	8.96	4.43%	5.69	6.0%	11.41
		1.33	0.95	0.89%	6.38	2.74%	5.86	2.23%	8.57
			∞	1.98%	5.48	1.84%	6.33	1.94%	8.77
		1.67	0.95	1.86%	5.11	2.76%	4.97	5.38%	9.45
			∞	1.27%	4.19	1.13%	5.42	2.29%	10.69
ftv64	0.2	1	0.95	———	———	6.73%	13.97	6.05%	22.32
			∞	6.46%	17.08	7.17%	10.9	6.55%	22.92
		1.33	0.95	———	———	4.47%	6.53	3.09%	12.28
			∞	3.89%	6.65	3.28%	8.36	5.04%	12.23
		1.67	0.95	4.6%	9.5	———	———	5.38%	12.75
			∞	3.99%	8.96	2.66%	5.97	4.98%	11.22
	0.35	1	0.95	2.81%	11.14	2.87%	12.65	4.5%	24.15
			∞	4.51%	9.17	3.5%	11.0	3.1%	25.52
		1.33	0.95	3.94%	7.74	2.78%	7.49	4.25%	16.05
			∞	3.39%	5.74	1.97%	7.98	3.48%	11.35
		1.67	0.95	2.32%	5.2	4.05%	3.86	3.03%	11.94
			∞	2.56%	5.6	2.81%	6.09	3.37%	10.3

Instance				ANN+ILS		AFI+ILS		ARI+ILS	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
ftv64	0.5	1	0.95	3.22%	13.06	———	———	1.03%	29.89
			∞	4.81%	9.02	1.75%	9.13	3.45%	14.59
		1.33	0.95	3.7%	6.16	1.27%	5.17	3.15%	10.61
			∞	2.28%	7.41	2.6%	5.57	2.54%	14.43
		1.67	0.95	2.61%	4.69	———	———	1.78%	14.3
			∞	0.71%	4.57	2.13%	4.47	2.39%	9.57
ft70	0.2	1	0.95	6.42% *	19.99	5.98% *	28.18	6.33% *	39.15
			∞	6.2% *	22.0	5.46% *	35.49	5.24% *	40.24
		1.33	0.95	3.98%	25.06	4.19%	22.02	3.62%	44.93
			∞	3.15%	17.93	2.46%	25.48	3.02%	38.7
		1.67	0.95	3.9%	24.65	3.41%	20.37	3.72%	35.59
			∞	3.84%	16.94	2.69%	17.67	3.03%	30.09
	0.35	1	0.95	3.28%	22.47	2.8%	21.67	2.52%	36.07
			∞	2.85%	24.31	3.11%	18.34	3.16%	45.98
		1.33	0.95	3.77%	18.69	2.81%	17.91	2.58%	35.88
			∞	2.43%	18.84	2.7%	25.09	3.46%	31.39
		1.67	0.95	———	———	3.41%	17.15	2.99%	25.1
			∞	2.75%	21.34	3.38%	16.78	3.26%	32.48
	0.5	1	0.95	———	———	3.09%	28.2	1.43%	150.83
			∞	3.4%	13.1	2.86%	18.01	3.26%	32.39
		1.33	0.95	———	———	3.4%	16.24	3.37%	24.13
			∞	3.09%	16.64	2.83%	15.01	3.17%	23.58
		1.67	0.95	———	———	3.15%	16.39	3.55%	30.66
			∞	2.74%	17.77	2.99%	18.15	3.46%	25.75
kro124p	0.2	1	0.95	———	———	6.36% *	45.32	8.04% *	91.73
			∞	9.21% *	54.72	6.04% *	57.84	8.76% *	100.51
		1.33	0.95	5.11%	31.54	1.96%	24.16	3.86%	43.09
			∞	4.66%	27.79	2.1%	27.81	1.86%	47.02
		1.67	0.95	7.94%	31.27	1.14%	31.86	3.48%	50.46
			∞	6.26%	32.83	1.38%	24.19	4.35%	52.33
	0.35	1	0.95	———	———	———	———	———	———
			∞	8.48% *	66.12	9.84% *	76.96	10.16% *	129.33
		1.33	0.95	5.47%	50.36	2.08%	33.67	1.03%	68.39
			∞	3.98%	36.04	1.65%	30.12	1.83%	74.11
		1.67	0.95	7.63%	34.69	1.49%	25.48	1.74%	48.99
			∞	8.31%	26.13	3.74%	19.69	1.9%	63.53
	0.5	1	0.95	———	———	4.04% *	41.89	4.36% *	60.84
			∞	8.18% *	65.2	4.9% *	43.52	5.6% *	99.93
		1.33	0.95	7.36%	40.26	4.33%	32.61	2.35%	98.6
			∞	8.62%	42.8	4.6%	39.21	3.39%	84.63
		1.67	0.95	5.89%	42.81	2.79%	29.3	2.22%	73.58
			∞	7.61%	28.53	2.21%	26.49	2.95%	36.46

Instance				**ANN+ILS**		**AFI+ILS**		**ARI+ILS**	
Matrix	α	β	γ	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**	**Av. Gap**	**Av. CPU**
ftv170	0.2	1	0.95	———	———	———	———	———	———
			∞	14.71% *	147.31	7.24% *	134.46	14.6% *	192.87
		1.33	0.95	8.93%	61.53	3.81%	72.52	4.89%	124.11
			∞	8.69%	72.25	4.79%	54.44	6.15%	93.23
		1.67	0.95	8.55%	54.25	6.58%	52.21	6.38%	119.05
			∞	10.17%	52.86	3.05%	73.86	7.45%	79.12
	0.35	1	0.95	———	———	11.63% *	91.66	12.13% *	217.51
			∞	7.12% *	105.43	11.41% *	101.41	9.71% *	228.71
		1.33	0.95	———	———	4.11%	51.59	9.16%	114.31
			∞	9.52%	56.95	6.37%	80.32	7.28%	128.3
		1.67	0.95	8.42%	38.32	6.03%	56.42	4.23%	78.42
			∞	6.29%	65.77	3.91%	44.95	8.13%	100.99
	0.5	1	0.95	———	———	9.81% *	140.17	9.28% *	213.35
			∞	6.72% *	71.2	9.6% *	126.29	10.39% *	143.61
		1.33	0.95	———	———	6.64%	82.85	8.68%	162.58
			∞	7.91%	48.68	10.16%	80.0	6.58%	114.47
		1.67	0.95	8.47%	54.77	3.95%	47.63	6.73%	131.88
			∞	6.85%	48.72	4.44%	48.42	3.48%	124.18

The ILS heuristic requires more CPU time whenever its starting point is determined by the ARI procedure, which is the only non-deterministic constructive heuristic we used. From Table 5.7, we are able to confirm that this behavior is consistent throughout all the instances we considered. In fact, the combination ARI+ILS tends to consume, at least, twice the average computational time of the combinations ANN+ILS and AFI+ILS. Let us consider the average CPU times we obtained for the symmetric instance which was obtained from the matrix titled "'pr226", for $\alpha = 0.2$, $\beta = 1$ and $\gamma = 0.95$. The AFI heuristic could not determine a feasible solution for this instance, so we can only compare the final solution of the ILS when we use initial solutions determined by the ANN and the ARI heuristic. It is possible to see that it took an average of 1018.22 seconds to run the combination ARI+ILS, whereas it only took an average of 213.85 seconds to run the combination ANN+ILS for the same exact instance. This is an even more significant difference because using an initial solution provided by the ARI method took us, not twice, but five times the average CPU time of running the ILS heuristic when it started from the initial solution the ANN method provided.

Let us consider the 18 different instances we obtained through each matrix we considered, either symmetric or asymmetric. For each matrix and value α we considered, it is possible to see that the average CPU time to run the ILS algorithm decreases as the value of the parameter β increases. This decrease in computational time happens regardless of the constructive method we decide to adopt before the ILS. It is important to remember that β regulates the number of black vertices we consider in the instance as a function of the total number of nodes in the graph, n, and Q $(= \lceil \alpha n \rceil)$.

The constructive heuristic we used before the ILS only appears to have a greater impact on the quality of the final solution in asymmetric instances of the BWTSP with a larger number of nodes. Nevertheless, the average of the real gaps we presented in Table 5.7 (which are not marked with *) indicate that the value of the solutions we obtained through the ILS heuristic are close to the global optimum for each instance we considered. For symmetric instances, the average real gaps are less than 3.5% and the vast majority of them is even less than 1%. However, for asymmetric instances, the average real gaps are less than 9% and the majority of them is less than 6%. The average real gaps we previously displayed suggest that our ILS algorithm determines solutions with a better quality for the symmetric case of the BWTSP.

The detailed results in regards to the gaps, and overestimations of those values in some cases, we obtained for the 5 different runs of the ANN+ILS, AFI+ILS and ARI+ILS, relatively, for each instance we considered, show that the difference between the maximum and the minimum gap of each instance is more accentuated in the asymmetric case. This information can be consulted in Appendix B.2.2. The resultant maximum and minimum gaps for each symmetric instance rarely differ more than 2% amongst each other and thus they do not differ significantly from the average gap. This is not the case for asymmetric instances: in some cases, the maximum and minimum gaps differ in 9%. Therefore, the ILS consistently returns solutions with an associated value very close to the global minimum for symmetric instances of the BWTSP, whereas the average gaps on the asymmetric cases are not as low and as consistent relatively to the values we obtained for the conducted tests on symmetric instances of the BWTSP. This is a consequence of the conclusion we mentioned in the previous section, which is that the Local Search procedure we incorporated on the ILS heuristic is more impactful on symmetric instances, making them less dependent on the quality and quantity of our perturbations to visit new local minimums, which is important to enable the ILS to decrease its incumbent value.

In comparison to the results we displayed in Section 5.2 regarding the quality of the feasible solutions for the BWTSP provided by the ANN, AFI and ARI heuristics, it is possible to verify a massive improvement in the quality of the final solutions we obtained for each instance after applying the ILS heuristic, for both symmetric and asymmetric instances.

Since we are overestimating the gap between the heuristic results we obtained and the optimal value of some instances, specially the ones with a larger number of nodes, it is hard to evaluate the performance of the ILS under these circumstances. Therefore, we have decided to additionally compare the quality of the best solution we obtained through the ILS heuristic and the cost of the best integer solution CPLEX determined in 18000 seconds (which corresponds to the upper bound of the optimal value), for all the instances we could not determine the optimal value in that time limit, with the associated lower bound. The lower and upper bounds we obtained through the solver were presented in Section 5.1.1.

Table 5.8: Gaps of the upper bounds, as well as the best solutions we obtained through the ILS algorithm, relatively to the lower bound.

Symmetric instances							
Instance				**Gap relatively to the lower bound**			
Matrix	α	β	γ	**CPLEX upper bound**	**ANN+ILS best solution**	**AFI+ILS best solution**	**ARI+ILS best solution**
berlin52	0.2	1	0.95	———	———	———	———
			∞	13.78%	4.49%	4.49%	4.49%
		1.33	0.95	6.12%	4.4%	4.4%	4.4%
			∞	6.5%	5.38%	5.38%	5.38%
pr76	0.2	1	0.95	———	———	———	———
			∞	1.38%	1.45%	1.45%	1.45%

Instance				**Gap relatively to the lower bound**			
Matrix	α	β	γ	**CPLEX upper bound**	**ANN+ILS best solution**	**AFI+ILS best solution**	**ARI+ILS best solution**
pr76	0.2	1.33	0.95	1.03%	1.54%	1.03%	3.86%
			∞	1.04%	1.04%	1.04%	1.11%
		1.67	0.95	0.39%	0.5%	0.39%	0.5%
			∞	0.33%	0.51%	0.33%	0.51%
	0.35	1	0.95	5.02%	—	—	4.22%
			∞	4.13%	4.13%	4.13%	4.13%
kroA100	0.2	1	0.95	—	—	—	—
			∞	27.42%	8.25%	8.25%	8.25%
	0.35	1	0.95	—	—	—	—
			∞	13.54%	6.33%	6.33%	6.33%
		1.33	0.95	—	—	2.79%	2.44%
			∞	1.56%	1.56%	1.56%	1.56%
	0.5	1	0.95	—	—	—	—
			∞	2.41%	2.57%	2.62%	2.41%
		1.33	0.95	2.16%	1.76%	1.76%	1.76%
			∞	1.76%	1.78%	1.78%	1.78%
pr124	0.2	1	0.95	—	—	6.84%	—
			∞	5.58%	5.58%	5.04%	6.06%
		1.33	0.95	—	9.44%	9.44%	9.44%
			∞	20.99%	9.23%	9.23%	9.3%
		1.67	0.95	35.6%	10.93%	10.93%	10.93%
			∞	—	11.0%	11.0%	11.0%
	0.35	1	0.95	—	—	—	—
			∞	10.88%	8.4%	8.4%	8.4%
		1.67	0.95	—	—	6.69%	6.69%
			∞	9.49%	5.92%	5.92%	5.92%
	0.5	1	0.95	—	—	—	—
			∞	4.32%	4.81%	4.32%	4.32%
		1.33	0.95	1.98%	—	1.98%	1.98%
			∞	4.33%	4.33%	4.33%	4.33%
pr152	0.2	1	0.95	—	13.54%	—	13.99%
			∞	—	13.98%	13.95%	14.02%
		1.33	0.95	—	13.55%	13.43%	14.16%
			∞	—	13.21%	13.19%	13.19%
		1.67	0.95	—	13.07%	12.48%	13.1%
			∞	—	12.95%	12.36%	12.36%
	0.35	1	0.95	—	23.31%	—	24.02%
			∞	62.05%	23.56%	22.65%	22.65%
		1.33	0.95	—	19.36%	18.38%	18.38%
			∞	—	21.19%	19.94%	19.85%

Instance				Gap relatively to the lower bound			
Matrix	α	β	γ	**CPLEX upper bound**	**ANN+ILS best solution**	**AFI+ILS best solution**	**ARI+ILS best solution**
pr152	0.35	1.67	0.95	———	14.64%	———	14.07%
			∞	328.14%	14.77%	14.1%	13.89%
	0.5	1	0.95	———	———	———	———
			∞	23.2%	9.56%	8.65%	8.65%
		1.33	0.95	———	8.64%	9.04%	8.44%
			∞	17.81%	9.34%	8.43%	8.43%
		1.67	0.95	———	———	———	———
			∞	———	23.61%	22.91%	22.24%
rat195	0.2	1	0.95	———	———	———	———
			∞	———	4.53%	5.43%	5.61%
		1.33	0.95	———	———	———	———
			∞	———	2.82%	4.07%	2.6%
		1.67	0.95	———	2.91%	2.29%	2.38%
			∞	———	2.6%	3.85%	3.94%
	0.35	1	0.95	7.46%	———	———	———
			∞	14.23%	2.44%	3.01%	1.77%
		1.33	0.95	———	1.56%	4.09%	2.98%
			∞	100.18%	2.14%	3.61%	2.27%
		1.67	0.95	———	———	———	———
			∞	———	2.6%	3.0%	1.88%
	0.5	1	∞	0.26%	1.28%	2.86%	2.46%
		1.33	0.95	0.09%	0.57%	0.88%	3.04%
			∞	3.51%	1.86%	2.4%	2.09%
		1.67	0.95	———	2.05%	2.41%	2.23%
pr226	0.2	1	0.95	———	22.1%	———	21.47%
			∞	———	22.31%	23.19%	22.35%
		1.33	0.95	———	———	9.19%	9.28%
			∞	———	9.36%	9.07%	9.25%
		1.67	0.95	———	6.72%	6.72%	6.73%
			∞	———	6.84%	6.85%	6.89%
	0.35	1	0.95	———	17.99%	———	14.09%
			∞	55.29%	15.23%	14.51%	14.51%
		1.33	0.95	———	———	11.34%	11.33%
			∞	18.02%	10.48%	10.48%	10.48%
		1.67	0.95	———	———	9.12%	9.18%
			∞	19.11%	8.89%	8.76%	8.8%
	0.5	1	0.95	———	14.7%	———	———
			∞	76.69%	17.32%	19.16%	19.01%
		1.33	0.95	———	10.48%	11.21%	11.4%
			∞	79.02%	10.57%	10.55%	10.52%

Instance				Gap relatively to the lower bound			
Matrix	α	β	γ	**CPLEX upper bound**	**ANN+ILS best solution**	**AFI+ILS best solution**	**ARI+ILS best solution**
pr226	0.5	1.67	0.95	—	—	11.04%	11.05%
			∞	—	10.52%	10.55%	10.57%
Asymmetric instances							
Instance				**Gap relatively to the lower bound**			
Matrix	α	β	γ	**CPLEX upper bound**	**ANN+ILS best solution**	**AFI+ILS best solution**	**ARI+ILS best solution**
ft53	0.2	1	0.95	17.11%	13.57%	—	11.52%
			∞	18.15%	11.58%	11.58%	12.62%
ft70	0.2	1	0.95	—	—	4.48%	5.19%
			∞	2.89%	5.5%	4.14%	3.58%
kro124p	0.2	1	0.95	—	—	4.58%	7.52%
			∞	19.72%	6.83%	4.81%	6.81%
	0.35	1	0.95	—	—	—	—
			∞	9.34%	6.47%	8.37%	6.59%
	0.5	1	0.95	0.68%	—	2.49%	3.39%
			∞	0.64%	3.74%	2.73%	0.85%
ftv170	0.2	1	0.95	42.77%	—	—	—
			∞	56.78%	8.13%	5.0%	6.49%
	0.35	1	0.95	4.49%	—	6.31%	8.02%
			∞	2.8%	4.07%	7.81%	5.19%
	0.5	1	0.95	1.16%	—	7.54%	5.77%
			∞	7.24%	4.99%	8.41%	6.37%

Let us analyse the results in Table 5.8. The values of the best solutions we obtained through the three different combinations ANN+ILS, AFI+ILS and ARI+ILS are very similar, except for the asymmetric instances which were obtained through the matrices "kro124p" and "ftv170". This is not a surprise since we previously concluded that the ILS appears to be more dependent on its initial solution whenever we are solving asymmetric instances with a larger number of nodes.

The best solutions we obtained through the ILS algorithm tend to be similar to the cheapest integer solution determined by CPLEX after 5 hours of solving the instances we considered, or they are solutions with a much more reduced cost in the objective function. This is a really good indicator, mainly because the CPU times we reported in our computational study regarding the ILS heuristic never exceeded 20 minutes. However, when the gap between CPLEX upper and lower bounds is less than 1.8%, for symmetric instances, or 8%, for asymmetric instances, the value of at least one of the three best solutions we obtained through the ILS is greater than the value of the cheapest integer solution CPLEX determined; nevertheless, the difference in these cases is never greater than 5%.

For some instances of the BWTSP, CPLEX could not determine a feasible integer solution within the time limit of 5 hours, but at least one of our constructive heuristics returned a feasible solution for that same instance. In other cases, neither CPLEX nor our constructive heuristics determined feasible integer solutions for the corresponding BWTSP instance, which makes us wonder whether these instances are even feasible or not, even though they were never proven to be infeasible.

Chapter 6

Conclusions

The goal of this dissertation was to contribute to the expansion of the existent heuristics for the BWTSP. We wanted to develop methods for both symmetric and asymmetric instances of the BWTSP, which had not been previously done in the literature due to the few scientific articles studying this variant of the TSP.

We developed three constructive heuristics, the Adapted Nearest Neighbor (ANN), the Adapted Farthest Insertion (AFI) and the Adapted Random Insertion (ARI). The first two heuristics are deterministic, and the third one has a random component to it. These heuristics result in adaptations on the selection methods of the Nearest Neighbor, Farthest Insertion and Random Insertion, which are constructive heuristics for the classical TSP. The ANN, AFI and ARI heuristics can be separated in two stages: a construction stage and a correction stage. In the construction stage, a Hamiltonian circuit is determined according to the selection and insertion methods of each heuristic. This first stage is what sets the difference between the three constructive algorithms we proposed, because the correction stage is common amongst the three and follows the same exact procedures. If the Hamiltonian circuit we obtained after the construction stage is infeasible for the BWTSP, we start by establishing cardinality feasibility and then we attempt to correct length infeasibility. We were able to conclude from our computational study (Chapter 5) that the AFI heuristic consumes significantly more CPU time, in average, to determine a feasible solution for the BWTSP in comparison to the other two methods, being the ANN the fastest of the three heuristics. The AFI and ARI heuristics proved to be more successful in determining feasible solutions for the BWTSP instances; nevertheless, the ANN algorithm still managed to determine feasible solutions for, at least, 3/4 of our sample of BWTSP instances, which is pretty remarkable. Another important observation is that whenever we eliminated the length constraint of the BWTSP, our constructive heuristics were always able to determine feasible solutions for the considered instances, which proves that the algorithm we used to correct length infeasibility is our main limitation to achieving feasible solutions when cardinality and length constraints are simultaneously considered.

We also developed an Iterated Local Search (ILS) algorithm as an improvement heuristic for the BWTSP. This algorithm applies a Local Search procedure to the initial solution we consider, and proceeds to apply the same Local Search procedure onto successive perturbations which were obtained either from the current or the incumbent solution of our ILS algorithm. In our computational study, we tested six different combinations of parameters for the ILS and compared the results to justify our particular choice of parameters. These parameters control the strength of our perturbation method and the maximum number of iterations of the algorithm. We also studied the performance of the ILS. We were able to see that the algorithm determined solutions with associated values which were close to the global minimum, however the gaps between the heuristic solutions and the corresponding optimal values were smaller for symmetric instances. When we ran the ILS multiple times for the same instance, the differences between the maximum and minimum gaps were less significant in the symmetric cases. Nevertheless, the algorithm proved to be efficient since it determined solutions with a relative good quality while not consuming a lot of computational time. It is also important to mention that for the sample of BWTSP instances we used where CPLEX could not determine the global optimum within our time limit of 5 hours, our heuristic results tended to be similar to the cost of the cheapest integer solution determined by CPLEX after solving the given instance for 5 hours, or significantly less in almost half of the cases. In these cases, it was not possible to compare our heuristic solutions with the optimal ones, however, if the ILS returned solutions with similar or better quality than the best integer solutions determined by CPLEX after 5 hours knowing that the ILS required at least 15 times less computational time, then this is a good indicator of the efficiency of our improvement heuristic.

6.1 Future work

One of the most evident limitations we faced throughout the computational study of this dissertation was the fact that the optimal values of some instances had not been previously determined, specially larger sized instances. It would be useful to know the global minimums of every BWTSP instance we considered in this dissertation, which could be achieved by imposing a larger time limit on each problem we solved in CPLEX. Such information would be important to get a more accurate perception on the quality of the heuristic results we obtained. Furthermore, it would also be interesting to test the heuristics we developed in symmetric and asymmetric instances with more than 250 nodes.

One of the most important conclusions from our computational study was that asymmetric BWTSP instances were more sensitive to the quality and quantity of perturbations we applied during the ILS algorithm in comparison to the symmetric instances we considered. Future computational tests could be conducted where we allow more iterations on the ILS heuristic in case our instance is asymmetric, while maintaining the same maximum number of iterations to the symmetric case since the ILS proved to be efficient in this scenario. Alternatively, for both symmetric and asymmetric BWTSP instances, we could set the maximum number of iterations of the ILS as a function of the number of nodes in the graph, n, instead of defining it as a constant, which is 2500 in the algorithm we developed. This alternative suggestion is motivated by the noticeable difference between the maximum and minimum gaps we reported for instances with larger sizes, even though this occured more frequently in asymmetric instances. If we do not want to change the perturbation method or the maximum number of iterations of the ILS, we could focus on defining different neighborhoods for the BWTSP and redefine the Local Search procedure we have proposed. This new Local Search procedure could decrease the difference regarding the performance of the ILS between symmetric and asymmetric BWTSP instances.

There were BWTSP instances where neither CPLEX, considering a time limit of 5 hours, nor our constructive heuristic were able to determine feasible solutions for them. It would be important to pay close attention to these cases and analyse whether they are in fact feasible instances for the BWTSP or not. If they actually happen to be infeasible, then they should not be counted in our statistics regarding the success rate of the constructive methods we developed in this dissertation. Note that we can only consider a constructive heuristic unsuccessful on determining feasible solutions for an instance if the given instance is feasible in the first place. If not, then not determining a feasible solution does not qualify as an "unsuccess". Nevertheless, we should strive to improve our methods to correct potential cardinality and length infeasibilities to increase the reliability of our constructive heuristics to determine feasible solutions for feasible BWTSP instances, either symmetric or asymmetric.

Finally, we implemented all our procedures in the Python language, which is an interpreted programming language. The computational times we reported could be even lower if we had implemented our heuristics in a compiled language, such as C or C++, for example.

Bibliography

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. Accessed on 10/04/2022.

Mélanie Bourgeois, Gilbert Laporte, and Frédéric Semet. Heuristics for the black and white traveling salesman problem. *Computers & Operations Research*, 30(1):75–85, 2003. ISSN 0305-0548. doi: https://doi.org/10.1016/S0305-0548(01)00082-X.

Lloyd Clarke, Ellis Johnson, George Nemhauser, and Zhongxi Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69, 01 1997. doi: https://doi.org/10.1023/A:1018945415148.

Martin Desrochers and Gilbert Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991. ISSN 0167-6377. doi: https://doi.org/10.1016/0167-6377(91)90083-2.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979. ISBN 0-7167-1045-5.

Gianpaolo Ghiani, Gilbert Laporte, and Frédéric Semet. The black and white traveling salesman problem. *Operations Research*, 54(2):366–378, 2006. doi: https://doi.org/10.1287/opre.1050.0218.

Luís Gouveia, Markus Leitner, and Mario Ruthmair. Extended formulations and branch-and-cut algorithms for the black-and-white traveling salesman problem. *European Journal of Operational Research*, 262(3):908–928, 2017. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.04.061.

Haitao Li and Bahram Alidaee. Tabu search for solving the black-and-white travelling salesman problem. *The Journal of the Operational Research Society*, 67(8):1061–1079, 2016. ISSN 01605682, 14769360.

Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. *Iterated Local Search*. Springer US, Boston, MA, 2003. ISBN 978-0-306-48056-0. doi: https://doi.org/10.1007/0-306-48056-5_11.

Vicky Mak and Natashia Boland. Heuristic approaches to the asymmetric travelling salesman problem with replenishment arcs. *International Transactions in Operational Research*, 7(4):431–447, 2000. ISSN 0969-6016. doi: https://doi.org/10.1016/S0969-6016(00)00006-X.

Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. doi: https://doi.org/10.1137/0206041.

Kalyan T. Talluri. The four-day aircraft maintenance routing problem. *Transportation Science*, 32(1):43–53, 1998. doi: https://doi.org/10.1287/trsc.32.1.43.

O.J. Wasem. An algorithm for designing rings for survivable fiber networks. *IEEE Transactions on Reliability*, 40(4):428 – 432, 1991. doi: https://doi.org/10.1109/24.93762. Cited by: 51.

İbrahim Muter. A new formulation and approach for the black and white traveling salesman problem. *Computers & Operations Research*, 53:96–106, 2015. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2014.07.019.

Appendix A

Characterization of instances

This appendix contains the set of black vertices for each BWTSP instance we considered for the computational study, as well as the corresponding values for Q and L, which were determined following the procedures we detailed in Section 5.1.

Table A.1: Characterization of every instance we used for the computational tests in the dissertation.

Symmetric instances						
Instance				**Characterization**		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
berlin52	0.2	1	0.95	11	2261.0	$\{1, 52, 33, 2, 47\}$
			$+\infty$	11	$+\infty$	$\{1, 52, 33, 2, 47\}$
		1.33	0.95	11	2334.15	$\{1, 52, 33, 2, 47, 9, 12\}$
			$+\infty$	11	$+\infty$	$\{1, 52, 33, 2, 47, 9, 12\}$
		1.67	0.95	11	1597.9	$\{1, 52, 33, 2, 47, 9, 12, 17, 29\}$
			$+\infty$	11	$+\infty$	$\{1, 52, 33, 2, 47, 9, 12, 17, 29\}$
	0.35	1	0.95	19	4233.2	$\{1, 52, 33\}$
			$+\infty$	19	$+\infty$	$\{1, 52, 33\}$
		1.33	0.95	19	2963.05	$\{1, 52, 33, 2\}$
			$+\infty$	19	$+\infty$	$\{1, 52, 33, 2\}$
		1.67	0.95	19	2891.8	$\{1, 52, 33, 2, 47, 9\}$
			$+\infty$	19	$+\infty$	$\{1, 52, 33, 2, 47, 9\}$
	0.5	1	0.95	26	4541.95	$\{1, 52\}$
			$+\infty$	26	$+\infty$	$\{1, 52\}$
		1.33	0.95	26	3585.3	$\{1, 52, 33\}$
			$+\infty$	26	$+\infty$	$\{1, 52, 33\}$
		1.67	0.95	26	3276.55	$\{1, 52, 33, 2\}$
			$+\infty$	26	$+\infty$	$\{1, 52, 33, 2\}$
pr76	0.2	1	0.95	16	24905.2	$\{1, 73, 70, 74, 34\}$
			$+\infty$	16	$+\infty$	$\{1, 73, 70, 74, 34\}$
		1.33	0.95	16	25915.05	$\{1, 73, 70, 74, 34, 45, 66\}$
			$+\infty$	16	$+\infty$	$\{1, 73, 70, 74, 34, 45, 66\}$
		1.67	0.95	16	26552.5	$\{1, 73, 70, 74, 34, 45, 66, 9, 62\}$

Instance				**Characterization**		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
pr76	0.2	1.67	$+\infty$	16	$+\infty$	$\{1, 73, 70, 74, 34, 45, 66, 9, 62\}$
	0.35	1	0.95	27	49431.35	$\{1, 73, 70\}$
			$+\infty$	27	$+\infty$	$\{1, 73, 70\}$
		1.33	0.95	27	48803.4	$\{1, 73, 70, 74\}$
			$+\infty$	27	$+\infty$	$\{1, 73, 70, 74\}$
		1.67	0.95	27	33440.95	$\{1, 73, 70, 74, 34, 45\}$
			$+\infty$	27	$+\infty$	$\{1, 73, 70, 74, 34, 45\}$
	0.5	1	0.95	38	74484.75	$\{1, 73\}$
			$+\infty$	38	$+\infty$	$\{1, 73\}$
		1.33	0.95	38	66127.6	$\{1, 73, 70\}$
			$+\infty$	38	$+\infty$	$\{1, 73, 70\}$
		1.67	0.95	38	43238.3	$\{1, 73, 70, 74\}$
			$+\infty$	38	$+\infty$	$\{1, 73, 70, 74\}$
kroA100	0.2	1	0.95	20	6168.35	$\{1, 41, 95, 26, 99\}$
			$+\infty$	20	$+\infty$	$\{1, 41, 95, 26, 99\}$
		1.33	0.95	20	4702.5	$\{1, 41, 95, 26, 99, 35, 69\}$
			$+\infty$	20	$+\infty$	$\{1, 41, 95, 26, 99, 35, 69\}$
		1.67	0.95	20	4382.35	$\{1, 41, 95, 26, 99, 35, 69, 17, 31\}$
			$+\infty$	20	$+\infty$	$\{1, 41, 95, 26, 99, 35, 69, 17, 31\}$
	0.35	1	0.95	35	9107.65	$\{1, 41, 95\}$
			$+\infty$	35	$+\infty$	$\{1, 41, 95\}$
		1.33	0.95	35	8579.45	$\{1, 41, 95, 26\}$
			$+\infty$	35	$+\infty$	$\{1, 41, 95, 26\}$
		1.67	0.95	35	7011.95	$\{1, 41, 95, 26, 99, 35\}$
			$+\infty$	35	$+\infty$	$\{1, 41, 95, 26, 99, 35\}$
	0.5	1	0.95	50	11651.75	$\{1, 41\}$
			$+\infty$	50	$+\infty$	$\{1, 41\}$
		1.33	0.95	50	11759.1	$\{1, 41, 95\}$
			$+\infty$	50	$+\infty$	$\{1, 41, 95\}$
		1.67	0.95	50	11695.45	$\{1, 41, 95, 26\}$
			$+\infty$	50	$+\infty$	$\{1, 41, 95, 26\}$
pr124	0.2	1	0.95	25	19168.15	$\{1, 105, 124, 13, 66\}$
			$+\infty$	25	$+\infty$	$\{1, 105, 124, 13, 66\}$
		1.33	0.95	25	15241.8	$\{1, 105, 124, 13, 66, 85, 41\}$
			$+\infty$	25	$+\infty$	$\{1, 105, 124, 13, 66, 85, 41\}$
		1.67	0.95	25	13168.9	$\{1, 105, 124, 13, 66, 85, 41, 57, 34\}$
			$+\infty$	25	$+\infty$	$\{1, 105, 124, 13, 66, 85, 41, 57, 34\}$
	0.35	1	0.95	44	28759.35	$\{1, 105, 124\}$
			$+\infty$	44	$+\infty$	$\{1, 105, 124\}$
		1.33	0.95	44	19089.3	$\{1, 105, 124, 13\}$
			$+\infty$	44	$+\infty$	$\{1, 105, 124, 13\}$
		1.67	0.95	44	18031.95	$\{1, 105, 124, 13, 66, 85\}$

Instance				**Characterization**		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
pr124	0.35	1.67	$+\infty$	44	$+\infty$	$\{1, 105, 124, 13, 66, 85\}$
	0.5	1	0.95	62	36572.15	$\{1, 105\}$
			$+\infty$	62	$+\infty$	$\{1, 105\}$
		1.33	0.95	62	29530.75	$\{1, 105, 124\}$
			$+\infty$	62	$+\infty$	$\{1, 105, 124\}$
		1.67	0.95	62	20465.85	$\{1, 105, 124, 13\}$
			$+\infty$	62	$+\infty$	$\{1, 105, 124, 13\}$
pr152	0.2	1	0.95	31	27932.85	$\{1, 140, 9, 124, 68\}$
			$+\infty$	31	$+\infty$	$\{1, 140, 9, 124, 68\}$
		1.33	0.95	31	17092.4	$\{1, 140, 9, 124, 68, 48, 108\}$
			$+\infty$	31	$+\infty$	$\{1, 140, 9, 124, 68, 48, 108\}$
		1.67	0.95	31	17283.35	$\{1, 140, 9, 124, 68, 48, 108, 13, 121\}$
			$+\infty$	31	$+\infty$	$\{1, 140, 9, 124, 68, 48, 108, 13, 121\}$
	0.35	1	0.95	54	38967.1	$\{1, 140, 9\}$
			$+\infty$	54	$+\infty$	$\{1, 140, 9\}$
		1.33	0.95	54	30402.85	$\{1, 140, 9, 124\}$
			$+\infty$	54	$+\infty$	$\{1, 140, 9, 124\}$
		1.67	0.95	54	18048.1	$\{1, 140, 9, 124, 68, 48\}$
			$+\infty$	54	$+\infty$	$\{1, 140, 9, 124, 68, 48\}$
	0.5	1	0.95	76	38455.05	$\{1, 140\}$
			$+\infty$	76	$+\infty$	$\{1, 140\}$
		1.33	0.95	76	38102.6	$\{1, 140, 9\}$
			$+\infty$	76	$+\infty$	$\{1, 140, 9\}$
		1.67	0.95	76	24822.55	$\{1, 140, 9, 124\}$
			$+\infty$	76	$+\infty$	$\{1, 140, 9, 124\}$
rat195	0.2	1	0.95	39	583.3	$\{1, 195, 106, 52, 183\}$
			$+\infty$	39	$+\infty$	$\{1, 195, 106, 52, 183\}$
		1.33	0.95	39	449.35	$\{1, 195, 106, 52, 183, 117, 57\}$
			$+\infty$	39	$+\infty$	$\{1, 195, 106, 52, 183, 117, 57\}$
		1.67	0.95	39	683.05	$\{1, 195, 106, 52, 183, 117, 57, 151, 9\}$
			$+\infty$	39	$+\infty$	$\{1, 195, 106, 52, 183, 117, 57, 151, 9\}$
	0.35	1	0.95	69	954.75	$\{1, 195, 106\}$
			$+\infty$	69	$+\infty$	$\{1, 195, 106\}$
		1.33	0.95	69	909.15	$\{1, 195, 106, 52\}$
			$+\infty$	69	$+\infty$	$\{1, 195, 106, 52\}$
		1.67	0.95	69	534.85	$\{1, 195, 106, 52, 183, 117\}$
			$+\infty$	69	$+\infty$	$\{1, 195, 106, 52, 183, 117\}$
	0.5	1	0.95	98	1218.85	$\{1, 195\}$
			$+\infty$	98	$+\infty$	$\{1, 195\}$
		1.33	0.95	98	1105.8	$\{1, 195, 106\}$
			$+\infty$	98	$+\infty$	$\{1, 195, 106\}$
		1.67	0.95	98	910.1	$\{1, 195, 106, 52\}$

Instance				**Characterization**		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
rat195	0.5	1.67	$+\infty$	98	$+\infty$	$\{1, 195, 106, 52\}$
pr226	0.2	1	0.95	46	34803.25	$\{1, 155, 217, 38, 64\}$
			$+\infty$	46	$+\infty$	$\{1, 155, 217, 38, 64\}$
		1.33	0.95	46	20424.05	$\{1, 155, 217, 38, 64, 92, 136\}$
			$+\infty$	46	$+\infty$	$\{1, 155, 217, 38, 64, 92, 136\}$
		1.67	0.95	46	20424.05	$\{1, 155, 217, 38, 64, 92, 136, 190, 111\}$
			$+\infty$	46	$+\infty$	$\{1, 155, 217, 38, 64, 92, 136, 190, 111\}$
	0.35	1	0.95	80	41523.55	$\{1, 155, 217\}$
			$+\infty$	80	$+\infty$	$\{1, 155, 217\}$
		1.33	0.95	80	31902.9	$\{1, 155, 217, 38\}$
			$+\infty$	80	$+\infty$	$\{1, 155, 217, 38\}$
		1.67	0.95	80	19851.2	$\{1, 155, 217, 38, 64, 92\}$
			$+\infty$	80	$+\infty$	$\{1, 155, 217, 38, 64, 92\}$
	0.5	1	0.95	113	56832.8	$\{1, 155\}$
			$+\infty$	113	$+\infty$	$\{1, 155\}$
		1.33	0.95	113	50361.4	$\{1, 155, 217\}$
			$+\infty$	113	$+\infty$	$\{1, 155, 217\}$
		1.67	0.95	113	31902.9	$\{1, 155, 217, 38\}$
			$+\infty$	113	$+\infty$	$\{1, 155, 217, 38\}$

Asymmetric instances

Instance				**Characterization**		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
ft53	0.2	1	0.95	11	2865.2	$\{1, 50, 33, 35, 29\}$
			$+\infty$	11	$+\infty$	$\{1, 50, 33, 35, 29\}$
		1.33	0.95	11	2762.6	$\{1, 50, 33, 35, 29, 25, 6\}$
			$+\infty$	11	$+\infty$	$\{1, 50, 33, 35, 29, 25, 6\}$
		1.67	0.95	11	2335.1	$\{1, 50, 33, 35, 29, 25, 6, 42, 31\}$
			$+\infty$	11	$+\infty$	$\{1, 50, 33, 35, 29, 25, 6, 42, 31\}$
	0.35	1	0.95	19	3886.45	$\{1, 50, 33\}$
			$+\infty$	19	$+\infty$	$\{1, 50, 33\}$
		1.33	0.95	19	3886.45	$\{1, 50, 33, 35\}$
			$+\infty$	19	$+\infty$	$\{1, 50, 33, 35\}$
		1.67	0.95	19	3152.1	$\{1, 50, 33, 35, 29, 25\}$
			$+\infty$	19	$+\infty$	$\{1, 50, 33, 35, 29, 25\}$
	0.5	1	0.95	27	5501.45	$\{1, 50\}$
			$+\infty$	27	$+\infty$	$\{1, 50\}$
		1.33	0.95	27	4537.2	$\{1, 50, 33\}$
			$+\infty$	27	$+\infty$	$\{1, 50, 33\}$
		1.67	0.95	27	4537.2	$\{1, 50, 33, 35\}$
			$+\infty$	27	$+\infty$	$\{1, 50, 33, 35\}$
ftv64	0.2	1	0.95	13	710.6	$\{1, 24, 63, 13, 7\}$
			$+\infty$	13	$+\infty$	$\{1, 24, 63, 13, 7\}$

Instance				Characterization		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
ftv64	0.2	1.33	0.95	13	634.6	$\{1, 24, 63, 13, 7, 15, 34\}$
			$+\infty$	13	$+\infty$	$\{1, 24, 63, 13, 7, 15, 34\}$
		1.67	0.95	13	416.1	$\{1, 24, 63, 13, 7, 15, 34, 59, 28\}$
			$+\infty$	13	$+\infty$	$\{1, 24, 63, 13, 7, 15, 34, 59, 28\}$
	0.35	1	0.95	23	1046.9	$\{1, 24, 63\}$
			$+\infty$	23	$+\infty$	$\{1, 24, 63\}$
		1.33	0.95	23	809.4	$\{1, 24, 63, 13\}$
			$+\infty$	23	$+\infty$	$\{1, 24, 63, 13\}$
		1.67	0.95	23	875.9	$\{1, 24, 63, 13, 7, 15\}$
			$+\infty$	23	$+\infty$	$\{1, 24, 63, 13, 7, 15\}$
	0.5	1	0.95	33	1370.85	$\{1, 24\}$
			$+\infty$	33	$+\infty$	$\{1, 24\}$
		1.33	0.95	33	1238.8	$\{1, 24, 63\}$
			$+\infty$	33	$+\infty$	$\{1, 24, 63\}$
		1.67	0.95	33	948.1	$\{1, 24, 63, 13\}$
			$+\infty$	33	$+\infty$	$\{1, 24, 63, 13\}$
ft70	0.2	1	0.95	14	9689.05	$\{1, 67, 47, 36, 41\}$
			$+\infty$	14	$+\infty$	$\{1, 67, 47, 36, 41\}$
		1.33	0.95	14	8033.2	$\{1, 67, 47, 36, 41, 50, 20\}$
			$+\infty$	14	$+\infty$	$\{1, 67, 47, 36, 41, 50, 20\}$
		1.67	0.95	14	8027.5	$\{1, 67, 47, 36, 41, 50, 20, 46, 45\}$
			$+\infty$	14	$+\infty$	$\{1, 67, 47, 36, 41, 50, 20, 46, 45\}$
	0.35	1	0.95	25	17130.4	$\{1, 67, 47\}$
			$+\infty$	25	$+\infty$	$\{1, 67, 47\}$
		1.33	0.95	25	16268.75	$\{1, 67, 47, 36\}$
			$+\infty$	25	$+\infty$	$\{1, 67, 47, 36\}$
		1.67	0.95	25	13874.75	$\{1, 67, 47, 36, 41, 50\}$
			$+\infty$	25	$+\infty$	$\{1, 67, 47, 36, 41, 50\}$
	0.5	1	0.95	35	21829.1	$\{1, 67\}$
			$+\infty$	35	$+\infty$	$\{1, 67\}$
		1.33	0.95	35	18450.9	$\{1, 67, 47\}$
			$+\infty$	35	$+\infty$	$\{1, 67, 47\}$
		1.67	0.95	35	18450.9	$\{1, 67, 47, 36\}$
			$+\infty$	35	$+\infty$	$\{1, 67, 47, 36\}$
kro124p	0.2	1	0.95	20	11474.1	$\{1, 71, 95, 26, 99\}$
			$+\infty$	20	$+\infty$	$\{1, 71, 95, 26, 99\}$
		1.33	0.95	20	8776.1	$\{1, 71, 95, 26, 99, 35, 40\}$
			$+\infty$	20	$+\infty$	$\{1, 71, 95, 26, 99, 35, 40\}$
		1.67	0.95	20	10059.55	$\{1, 71, 95, 26, 99, 35, 40, 17, 31\}$
			$+\infty$	20	$+\infty$	$\{1, 71, 95, 26, 99, 35, 40, 17, 31\}$
	0.35	1	0.95	35	15836.5	$\{1, 71, 95\}$
			$+\infty$	35	$+\infty$	$\{1, 71, 95\}$

Instance				Characterization		
Matrix name	α	β	γ	Q	L	**Set of black nodes**
kro124p	0.35	1.33	0.95	35	14595.8	$\{1, 71, 95, 26\}$
			$+\infty$	35	$+\infty$	$\{1, 71, 95, 26\}$
		1.67	0.95	35	15536.3	$\{1, 71, 95, 26, 99, 35\}$
			$+\infty$	35	$+\infty$	$\{1, 71, 95, 26, 99, 35\}$
	0.5	1	0.95	50	26600.95	$\{1, 71\}$
			$+\infty$	50	$+\infty$	$\{1, 71\}$
		1.33	0.95	50	24275.35	$\{1, 71, 95\}$
			$+\infty$	50	$+\infty$	$\{1, 71, 95\}$
		1.67	0.95	50	17048.7	$\{1, 71, 95, 26\}$
			$+\infty$	50	$+\infty$	$\{1, 71, 95, 26\}$
ftv170	0.2	1	0.95	35	974.7	$\{1, 67, 163, 11, 146\}$
			$+\infty$	35	$+\infty$	$\{1, 67, 163, 11, 146\}$
		1.33	0.95	35	930.05	$\{1, 67, 163, 11, 146, 160, 49\}$
			$+\infty$	35	$+\infty$	$\{1, 67, 163, 11, 146, 160, 49\}$
		1.67	0.95	35	863.55	$\{1, 67, 163, 11, 146, 160, 49, 162, 111\}$
			$+\infty$	35	$+\infty$	$\{1, 67, 163, 11, 146, 160, 49, 162, 111\}$
	0.35	1	0.95	60	1507.65	$\{1, 67, 163\}$
			$+\infty$	60	$+\infty$	$\{1, 67, 163\}$
		1.33	0.95	60	1442.1	$\{1, 67, 163, 11\}$
			$+\infty$	60	$+\infty$	$\{1, 67, 163, 11\}$
		1.67	0.95	60	1213.15	$\{1, 67, 163, 11, 146, 160\}$
			$+\infty$	60	$+\infty$	$\{1, 67, 163, 11, 146, 160\}$
	0.5	1	0.95	86	2107.1	$\{1, 67\}$
			$+\infty$	86	$+\infty$	$\{1, 67\}$
		1.33	0.95	86	1931.35	$\{1, 67, 163\}$
			$+\infty$	86	$+\infty$	$\{1, 67, 163\}$
		1.67	0.95	86	2203.05	$\{1, 67, 163, 11\}$
			$+\infty$	86	$+\infty$	$\{1, 67, 163, 11\}$

Appendix B

Detailed results of the tests on the parameters of the ILS algorithm

B.1 Results for $\omega = 1/3$

B.1.1 Results for a maximum of 1000 iterations

ANN as the constructive heuristic

Table B.1: Computational results, for $\omega = 1/3$ and $MaxIt = 1000$, using the solution obtained through the ANN heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.68% *	5.45% *	2.75
		1.33	0.95	4.4% *	4.68% *	5.04% *	2.15
			∞	5.38% *	6.01% *	7.81% *	2.19
		1.67	0.95	—	—	—	—
			∞	0.69%	1.98%	3.26%	2.27
	0.35	1	0.95	0.0%	0.72%	1.17%	2.44
			∞	0.23%	0.6%	1.17%	1.88
		1.33	0.95	0.0%	1.44%	4.28%	2.58
			∞	0.0%	1.63%	3.55%	1.74
		1.67	0.95	0.0%	0.98%	1.95%	1.23
			∞	0.0%	0.78%	1.95%	1.35
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.97%	4.87%	2.2
		1.33	0.95	—	—	—	—
			∞	1.11%	2.19%	3.7%	1.66
		1.67	0.95	0.51%	2.49%	3.87%	1.56
			∞	0.14%	2.07%	3.12%	1.65

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr76	0.2	1	0.95	———	———	———	———
pr76	0.2	1	∞	1.45% *	1.63% *	1.88% *	4.45
pr76	0.2	1.33	0.95	1.54% *	2.19% *	3.25% *	6.19
pr76	0.2	1.33	∞	1.13% *	2.36% *	5.72% *	3.79
pr76	0.2	1.67	0.95	0.58% *	1.55% *	2.64% *	3.76
pr76	0.2	1.67	∞	0.46% *	0.71% *	0.8% *	4.01
pr76	0.35	1	0.95	———	———	———	———
pr76	0.35	1	∞	4.13% *	4.54% *	5.42% *	7.47
pr76	0.35	1.33	0.95	0.01%	0.1%	0.26%	3.24
pr76	0.35	1.33	∞	0.01%	0.06%	0.14%	3.76
pr76	0.35	1.67	0.95	0.86%	1.31%	1.86%	5.49
pr76	0.35	1.67	∞	0.26%	0.82%	1.08%	3.38
pr76	0.5	1	0.95	———	———	———	———
pr76	0.5	1	∞	0.01%	1.54%	4.81%	6.67
pr76	0.5	1.33	0.95	0.0%	0.05%	0.09%	4.18
pr76	0.5	1.33	∞	0.0%	0.4%	1.38%	7.35
pr76	0.5	1.67	0.95	0.55%	0.57%	0.63%	3.34
pr76	0.5	1.67	∞	0.82%	0.98%	1.1%	3.16
kroA100	0.2	1	0.95	———	———	———	———
kroA100	0.2	1	∞	8.25% *	9.23% *	10.09% *	34.05
kroA100	0.2	1.33	0.95	———	———	———	———
kroA100	0.2	1.33	∞	0.0%	0.29%	0.95%	6.63
kroA100	0.2	1.67	0.95	———	———	———	———
kroA100	0.2	1.67	∞	0.0%	0.06%	0.09%	7.12
kroA100	0.35	1	0.95	———	———	———	———
kroA100	0.35	1	∞	6.33% *	6.6% *	6.88% *	24.3
kroA100	0.35	1.33	0.95	———	———	———	———
kroA100	0.35	1.33	∞	1.56% *	2.66% *	4.93% *	19.86
kroA100	0.35	1.67	0.95	0.0%	0.01%	0.04%	6.23
kroA100	0.35	1.67	∞	0.0%	0.29%	0.55%	8.07
kroA100	0.5	1	0.95	———	———	———	———
kroA100	0.5	1	∞	2.62% *	3.92% *	4.72% *	21.54
kroA100	0.5	1.33	0.95	1.79% *	2.38% *	3.39% *	20.43
kroA100	0.5	1.33	∞	1.69% *	2.07% *	3.27% *	16.36
kroA100	0.5	1.67	0.95	0.0%	0.13%	0.61%	8.48
kroA100	0.5	1.67	∞	0.0%	0.2%	0.47%	6.85
pr124	0.2	1	0.95	———	———	———	———
pr124	0.2	1	∞	5.04% *	6.87% *	7.78% *	17.16
pr124	0.2	1.33	0.95	9.44% *	10.1% *	11.22% *	7.27
pr124	0.2	1.33	∞	9.3% *	9.73% *	9.83% *	6.15
pr124	0.2	1.67	0.95	10.93% *	10.93% *	10.93% *	7.26

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr124	0.2	1.67	∞	11.0% *	11.06% *	11.1% *	8.66
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	11.62% *	14.72% *	7.17
		1.33	0.95	—	—	—	—
			∞	0.78%	0.86%	0.88%	7.8
		1.67	0.95	—	—	—	—
			∞	6.56% *	6.78% *	6.9% *	7.08
	0.5	1	0.95	—	—	—	—
			∞	6.18% *	6.32% *	6.87% *	15.38
		1.33	0.95	—	—	—	—
			∞	5.24% *	5.24% *	5.24% *	7.73
		1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	6.14
pr152	0.2	1	0.95	13.48% *	14.4% *	15.24% *	32.85
			∞	13.98% *	14.99% *	15.8% *	31.73
		1.33	0.95	13.2% *	14.28% *	15.41% *	26.47
			∞	13.21% *	13.62% *	13.99% *	37.42
		1.67	0.95	13.07% *	13.38% *	14.53% *	25.69
			∞	13.4% *	14.33% *	15.12% *	20.01
	0.35	1	0.95	23.38% *	24.75% *	27.38% *	34.45
			∞	24.31% *	25.07% *	26.81% *	35.6
		1.33	0.95	19.7% *	19.85% *	20.46% *	18.08
			∞	20.94% *	21.09% *	21.19% *	18.64
		1.67	0.95	13.98% *	14.51% *	14.64% *	20.18
			∞	14.1% *	15.24% *	16.76% *	19.79
	0.5	1	0.95	—	—	—	—
			∞	9.56% *	9.6% *	9.78% *	24.99
		1.33	0.95	9.55% *	10.89% *	11.4% *	24.26
			∞	9.43% *	10.73% *	11.17% *	22.08
		1.67	0.95	—	—	—	—
			∞	23.61% *	23.61% *	23.61% *	15.56
rat195	0.2	1	0.95	—	—	—	—
			∞	5.47% *	7.7% *	8.79% *	76.99
		1.33	0.95	—	—	—	—
			∞	2.82% *	4.1% *	6.04% *	70.25
		1.67	0.95	3.45% *	3.88% *	4.39% *	43.52
			∞	3.0% *	3.34% *	3.63% *	41.14
	0.35	1	0.95	—	—	—	—
			∞	2.08% *	4.12% *	6.07% *	71.36
		1.33	0.95	2.23% *	2.74% *	3.16% *	36.73
			∞	1.78% *	2.69% *	3.25% *	39.89

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
rat195	0.35	1.67	0.95	—	—	—	—
			∞	2.51% *	3.1% *	3.86% *	35.3
	0.5	1	0.95	—	—	—	—
			∞	2.16% *	2.72% *	3.56% *	65.6
		1.33	0.95	1.1% *	2.11% *	2.82% *	52.26
			∞	2.4% *	2.74% *	3.33% *	41.22
		1.67	0.95	2.63% *	2.82% *	3.07% *	39.46
			∞	2.05%	2.21%	2.36%	41.63
pr226	0.2	1	0.95	22.79% *	26.82% *	33.29% *	133.62
			∞	22.12% *	23.76% *	28.09% *	139.03
		1.33	0.95	—	—	—	—
			∞	10.54% *	11.7% *	12.84% *	49.39
		1.67	0.95	6.82% *	7.55% *	7.98% *	67.81
			∞	6.83% *	7.31% *	7.79% *	66.7
	0.35	1	0.95	17.99% *	20.84% *	24.78% *	143.51
			∞	17.67% *	18.27% *	18.63% *	60.79
		1.33	0.95	—	—	—	—
			∞	10.47% *	10.94% *	11.26% *	65.68
		1.67	0.95	—	—	—	—
			∞	8.77% *	9.21% *	9.81% *	44.17
	0.5	1	0.95	14.87% *	21.34% *	32.29% *	231.01
			∞	14.79% *	17.64% *	20.26% *	107.87
		1.33	0.95	10.95% *	12.35% *	14.27% *	66.62
			∞	10.6% *	12.53% *	18.18% *	97.75
		1.67	0.95	—	—	—	—
			∞	10.84% *	11.16% *	11.35% *	57.56

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	13.71% *	20.05% *	23.76% *	7.16
			∞	17.17% *	20.04% *	23.92% *	8.67
		1.33	0.95	4.46%	6.33%	7.78%	2.91
			∞	5.69%	8.2%	10.19%	2.36
		1.67	0.95	2.55%	4.97%	7.96%	1.9
			∞	2.75%	5.66%	10.58%	2.47
	0.35	1	0.95	5.81%	8.01%	11.11%	6.37
			∞	3.39%	7.16%	9.81%	5.44
		1.33	0.95	3.67%	6.96%	11.16%	4.44
			∞	5.26%	7.47%	10.25%	5.26
		1.67	0.95	2.39%	3.96%	4.91%	2.32
			∞	4.2%	7.56%	9.11%	1.41

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.5	1	0.95	6.68%	8.7%	11.34%	5.16
			∞	1.23%	2.82%	3.78%	2.32
		1.33	0.95	0.0%	4.5%	12.43%	2.12
			∞	3.39%	7.19%	10.89%	1.69
		1.67	0.95	3.26%	5.77%	7.83%	1.81
			∞	2.74%	7.12%	13.11%	2.4
ftv64	0.2	1	0.95	—	—	—	—
			∞	3.88%	7.47%	12.06%	6.36
		1.33	0.95	—	—	—	—
			∞	3.29%	8.27%	11.59%	3.02
		1.67	0.95	3.7%	6.46%	10.44%	3.42
			∞	3.64%	7.21%	11.69%	2.89
	0.35	1	0.95	4.6%	8.04%	14.36%	5.03
			∞	0.98%	4.1%	8.34%	4.38
		1.33	0.95	2.87%	6.05%	11.27%	3.16
			∞	3.3%	5.18%	8.67%	2.46
		1.67	0.95	1.79%	3.81%	6.47%	2.24
			∞	4.89%	8.48%	11.47%	2.24
	0.5	1	0.95	5.03%	7.88%	12.54%	7.4
			∞	2.49%	7.44%	10.76%	4.43
		1.33	0.95	2.06%	5.47%	7.17%	1.85
			∞	0.98%	6.04%	10.64%	2.39
		1.67	0.95	1.79%	3.77%	6.68%	2.79
			∞	1.85%	3.59%	4.72%	2.22
ft70	0.2	1	0.95	3.84% *	5.22% *	6.51% *	9.81
			∞	4.14% *	5.9% *	7.49% *	10.56
		1.33	0.95	2.89%	4.12%	4.55%	9.42
			∞	1.97%	3.56%	4.44%	6.34
		1.67	0.95	2.67%	3.92%	5.34%	9.44
			∞	3.75%	3.99%	4.21%	6.29
	0.35	1	0.95	2.56%	3.72%	4.33%	7.99
			∞	2.68%	3.8%	5.41%	7.53
		1.33	0.95	2.08%	3.22%	4.08%	9.6
			∞	2.91%	3.89%	4.91%	6.83
		1.67	0.95	—	—	—	—
			∞	3.34%	4.02%	5.16%	6.82
	0.5	1	0.95	—	—	—	—
			∞	1.81%	3.24%	4.3%	7.09
		1.33	0.95	—	—	—	—
			∞	1.61%	3.11%	3.78%	6.68

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft70	0.5	1.67	0.95	—	—	—	—
			∞	2.47%	3.9%	5.33%	6.47
kro124p	0.2	1	0.95	—	—	—	—
			∞	7.37% *	11.15% *	14.41% *	25.18
		1.33	0.95	1.38%	6.25%	9.43%	17.32
			∞	1.47%	6.15%	9.63%	14.58
		1.67	0.95	4.08%	7.92%	12.81%	17.91
			∞	7.64%	9.15%	11.27%	12.97
	0.35	1	0.95	—	—	—	—
			∞	9.45% *	10.66% *	12.69% *	17.75
		1.33	0.95	4.42%	6.63%	8.32%	18.89
			∞	4.33%	5.83%	6.91%	14.62
		1.67	0.95	9.0%	10.91%	13.83%	14.35
			∞	3.99%	7.9%	12.7%	13.02
	0.5	1	0.95	—	—	—	—
			∞	2.89% *	9.34% *	17.74% *	26.72
		1.33	0.95	5.45%	7.53%	9.42%	17.66
			∞	8.66%	9.45%	11.51%	18.9
		1.67	0.95	5.21%	6.63%	9.82%	15.38
			∞	9.07%	10.33%	11.67%	16.66
ftv170	0.2	1	0.95	—	—	—	—
			∞	6.13% *	16.81% *	28.26% *	55.17
		1.33	0.95	7.61%	10.42%	13.67%	39.68
			∞	11.2%	12.23%	13.27%	24.72
		1.67	0.95	5.91%	9.2%	13.2%	30.07
			∞	8.23%	12.6%	17.66%	19.95
	0.35	1	0.95	—	—	—	—
			∞	7.66% *	10.03% *	13.54% *	48.79
		1.33	0.95	—	—	—	—
			∞	4.75%	9.07%	16.48%	33.83
		1.67	0.95	7.22%	8.86%	10.85%	26.39
			∞	5.41%	9.34%	11.62%	24.4
	0.5	1	0.95	—	—	—	—
			∞	9.14% *	10.56% *	13.07% *	22.91
		1.33	0.95	—	—	—	—
			∞	4.77%	10.74%	14.61%	32.86
		1.67	0.95	7.51%	9.57%	11.87%	20.04
			∞	6.13%	11.18%	16.52%	21.06

AFI as the constructive heuristic

Table B.2: Computational results, for $\omega = 1/3$ and $MaxIt = 1000$, using the solution obtained through the AFI heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	2.11
		1.33	0.95	4.27% *	4.65% *	5.36% *	2.69
			∞	5.38% *	5.48% *	5.74% *	2.69
		1.67	0.95	1.99%	2.28%	3.41%	1.87
			∞	1.99%	2.03%	2.16%	1.72
	0.35	1	0.95	0.0%	0.36%	1.17%	1.85
			∞	0.23%	0.52%	1.17%	1.52
		1.33	0.95	0.0%	0.25%	1.24%	2.58
			∞	0.0%	1.81%	4.31%	1.75
		1.67	0.95	1.92%	2.73%	4.38%	1.87
			∞	0.0%	1.66%	4.38%	2.02
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.59%	2.95%	2.32
		1.33	0.95	1.07%	1.25%	1.44%	2.61
			∞	0.29%	0.95%	3.61%	1.48
		1.67	0.95	0.0%	2.42%	3.87%	1.99
			∞	0.56%	1.42%	2.53%	1.58
pr76	0.2	1	0.95	—	—	—	—
			∞	1.78% *	2.02% *	2.18% *	6.19
		1.33	0.95	1.1% *	2.75% *	3.86% *	4.96
			∞	1.87% *	3.21% *	3.84% *	3.86
		1.67	0.95	0.39% *	0.61% *	1.16% *	4.4
			∞	0.46% *	0.7% *	1.44% *	4.13
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.36% *	5.27% *	7.89
		1.33	0.95	0.0%	0.02%	0.08%	4.21
			∞	0.08%	0.67%	1.11%	3.62
		1.67	0.95	0.07%	0.89%	2.82%	3.86
			∞	0.0%	1.26%	2.97%	4.97
	0.5	1	0.95	—	—	—	—
			∞	2.15%	3.32%	4.07%	4.66
		1.33	0.95	2.13%	2.62%	2.74%	4.16
			∞	2.74%	2.74%	2.74%	4.55

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr76	0.5	1.67	0.95	———	———	———	———
			∞	0.07%	1.36%	2.82%	3.61
kroA100	0.2	1	0.95	———	———	———	———
			∞	8.25% *	9.83% *	11.88% *	40.75
		1.33	0.95	0.0%	0.0%	0.0%	8.76
			∞	0.0%	0.22%	0.46%	8.2
		1.67	0.95	0.0%	0.19%	0.46%	8.24
			∞	0.0%	0.27%	0.46%	7.95
	0.35	1	0.95	———	———	———	———
			∞	6.63% *	7.01% *	7.73% *	29.98
		1.33	0.95	2.44% *	3.65% *	5.3% *	18.68
			∞	1.74% *	3.3% *	4.4% *	11.94
		1.67	0.95	0.0%	0.83%	3.33%	8.83
			∞	0.0%	0.02%	0.09%	11.0
	0.5	1	0.95	———	———	———	———
			∞	3.32% *	4.53% *	5.13% *	25.1
		1.33	0.95	1.76% *	2.45% *	4.72% *	19.54
			∞	1.78% *	2.8% *	4.8% *	19.19
		1.67	0.95	0.0%	0.09%	0.46%	8.28
			∞	0.0%	0.09%	0.45%	7.81
pr124	0.2	1	0.95	6.84% *	8.09% *	9.74% *	19.58
			∞	5.91% *	6.72% *	8.7% *	16.06
		1.33	0.95	9.44% *	10.07% *	11.0% *	9.08
			∞	9.3% *	10.21% *	10.86% *	9.42
		1.67	0.95	10.86% *	10.96% *	11.14% *	11.95
			∞	11.0% *	11.35% *	12.57% *	8.43
	0.35	1	0.95	———	———	———	———
			∞	8.4% *	8.4% *	8.4% *	8.15
		1.33	0.95	———	———	———	———
			∞	0.0%	0.37%	0.98%	10.35
		1.67	0.95	7.25% *	7.81% *	9.07% *	14.9
			∞	5.92% *	6.69% *	7.44% *	12.44
	0.5	1	0.95	———	———	———	———
			∞	5.27% *	5.27% *	5.27% *	12.29
		1.33	0.95	1.98% *	2.51% *	2.87% *	13.03
			∞	4.95% *	5.13% *	5.24% *	12.67
		1.67	0.95	0.6%	0.71%	0.78%	9.9
			∞	0.88%	0.88%	0.88%	9.6
pr152	0.2	1	0.95	———	———	———	———
			∞	13.95% *	15.49% *	18.84% *	42.26
		1.33	0.95	15.2% *	15.73% *	15.92% *	26.43

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr152	0.2	1.33	∞	14.23% *	14.77% *	15.26% *	30.67
		1.67	0.95	12.48% *	12.97% *	13.1% *	14.56
			∞	12.98% *	13.24% *	14.01% *	22.78
	0.35	1	0.95	—	—	—	—
			∞	24.35% *	25.04% *	26.22% *	23.35
		1.33	0.95	19.12% *	19.32% *	20.08% *	19.06
			∞	19.94% *	20.45% *	20.6% *	26.12
		1.67	0.95	—	—	—	—
			∞	14.73% *	15.08% *	15.74% *	16.34
	0.5	1	0.95	—	—	—	—
			∞	9.33% *	9.86% *	10.21% *	20.65
		1.33	0.95	8.73% *	9.23% *	9.42% *	21.72
			∞	9.32% *	9.35% *	9.41% *	18.88
		1.67	0.95	—	—	—	—
			∞	23.01% *	23.6% *	24.0% *	17.95
rat195	0.2	1	0.95	—	—	—	—
			∞	6.64% *	8.13% *	9.37% *	98.61
		1.33	0.95	—	—	—	—
			∞	4.34% *	5.22% *	6.04% *	67.52
		1.67	0.95	3.94% *	4.83% *	6.14% *	54.31
			∞	3.99% *	5.08% *	6.0% *	54.82
	0.35	1	0.95	—	—	—	—
			∞	3.63% *	5.07% *	7.18% *	71.12
		1.33	0.95	4.58% *	5.06% *	5.34% *	75.1
			∞	3.61% *	5.41% *	7.94% *	58.83
		1.67	0.95	—	—	—	—
			∞	3.27% *	4.44% *	6.86% *	60.82
	0.5	1	0.95	—	—	—	—
			∞	3.87% *	4.58% *	5.28% *	59.34
		1.33	0.95	1.32% *	2.17% *	2.95% *	118.79
			∞	2.53% *	4.01% *	5.06% *	57.54
		1.67	0.95	3.16% *	5.23% *	6.46% *	81.51
			∞	3.38%	4.28%	5.16%	79.22
pr226	0.2	1	0.95	—	—	—	—
			∞	22.35% *	29.13% *	38.4% *	160.58
		1.33	0.95	9.68% *	10.62% *	12.19% *	70.07
			∞	9.12% *	10.33% *	12.17% *	117.81
		1.67	0.95	6.8% *	6.95% *	7.25% *	51.77
			∞	6.84% *	7.12% *	7.52% *	59.5
	0.35	1	0.95	—	—	—	—
			∞	14.52% *	14.66% *	14.84% *	84.45

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr226	0.35	1.33	0.95	11.45% *	11.92% *	12.31% *	58.0
			∞	10.47% *	10.68% *	10.9% *	63.27
		1.67	0.95	9.12% *	9.18% *	9.24% *	63.57
			∞	8.77% *	8.88% *	8.98% *	70.75
	0.5	1	0.95	—	—	—	—
			∞	19.22% *	19.85% *	20.53% *	154.95
		1.33	0.95	11.28% *	11.66% *	11.91% *	82.33
			∞	10.52% *	10.83% *	11.46% *	75.18
		1.67	0.95	10.97% *	11.07% *	11.23% *	68.98
			∞	10.65% *	10.87% *	11.13% *	50.05

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	—	—	—	—
			∞	14.14% *	16.53% *	19.38% *	7.36
		1.33	0.95	0.32%	3.33%	5.57%	4.24
			∞	3.28%	6.72%	10.71%	3.39
		1.67	0.95	5.0%	8.87%	11.12%	2.5
			∞	8.71%	9.32%	10.05%	2.52
	0.35	1	0.95	—	—	—	—
			∞	4.11%	7.06%	8.52%	4.79
		1.33	0.95	3.67%	5.59%	7.08%	3.09
			∞	3.77%	5.65%	8.03%	3.7
		1.67	0.95	0.94%	5.32%	12.4%	3.37
			∞	1.38%	9.49%	14.44%	2.23
	0.5	1	0.95	4.2%	6.21%	9.96%	4.83
			∞	4.24%	6.23%	9.75%	2.42
		1.33	0.95	1.71%	3.22%	5.1%	2.94
			∞	1.12%	5.68%	8.5%	2.14
		1.67	0.95	1.96%	3.45%	5.24%	1.39
			∞	1.71%	2.92%	4.08%	1.63
ftv64	0.2	1	0.95	4.83%	9.03%	12.9%	4.94
			∞	3.77%	5.98%	8.6%	6.02
		1.33	0.95	0.97%	4.84%	8.03%	2.4
			∞	3.88%	6.4%	7.71%	3.26
		1.67	0.95	—	—	—	—
			∞	2.28%	5.47%	7.78%	2.52
	0.35	1	0.95	2.82%	6.15%	10.18%	6.65
			∞	2.6%	7.43%	9.91%	3.82
		1.33	0.95	2.82%	4.47%	8.02%	2.45
			∞	1.57%	7.5%	12.08%	3.98

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.35	1.67	0.95	2.28%	3.85%	5.38%	2.73
			∞	3.1%	4.31%	6.36%	2.0
	0.5	1	0.95	—	—	—	—
			∞	1.35%	3.81%	6.16%	3.31
		1.33	0.95	0.98%	4.76%	8.63%	2.76
			∞	0.87%	3.9%	7.33%	2.72
		1.67	0.95	—	—	—	—
			∞	0.87%	2.17%	3.04%	1.86
ft70	0.2	1	0.95	5.02% *	5.66% *	6.38% *	10.19
			∞	4.07% *	5.34% *	6.86% *	11.15
		1.33	0.95	2.89%	4.49%	7.44%	8.49
			∞	1.02%	2.5%	4.29%	8.2
		1.67	0.95	2.5%	3.48%	5.05%	10.83
			∞	2.24%	3.12%	4.79%	7.38
	0.35	1	0.95	1.98%	3.05%	4.48%	8.82
			∞	1.4%	3.01%	5.19%	10.49
		1.33	0.95	2.57%	3.04%	3.66%	9.08
			∞	1.11%	3.27%	5.8%	8.82
		1.67	0.95	2.68%	3.33%	4.26%	7.37
			∞	2.82%	3.5%	4.46%	5.85
	0.5	1	0.95	2.23%	3.33%	4.66%	15.55
			∞	3.11%	4.03%	4.99%	5.73
		1.33	0.95	2.5%	3.53%	5.3%	9.02
			∞	3.03%	3.78%	4.77%	4.8
		1.67	0.95	1.98%	4.63%	6.24%	6.92
			∞	2.72%	4.0%	4.94%	5.68
kro124p	0.2	1	0.95	5.95% *	7.47% *	9.78% *	15.78
			∞	5.58% *	8.51% *	10.96% *	15.85
		1.33	0.95	0.54%	3.09%	4.4%	12.81
			∞	3.65%	4.34%	4.79%	14.38
		1.67	0.95	0.61%	1.38%	2.3%	12.02
			∞	0.72%	1.51%	2.03%	8.84
	0.35	1	0.95	—	—	—	—
			∞	11.3% *	13.68% *	16.76% *	20.74
		1.33	0.95	0.48%	1.65%	2.86%	13.29
			∞	2.14%	2.5%	3.29%	9.71
		1.67	0.95	0.64%	2.0%	3.33%	10.98
			∞	1.9%	3.4%	4.37%	12.24
	0.5	1	0.95	4.67% *	5.5% *	6.07% *	23.46
			∞	5.57% *	7.28% *	8.44% *	19.47
		1.33	0.95	2.22%	3.9%	7.37%	13.78

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.5	1.33	∞	2.63%	3.28%	3.69%	16.72
		1.67	0.95	2.24%	3.14%	3.63%	9.92
			∞	2.22%	3.22%	4.18%	12.72
ftv170	0.2	1	0.95	—	—	—	—
			∞	6.46% *	9.79% *	12.51% *	56.52
		1.33	0.95	5.69%	10.78%	20.16%	25.3
			∞	5.47%	9.7%	19.04%	23.75
		1.67	0.95	5.51%	7.21%	10.8%	26.92
			∞	5.91%	7.19%	9.35%	28.51
	0.35	1	0.95	9.81% *	13.41% *	17.87% *	32.5
			∞	13.87% *	16.27% *	18.3% *	29.3
		1.33	0.95	4.39%	7.33%	12.45%	29.57
			∞	5.26%	8.01%	12.99%	33.26
		1.67	0.95	4.21%	8.64%	11.4%	16.93
			∞	4.79%	7.99%	12.89%	20.57
	0.5	1	0.95	10.23% *	12.96% *	17.92% *	58.18
			∞	6.19% *	11.56% *	18.67% *	45.2
		1.33	0.95	9.91%	12.22%	15.04%	36.64
			∞	9.18%	10.59%	12.87%	38.08
		1.67	0.95	2.0%	5.38%	8.71%	24.06
			∞	2.43%	8.54%	12.85%	18.18

ARI as the constructive heuristic

Table B.3: Computational results, for $\omega = 1/3$ and $MaxIt = 1000$, using the solution obtained through the ARI heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.51% *	4.57% *	4.88
		1.33	0.95	4.4% *	4.67% *	4.96% *	6.4
			∞	5.38% *	5.63% *	6.02% *	3.46
		1.67	0.95	1.01%	2.44%	3.57%	5.25
			∞	0.69%	1.73%	1.99%	5.29
	0.35	1	0.95	0.42%	0.42%	0.42%	3.91
			∞	0.33%	0.99%	1.32%	3.2
		1.33	0.95	0.0%	2.47%	4.34%	3.31
			∞	1.24%	1.87%	2.91%	2.61

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.35	1.67	0.95	0.0%	1.38%	2.93%	2.69
			∞	0.0%	0.39%	1.04%	2.62
	0.5	1	0.95	———	———	———	———
			∞	0.0%	0.22%	1.08%	3.59
		1.33	0.95	0.42%	0.42%	0.42%	7.17
			∞	0.29%	0.76%	2.67%	3.02
		1.67	0.95	0.0%	1.5%	3.41%	4.12
			∞	0.0%	1.13%	3.63%	2.81
pr76	0.2	1	0.95	———	———	———	———
			∞	1.45% *	1.85% *	2.16% *	10.26
		1.33	0.95	1.03% *	1.03% *	1.03% *	14.52
			∞	1.55% *	2.24% *	3.83% *	9.74
		1.67	0.95	0.67% *	1.52% *	2.23% *	8.22
			∞	0.61% *	1.47% *	2.64% *	6.69
	0.35	1	0.95	———	———	———	———
			∞	4.13% *	5.11% *	6.59% *	12.97
		1.33	0.95	0.0%	1.1%	2.39%	10.55
			∞	0.0%	0.1%	0.21%	11.18
		1.67	0.95	0.07%	0.73%	0.95%	6.54
			∞	0.0%	0.42%	1.02%	6.8
	0.5	1	0.95	———	———	———	———
			∞	0.0%	0.86%	2.22%	12.66
		1.33	0.95	0.01%	0.9%	2.21%	8.4
			∞	0.07%	1.79%	2.8%	9.84
		1.67	0.95	0.63%	0.63%	0.63%	5.28
			∞	0.0%	0.54%	1.55%	10.95
kroA100	0.2	1	0.95	———	———	———	———
			∞	9.09% *	10.15% *	11.48% *	70.62
		1.33	0.95	0.93%	0.93%	0.93%	12.26
			∞	0.0%	0.0%	0.0%	14.51
		1.67	0.95	0.0%	0.18%	0.46%	14.52
			∞	0.0%	2.37%	5.67%	12.21
	0.35	1	0.95	———	———	———	———
			∞	6.33% *	7.44% *	8.08% *	45.21
		1.33	0.95	3.26% *	4.86% *	5.88% *	28.86
			∞	1.56% *	2.76% *	4.09% *	29.93
		1.67	0.95	0.0%	0.0%	0.0%	17.83
			∞	0.0%	0.0%	0.0%	16.89
	0.5	1	0.95	———	———	———	———
			∞	2.97% *	3.97% *	5.41% *	48.71
		1.33	0.95	1.76% *	2.76% *	4.3% *	33.1

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kroA100	0.5	1.33	∞	1.79% *	2.83% *	4.48% *	29.9
		1.67	0.95	0.0%	0.77%	1.99%	14.72
			∞	0.0%	0.49%	0.88%	19.0
pr124	0.2	1	0.95	—	—	—	—
			∞	5.04% *	6.1% *	8.23% *	24.02
		1.33	0.95	9.45% *	10.87% *	11.85% *	14.55
			∞	9.3% *	10.24% *	10.86% *	17.06
		1.67	0.95	10.93% *	11.27% *	11.94% *	19.29
			∞	11.0% *	11.31% *	12.57% *	19.31
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	9.18% *	12.28% *	16.35
		1.33	0.95	—	—	—	—
			∞	0.0%	0.51%	0.88%	15.42
		1.67	0.95	6.69% *	7.9% *	8.78% *	18.02
			∞	5.92% *	7.07% *	7.82% *	19.03
	0.5	1	0.95	—	—	—	—
			∞	4.32% *	6.27% *	6.92% *	31.65
		1.33	0.95	2.88% *	3.56% *	4.35% *	21.48
			∞	4.33% *	5.12% *	6.45% *	15.54
		1.67	0.95	2.04%	2.04%	2.04%	20.32
			∞	0.0%	0.3%	0.88%	21.17
pr152	0.2	1	0.95	—	—	—	—
			∞	14.02% *	14.48% *	15.83% *	72.46
		1.33	0.95	13.46% *	14.17% *	16.35% *	69.16
			∞	13.26% *	14.57% *	15.99% *	54.81
		1.67	0.95	12.57% *	13.26% *	13.98% *	36.46
			∞	12.46% *	13.65% *	15.51% *	38.53
	0.35	1	0.95	—	—	—	—
			∞	23.05% *	26.73% *	38.32% *	40.62
		1.33	0.95	18.38% *	19.13% *	19.7% *	27.69
			∞	19.94% *	21.0% *	22.3% *	45.64
		1.67	0.95	—	—	—	—
			∞	13.89% *	14.7% *	15.15% *	33.51
	0.5	1	0.95	—	—	—	—
			∞	9.25% *	12.1% *	19.04% *	45.45
		1.33	0.95	8.44% *	9.21% *	10.87% *	29.41
			∞	8.43% *	8.87% *	9.41% *	24.85
		1.67	0.95	—	—	—	—
			∞	22.58% *	23.22% *	23.53% *	40.64
rat195	0.2	1	0.95	—	—	—	—
			∞	4.66% *	6.55% *	7.76% *	250.16

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
rat195	0.2	1.33	0.95	—	—	—	—
			∞	3.67% *	4.53% *	5.95% *	124.08
		1.67	0.95	3.94% *	4.94% *	6.28% *	115.08
			∞	3.94% *	5.63% *	9.0% *	103.67
	0.35	1	0.95	—	—	—	—
			∞	3.24% *	5.24% *	7.45% *	172.07
		1.33	0.95	3.65% *	4.7% *	5.38% *	115.14
			∞	3.66% *	5.28% *	7.85% *	159.48
		1.67	0.95	—	—	—	—
			∞	2.91% *	4.62% *	5.87% *	120.37
	0.5	1	0.95	—	—	—	—
			∞	2.77% *	3.77% *	4.58% *	197.64
		1.33	0.95	1.76% *	2.75% *	3.74% *	161.34
			∞	3.11% *	4.44% *	5.86% *	112.7
		1.67	0.95	4.23% *	5.11% *	6.01% *	77.43
			∞	2.62%	3.79%	5.2%	169.47
pr226	0.2	1	0.95	23.27% *	23.73% *	24.2% *	375.52
			∞	20.98% *	25.09% *	28.31% *	218.83
		1.33	0.95	9.36% *	11.15% *	13.58% *	148.88
			∞	9.24% *	13.03% *	17.16% *	153.83
		1.67	0.95	6.76% *	6.83% *	6.92% *	125.85
			∞	6.84% *	7.31% *	7.96% *	142.93
	0.35	1	0.95	20.28% *	20.99% *	21.71% *	119.51
			∞	14.79% *	17.58% *	22.2% *	126.83
		1.33	0.95	11.38% *	11.86% *	12.56% *	94.68
			∞	10.72% *	11.34% *	11.74% *	101.59
		1.67	0.95	9.3% *	9.87% *	10.26% *	80.87
			∞	9.47% *	12.32% *	20.38% *	107.86
	0.5	1	0.95	—	—	—	—
			∞	19.44% *	19.81% *	20.5% *	184.99
		1.33	0.95	11.0% *	13.41% *	17.41% *	194.99
			∞	10.57% *	12.73% *	20.0% *	105.26
		1.67	0.95	10.97% *	11.35% *	12.17% *	164.5
			∞	10.55% *	10.77% *	10.96% *	89.45
Asymmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	13.43% *	16.12% *	21.19% *	15.63
			∞	12.6% *	15.72% *	20.4% *	11.9
		1.33	0.95	2.51%	7.72%	13.01%	9.52
			∞	1.02%	4.18%	7.94%	8.35

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1.67	0.95	0.2%	5.0%	9.43%	3.83
			∞	1.05%	2.14%	3.77%	5.84
	0.35	1	0.95	3.39%	3.39%	3.39%	8.87
			∞	5.39%	9.56%	13.63%	9.33
		1.33	0.95	1.65%	5.61%	10.92%	7.37
			∞	0.0%	5.21%	10.4%	8.41
		1.67	0.95	0.88%	3.1%	6.18%	6.1
			∞	2.51%	6.2%	12.48%	3.69
	0.5	1	0.95	3.64%	6.45%	8.25%	6.24
			∞	3.75%	6.54%	11.83%	5.04
		1.33	0.95	3.36%	9.86%	13.79%	3.36
			∞	1.06%	5.78%	8.92%	4.76
		1.67	0.95	0.65%	7.94%	15.39%	4.96
			∞	2.61%	6.07%	8.57%	4.11
ftv64	0.2	1	0.95	6.37%	9.02%	12.16%	7.98
			∞	6.37%	9.45%	15.51%	9.96
		1.33	0.95	4.37%	6.61%	9.6%	5.26
			∞	3.45%	6.33%	9.81%	4.17
		1.67	0.95	3.75%	5.89%	7.83%	5.83
			∞	3.37%	4.08%	5.17%	5.16
	0.35	1	0.95	3.03%	9.75%	14.08%	7.54
			∞	3.14%	6.49%	11.81%	9.11
		1.33	0.95	1.63%	3.28%	6.93%	5.26
			∞	2.87%	4.29%	5.8%	5.34
		1.67	0.95	4.46%	5.61%	7.29%	4.74
			∞	1.85%	4.97%	5.87%	4.35
	0.5	1	0.95	5.68%	6.95%	9.41%	8.17
			∞	2.7%	8.0%	14.11%	5.77
		1.33	0.95	2.93%	7.25%	10.48%	6.87
			∞	2.06%	5.44%	9.28%	4.18
		1.67	0.95	2.99%	3.95%	6.35%	4.26
			∞	2.71%	5.52%	10.8%	3.34
ft70	0.2	1	0.95	4.76% *	5.46% *	6.37% *	21.93
			∞	5.46% *	6.12% *	7.71% *	14.6
		1.33	0.95	2.68%	4.9%	6.56%	14.48
			∞	1.63%	2.58%	3.22%	21.68
		1.67	0.95	2.22%	3.91%	4.75%	19.44
			∞	2.06%	3.0%	3.9%	13.01
	0.35	1	0.95	1.98%	3.46%	4.41%	15.43
			∞	2.05%	3.41%	5.04%	12.92
		1.33	0.95	2.26%	3.55%	4.14%	11.99

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft70	0.35	1.33	∞	1.92%	2.8%	3.32%	11.61
		1.67	0.95	1.52%	2.74%	4.05%	16.34
			∞	2.59%	3.34%	4.49%	13.72
	0.5	1	0.95	2.39%	2.39%	2.39%	28.58
			∞	2.16%	2.91%	4.3%	18.2
		1.33	0.95	1.99%	2.54%	3.07%	16.56
			∞	1.91%	2.49%	2.87%	10.82
		1.67	0.95	2.55%	2.87%	3.18%	16.4
			∞	1.29%	2.92%	4.18%	12.23
kro124p	0.2	1	0.95	6.74% *	9.53% *	13.11% *	41.79
			∞	5.39% *	8.5% *	11.22% *	53.23
		1.33	0.95	0.98%	3.45%	5.69%	25.97
			∞	0.1%	1.89%	4.7%	26.5
		1.67	0.95	0.93%	2.91%	7.81%	16.07
			∞	0.43%	1.94%	3.78%	21.68
	0.35	1	0.95	—	—	—	—
			∞	7.7% *	10.7% *	15.01% *	50.65
		1.33	0.95	0.23%	1.12%	3.14%	31.15
			∞	1.53%	2.76%	5.12%	25.28
		1.67	0.95	0.33%	2.25%	3.99%	27.24
			∞	0.45%	2.64%	6.35%	20.7
	0.5	1	0.95	4.91% *	7.74% *	12.66% *	26.82
			∞	2.0% *	4.49% *	7.49% *	45.75
		1.33	0.95	1.22%	4.06%	6.9%	25.78
			∞	1.17%	2.86%	5.8%	22.25
		1.67	0.95	0.67%	3.37%	5.4%	27.16
			∞	0.23%	1.89%	5.58%	24.26
ftv170	0.2	1	0.95	—	—	—	—
			∞	10.98% *	15.91% *	20.31% *	116.4
		1.33	0.95	5.84%	11.82%	14.47%	53.82
			∞	3.34%	6.24%	10.66%	37.44
		1.67	0.95	4.5%	7.14%	10.91%	41.23
			∞	5.11%	8.27%	14.14%	33.24
	0.35	1	0.95	8.75% *	11.75% *	13.93% *	91.48
			∞	10.75% *	15.34% *	24.0% *	72.62
		1.33	0.95	7.22%	9.52%	11.76%	59.88
			∞	6.82%	9.57%	11.58%	46.34
		1.67	0.95	7.11%	8.89%	10.85%	36.67
			∞	5.84%	7.46%	9.4%	39.5
	0.5	1	0.95	9.18% *	9.61% *	10.05% *	101.9
			∞	11.72% *	13.07% *	17.15% *	79.35

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv170	0.5	1.33	0.95	8.57%	11.31%	14.39%	63.99
			∞	7.48%	9.96%	11.28%	40.04
		1.67	0.95	5.84%	8.65%	10.74%	41.51
			∞	5.7%	8.43%	10.74%	32.67

B.1.2 Results for a maximum of 2500 iterations

ANN as the constructive heuristic

Table B.4: Computational results, for $\omega = 1/3$ and $MaxIt = 2500$, using the solution obtained through the ANN heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.92% *	5.97% *	6.03
		1.33	0.95	4.4% *	4.63% *	5.04% *	6.9
			∞	5.38% *	5.57% *	5.94% *	7.28
		1.67	0.95	—	—	—	—
			∞	0.83%	2.06%	3.49%	5.71
	0.35	1	0.95	0.0%	0.31%	1.17%	4.9
			∞	0.33%	0.54%	0.87%	6.44
		1.33	0.95	0.0%	0.78%	3.88%	5.36
			∞	0.0%	0.56%	1.41%	5.78
		1.67	0.95	0.0%	0.0%	0.0%	4.5
			∞	0.0%	0.0%	0.0%	3.37
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.63%	3.14%	4.24
		1.33	0.95	—	—	—	—
			∞	0.29%	2.12%	3.09%	3.68
		1.67	0.95	0.0%	1.77%	3.12%	4.88
			∞	0.46%	1.83%	3.87%	4.97
pr76	0.2	1	0.95	—	—	—	—
			∞	1.38% *	1.67% *	1.88% *	8.25
		1.33	0.95	1.54% *	1.89% *	3.25% *	13.25
			∞	1.11% *	2.0% *	4.26% *	8.64
		1.67	0.95	0.39% *	0.8% *	1.27% *	8.47
			∞	0.76% *	1.09% *	2.4% *	9.09
	0.35	1	0.95	—	—	—	—

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr76	0.35	1	∞	4.13% *	4.13% *	4.13% *	13.78
		1.33	0.95	0.0%	0.06%	0.08%	6.5
			∞	0.0%	0.04%	0.08%	8.26
		1.67	0.95	0.84%	1.01%	1.49%	9.83
			∞	0.82%	0.86%	0.91%	6.55
	0.5	1	0.95	—	—	—	—
			∞	0.0%	1.55%	4.84%	10.03
		1.33	0.95	0.0%	0.04%	0.09%	6.63
			∞	0.0%	0.82%	2.52%	7.87
		1.67	0.95	0.55%	0.55%	0.55%	7.35
			∞	0.82%	0.87%	1.08%	8.99
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.31% *	10.43% *	11.38% *	49.03
		1.33	0.95	—	—	—	—
			∞	0.0%	0.36%	1.8%	13.32
		1.67	0.95	—	—	—	—
			∞	0.0%	0.06%	0.09%	12.47
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.33% *	6.33% *	32.71
		1.33	0.95	—	—	—	—
			∞	1.56% *	3.01% *	3.95% *	30.5
		1.67	0.95	0.0%	0.0%	0.0%	15.3
			∞	0.0%	0.06%	0.09%	15.82
	0.5	1	0.95	—	—	—	—
			∞	2.62% *	2.9% *	3.19% *	80.82
		1.33	0.95	1.76% *	1.88% *	2.38% *	51.54
			∞	1.79% *	2.26% *	2.81% *	39.36
		1.67	0.95	0.0%	0.0%	0.0%	21.51
			∞	0.0%	0.02%	0.04%	16.66
pr124	0.2	1	0.95	—	—	—	—
			∞	6.06% *	6.59% *	8.7% *	40.67
		1.33	0.95	9.44% *	9.75% *	11.0% *	22.04
			∞	9.3% *	9.73% *	9.83% *	16.65
		1.67	0.95	10.93% *	10.93% *	10.94% *	18.84
			∞	11.0% *	11.0% *	11.0% *	16.33
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	9.67% *	14.72% *	18.19
		1.33	0.95	—	—	—	—
			∞	0.6%	0.79%	0.88%	21.8
		1.67	0.95	—	—	—	—
			∞	6.56% *	6.65% *	6.85% *	22.57

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr124	0.5	1	0.95	—	—	—	—
			∞	4.86% *	5.68% *	6.18% *	46.34
		1.33	0.95	—	—	—	—
			∞	4.95% *	5.13% *	5.24% *	19.8
		1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	16.13
pr152	0.2	1	0.95	14.57% *	14.7% *	14.8% *	78.34
			∞	14.02% *	14.67% *	15.23% *	88.43
		1.33	0.95	13.37% *	13.99% *	14.79% *	61.38
			∞	13.21% *	13.64% *	14.02% *	62.8
		1.67	0.95	13.07% *	13.9% *	15.25% *	42.45
			∞	12.95% *	13.34% *	14.89% *	42.44
	0.35	1	0.95	23.18% *	24.34% *	26.82% *	75.32
			∞	24.39% *	26.03% *	29.31% *	80.5
		1.33	0.95	19.46% *	19.65% *	19.7% *	42.98
			∞	20.94% *	21.35% *	21.71% *	31.91
		1.67	0.95	14.01% *	16.14% *	18.45% *	79.77
			∞	14.1% *	14.79% *	16.21% *	33.32
	0.5	1	0.95	—	—	—	—
			∞	9.56% *	9.56% *	9.56% *	54.59
		1.33	0.95	8.64% *	10.13% *	11.18% *	57.47
			∞	8.43% *	9.59% *	11.08% *	44.66
		1.67	0.95	—	—	—	—
			∞	22.58% *	23.28% *	23.61% *	58.5
rat195	0.2	1	0.95	—	—	—	—
			∞	5.07% *	6.47% *	9.1% *	181.68
		1.33	0.95	—	—	—	—
			∞	2.95% *	3.6% *	4.57% *	108.43
		1.67	0.95	2.87% *	3.28% *	3.54% *	89.13
			∞	2.55% *	2.83% *	3.09% *	88.87
	0.35	1	0.95	—	—	—	—
			∞	2.7% *	4.12% *	5.45% *	130.16
		1.33	0.95	2.18% *	2.56% *	2.85% *	82.56
			∞	2.72% *	3.25% *	4.99% *	84.63
		1.67	0.95	—	—	—	—
			∞	2.2% *	3.09% *	3.81% *	83.81
	0.5	1	0.95	—	—	—	—
			∞	2.2% *	2.98% *	4.71% *	112.3
		1.33	0.95	1.32% *	2.3% *	3.13% *	121.79
			∞	1.82% *	2.41% *	2.93% *	96.98

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
rat195	0.5	1.67	0.95	2.05% *	2.43% *	3.21% *	142.83
			∞	1.2%	2.4%	3.6%	82.86
pr226	0.2	1	0.95	21.32% *	24.36% *	27.83% *	231.62
			∞	22.53% *	23.05% *	23.83% *	244.96
		1.33	0.95	—	—	—	—
			∞	10.5% *	11.41% *	11.81% *	183.84
		1.67	0.95	6.73% *	7.19% *	7.49% *	146.81
			∞	6.87% *	7.26% *	7.85% *	142.59
	0.35	1	0.95	17.93% *	17.99% *	18.05% *	158.56
			∞	17.73% *	18.57% *	21.12% *	146.21
		1.33	0.95	—	—	—	—
			∞	10.47% *	10.65% *	11.17% *	193.13
		1.67	0.95	—	—	—	—
			∞	8.75% *	9.66% *	12.46% *	153.66
	0.5	1	0.95	17.32% *	19.93% *	28.55% *	690.93
			∞	15.63% *	18.26% *	20.39% *	211.81
		1.33	0.95	10.53% *	13.21% *	17.09% *	156.15
			∞	11.34% *	14.66% *	18.16% *	100.46
		1.67	0.95	—	—	—	—
			∞	10.51% *	10.92% *	11.32% *	152.66

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	16.83% *	20.37% *	22.39% *	13.43
			∞	14.31% *	19.69% *	21.89% *	13.27
		1.33	0.95	4.46%	5.55%	7.24%	4.37
			∞	2.2%	4.46%	6.91%	4.9
		1.67	0.95	0.14%	4.17%	7.28%	7.1
			∞	5.38%	7.01%	9.33%	4.78
	0.35	1	0.95	3.16%	6.8%	10.1%	13.58
			∞	1.89%	4.5%	7.37%	10.04
		1.33	0.95	3.05%	5.78%	7.59%	7.91
			∞	4.67%	7.01%	8.58%	6.71
		1.67	0.95	2.26%	4.88%	8.33%	3.95
			∞	2.61%	6.39%	8.33%	3.23
	0.5	1	0.95	2.9%	7.68%	11.43%	7.68
			∞	0.0%	5.24%	10.18%	7.34
		1.33	0.95	0.0%	3.79%	8.73%	3.36
			∞	3.36%	5.88%	6.53%	2.9
		1.67	0.95	0.0%	6.98%	11.54%	4.46
			∞	2.74%	6.31%	11.34%	3.31

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv64	0.2	1	0.95	—	—	—	—
			∞	3.13%	7.25%	11.58%	10.78
		1.33	0.95	—	—	—	—
			∞	0.97%	4.89%	7.76%	5.52
		1.67	0.95	1.14%	6.07%	10.82%	5.06
			∞	1.25%	4.33%	8.37%	5.87
	0.35	1	0.95	0.05%	3.22%	12.84%	6.95
			∞	0.05%	4.42%	12.24%	6.02
		1.33	0.95	2.87%	3.03%	3.41%	4.25
			∞	3.14%	4.7%	6.83%	5.68
		1.67	0.95	2.18%	3.86%	4.89%	4.78
			∞	2.83%	4.72%	7.5%	4.57
	0.5	1	0.95	1.89%	5.61%	8.86%	12.31
			∞	2.49%	5.74%	8.54%	8.25
		1.33	0.95	0.98%	3.94%	9.83%	5.74
			∞	3.47%	4.95%	6.35%	3.25
		1.67	0.95	0.98%	3.27%	5.65%	7.19
			∞	0.33%	1.75%	3.47%	4.17
ft70	0.2	1	0.95	3.61% *	5.01% *	6.05% *	25.55
			∞	3.9% *	4.31% *	4.76% *	27.76
		1.33	0.95	2.62%	3.35%	4.56%	23.12
			∞	2.29%	3.26%	4.3%	12.23
		1.67	0.95	1.52%	2.87%	4.52%	21.43
			∞	0.63%	2.2%	3.1%	12.39
	0.35	1	0.95	1.46%	2.56%	4.34%	18.3
			∞	2.12%	3.3%	4.31%	17.87
		1.33	0.95	3.15%	3.84%	4.45%	12.11
			∞	1.11%	1.89%	3.25%	20.79
		1.67	0.95	—	—	—	—
			∞	1.95%	2.73%	4.71%	12.66
	0.5	1	0.95	—	—	—	—
			∞	1.83%	2.83%	3.64%	15.2
		1.33	0.95	—	—	—	—
			∞	1.67%	2.68%	3.16%	14.56
		1.67	0.95	—	—	—	—
			∞	1.25%	2.26%	2.92%	10.93
kro124p	0.2	1	0.95	—	—	—	—
			∞	5.77% *	10.04% *	13.19% *	49.82
		1.33	0.95	2.26%	4.69%	6.5%	33.47
			∞	2.87%	6.02%	8.1%	26.89
		1.67	0.95	4.76%	8.38%	9.94%	28.14

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kro124p	0.2	1.67	∞	3.96%	6.42%	11.42%	33.05
	0.35	1	0.95	—	—	—	—
			∞	6.88% *	8.26% *	9.49% *	67.09
		1.33	0.95	5.11%	6.5%	9.02%	32.36
			∞	3.05%	5.25%	8.03%	48.77
		1.67	0.95	4.56%	8.15%	10.92%	26.91
			∞	3.95%	7.89%	11.5%	27.02
	0.5	1	0.95	—	—	—	—
			∞	4.19% *	7.8% *	10.86% *	46.31
		1.33	0.95	7.03%	8.48%	9.89%	23.22
			∞	7.74%	8.87%	10.41%	38.6
		1.67	0.95	5.39%	6.41%	8.05%	37.43
			∞	6.38%	8.91%	10.71%	24.32
ftv170	0.2	1	0.95	—	—	—	—
			∞	12.07% *	17.74% *	20.86% *	116.57
		1.33	0.95	5.44%	8.02%	12.94%	52.82
			∞	5.51%	9.53%	11.71%	62.4
		1.67	0.95	6.31%	11.04%	13.42%	44.5
			∞	6.16%	10.4%	12.84%	41.45
	0.35	1	0.95	—	—	—	—
			∞	8.06% *	10.04% *	14.2% *	58.32
		1.33	0.95	—	—	—	—
			∞	5.48%	7.96%	10.2%	60.47
		1.67	0.95	6.72%	9.2%	10.64%	52.07
			∞	6.03%	8.44%	10.16%	51.67
	0.5	1	0.95	—	—	—	—
			∞	7.43% *	8.35% *	9.36% *	51.68
		1.33	0.95	—	—	—	—
			∞	5.42%	11.06%	13.38%	49.26
		1.67	0.95	7.73%	9.38%	11.51%	51.97
			∞	5.88%	7.44%	10.42%	52.08

AFI as the constructive heuristic

Table B.5: Computational results, for $\omega = 1/3$ and $MaxIt = 2500$, using the solution obtained through the AFI heuristic as the initial solution.

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
Symmetric instances							
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	6.79
		1.33	0.95	4.76% *	4.98% *	5.36% *	5.51
			∞	5.24% *	5.49% *	6.02% *	6.07
		1.67	0.95	0.39%	1.23%	1.99%	7.31
			∞	1.99%	1.99%	1.99%	5.14
	0.35	1	0.95	0.0%	0.25%	0.87%	6.3
			∞	0.0%	0.6%	1.17%	6.48
		1.33	0.95	0.0%	0.0%	0.0%	6.08
			∞	0.0%	1.4%	2.93%	4.56
		1.67	0.95	0.46%	2.26%	3.93%	4.43
			∞	0.0%	1.45%	3.32%	4.31
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.96%	4.79%	4.59
		1.33	0.95	0.0%	1.08%	1.39%	6.64
			∞	0.29%	0.29%	0.29%	3.66
		1.67	0.95	0.0%	1.91%	3.87%	5.76
			∞	0.0%	1.13%	1.65%	5.25
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.9% *	2.21% *	12.27
		1.33	0.95	1.1% *	2.62% *	3.86% *	11.24
			∞	1.13% *	2.4% *	3.25% *	7.31
		1.67	0.95	0.5% *	0.68% *	0.82% *	11.02
			∞	0.33% *	0.51% *	1.1% *	10.42
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.28% *	4.89% *	8.15
		1.33	0.95	0.0%	0.32%	1.53%	7.81
			∞	0.0%	0.28%	0.97%	11.49
		1.67	0.95	0.0%	0.62%	2.82%	11.33
			∞	0.0%	0.17%	0.26%	7.86
	0.5	1	0.95	—	—	—	—
			∞	4.07%	4.07%	4.07%	7.05
		1.33	0.95	2.74%	2.74%	2.74%	8.65
			∞	2.74%	2.74%	2.74%	7.3

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.2	1.67	0.95	—	—	—	—
			∞	0.07%	0.74%	2.82%	7.65
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.77% *	9.56% *	11.68% *	42.59
		1.33	0.95	0.0%	0.01%	0.04%	17.24
			∞	0.0%	0.09%	0.46%	19.38
		1.67	0.95	0.0%	0.09%	0.46%	15.09
			∞	0.0%	0.18%	0.46%	12.99
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	7.1% *	8.19% *	44.9
		1.33	0.95	2.44% *	4.05% *	4.96% *	38.54
			∞	1.91% *	3.23% *	4.4% *	38.41
		1.67	0.95	0.0%	0.06%	0.13%	21.09
			∞	0.0%	0.29%	0.91%	18.4
	0.5	1	0.95	—	—	—	—
			∞	2.66% *	3.98% *	6.08% *	52.51
		1.33	0.95	1.76% *	2.52% *	4.3% *	57.57
			∞	1.69% *	1.77% *	1.81% *	40.19
		1.67	0.95	0.0%	0.01%	0.04%	20.92
			∞	0.0%	0.0%	0.0%	20.9
pr124	0.2	1	0.95	6.84% *	7.9% *	8.93% *	35.83
			∞	6.06% *	6.59% *	8.7% *	42.71
		1.33	0.95	9.44% *	9.75% *	11.0% *	23.9
			∞	9.23% *	9.6% *	10.86% *	22.21
		1.67	0.95	10.86% *	11.01% *	11.47% *	28.64
			∞	11.0% *	11.0% *	11.0% *	20.82
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	20.92
		1.33	0.95	—	—	—	—
			∞	0.0%	0.61%	0.88%	20.66
		1.67	0.95	6.69% *	7.01% *	7.3% *	40.49
			∞	5.92% *	6.54% *	7.44% *	22.97
	0.5	1	0.95	—	—	—	—
			∞	5.27% *	5.27% *	5.27% *	28.52
		1.33	0.95	1.98% *	2.51% *	2.87% *	31.74
			∞	4.33% *	4.69% *	5.24% *	21.18
		1.67	0.95	0.0%	0.59%	0.78%	21.81
			∞	0.0%	0.63%	0.88%	19.26
pr152	0.2	1	0.95	—	—	—	—
			∞	13.95% *	14.11% *	14.54% *	76.47
		1.33	0.95	14.48% *	15.42% *	15.79% *	60.51

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr152	0.2	1.33	∞	13.5% *	14.58% *	15.8% *	46.95
		1.67	0.95	13.1% *	13.3% *	14.12% *	28.81
			∞	12.36% *	12.78% *	13.33% *	47.7
	0.35	1	0.95	—	—	—	—
			∞	23.76% *	24.08% *	24.35% *	54.67
		1.33	0.95	18.38% *	19.02% *	20.08% *	45.58
			∞	20.51% *	20.54% *	20.6% *	40.47
		1.67	0.95	—	—	—	—
			∞	14.1% *	14.86% *	15.74% *	46.16
	0.5	1	0.95	—	—	—	—
			∞	9.25% *	9.26% *	9.33% *	61.12
		1.33	0.95	9.04% *	9.24% *	9.33% *	36.69
			∞	9.03% *	9.24% *	9.32% *	42.9
		1.67	0.95	—	—	—	—
			∞	22.91% *	23.36% *	24.0% *	47.17
rat195	0.2	1	0.95	—	—	—	—
			∞	5.52% *	6.8% *	8.12% *	256.8
		1.33	0.95	—	—	—	—
			∞	3.85% *	4.91% *	6.58% *	126.07
		1.67	0.95	3.77% *	4.47% *	5.38% *	142.18
			∞	2.19% *	4.44% *	6.09% *	138.05
	0.35	1	0.95	—	—	—	—
			∞	3.15% *	4.17% *	5.01% *	160.68
		1.33	0.95	3.38% *	4.75% *	5.92% *	124.35
			∞	3.7% *	4.78% *	5.93% *	127.79
		1.67	0.95	—	—	—	—
			∞	3.32% *	3.97% *	5.25% *	160.4
	0.5	1	0.95	—	—	—	—
			∞	2.73% *	2.97% *	3.34% *	210.07
		1.33	0.95	2.25% *	3.31% *	5.07% *	147.45
			∞	2.35% *	3.41% *	4.04% *	138.22
		1.67	0.95	3.07% *	3.71% *	4.72% *	166.53
			∞	2.22%	3.72%	4.72%	101.52
pr226	0.2	1	0.95	—	—	—	—
			∞	22.37% *	24.81% *	26.08% *	294.33
		1.33	0.95	9.17% *	9.92% *	10.34% *	277.02
			∞	9.43% *	10.37% *	11.58% *	326.59
		1.67	0.95	6.76% *	6.87% *	7.04% *	114.76
			∞	6.84% *	6.88% *	6.9% *	128.58
	0.35	1	0.95	—	—	—	—
			∞	14.56% *	15.12% *	15.44% *	239.37

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr226	0.35	1.33	0.95	11.32% *	11.35% *	11.42% *	158.3
pr226	0.35	1.33	∞	10.47% *	10.51% *	10.56% *	142.31
pr226	0.35	1.67	0.95	9.07% *	9.11% *	9.18% *	142.41
pr226	0.35	1.67	∞	8.77% *	8.81% *	8.85% *	141.28
pr226	0.5	1	0.95	—	—	—	—
pr226	0.5	1	∞	19.4% *	19.74% *	20.34% *	280.94
pr226	0.5	1.33	0.95	10.72% *	11.16% *	11.43% *	273.89
pr226	0.5	1.33	∞	10.51% *	10.62% *	10.87% *	195.56
pr226	0.5	1.67	0.95	10.96% *	11.07% *	11.19% *	147.64
pr226	0.5	1.67	∞	10.52% *	10.56% *	10.59% *	149.7

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	—	—	—	—
ft53	0.2	1	∞	13.22% *	15.84% *	18.07% *	16.34
ft53	0.2	1.33	0.95	0.52%	4.68%	6.59%	4.64
ft53	0.2	1.33	∞	4.89%	7.28%	9.2%	4.15
ft53	0.2	1.67	0.95	0.7%	5.33%	10.52%	4.75
ft53	0.2	1.67	∞	0.14%	4.37%	9.84%	4.81
ft53	0.35	1	0.95	—	—	—	—
ft53	0.35	1	∞	7.03%	8.38%	10.27%	6.26
ft53	0.35	1.33	0.95	1.82%	5.76%	10.29%	7.92
ft53	0.35	1.33	∞	0.85%	3.64%	7.95%	10.2
ft53	0.35	1.67	0.95	0.0%	2.06%	4.48%	6.69
ft53	0.35	1.67	∞	0.0%	1.16%	2.49%	4.37
ft53	0.5	1	0.95	2.61%	5.58%	6.75%	5.25
ft53	0.5	1	∞	2.61%	6.41%	9.75%	4.19
ft53	0.5	1.33	0.95	1.12%	2.87%	5.43%	3.81
ft53	0.5	1.33	∞	2.3%	3.36%	3.98%	3.91
ft53	0.5	1.67	0.95	3.19%	3.4%	3.56%	2.12
ft53	0.5	1.67	∞	2.88%	3.39%	3.62%	2.65
ftv64	0.2	1	0.95	7.81%	9.64%	12.32%	9.11
ftv64	0.2	1	∞	3.61%	6.95%	10.2%	7.93
ftv64	0.2	1.33	0.95	0.97%	4.28%	6.42%	5.14
ftv64	0.2	1.33	∞	4.04%	5.22%	6.25%	4.57
ftv64	0.2	1.67	0.95	—	—	—	—
ftv64	0.2	1.67	∞	3.32%	5.64%	7.5%	3.8
ftv64	0.35	1	0.95	0.98%	4.86%	8.88%	9.07
ftv64	0.35	1	∞	3.14%	9.62%	13.87%	6.76
ftv64	0.35	1.33	0.95	2.82%	3.99%	8.07%	4.87
ftv64	0.35	1.33	∞	2.82%	4.51%	6.55%	5.0

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.35	1.67	0.95	3.37%	4.57%	6.04%	3.66
			∞	2.28%	3.58%	5.6%	4.18
	0.5	1	0.95	———	———	———	———
			∞	0.27%	3.35%	11.84%	6.46
		1.33	0.95	0.87%	1.32%	2.01%	4.11
			∞	0.0%	1.64%	3.75%	4.77
		1.67	0.95	———	———	———	———
			∞	0.71%	1.69%	3.15%	6.79
ft70	0.2	1	0.95	3.64% *	5.06% *	5.89% *	21.44
			∞	4.16% *	5.13% *	5.98% *	18.97
		1.33	0.95	2.54%	3.78%	5.38%	18.89
			∞	1.48%	2.43%	4.26%	20.0
		1.67	0.95	0.66%	2.95%	4.3%	22.94
			∞	1.53%	2.39%	3.12%	17.23
	0.35	1	0.95	1.12%	2.19%	3.18%	20.59
			∞	1.14%	2.72%	3.65%	17.02
		1.33	0.95	1.22%	2.77%	3.82%	15.12
			∞	2.12%	2.75%	4.02%	13.81
		1.67	0.95	1.66%	2.21%	2.54%	18.51
			∞	1.93%	2.67%	3.59%	17.43
	0.5	1	0.95	2.0%	2.8%	3.49%	23.57
			∞	2.06%	2.79%	3.21%	17.83
		1.33	0.95	1.53%	2.39%	3.1%	14.17
			∞	1.54%	2.17%	3.35%	13.13
		1.67	0.95	2.65%	3.05%	3.38%	15.01
			∞	2.07%	2.81%	3.76%	15.31
kro124p	0.2	1	0.95	4.42% *	5.12% *	6.2% *	70.22
			∞	5.98% *	7.02% *	8.19% *	37.21
		1.33	0.95	0.15%	2.95%	3.97%	19.9
			∞	1.56%	2.54%	3.39%	17.76
		1.67	0.95	0.03%	1.14%	3.05%	22.43
			∞	0.18%	1.48%	2.89%	22.37
	0.35	1	0.95	———	———	———	———
			∞	7.64% *	9.68% *	12.76% *	59.59
		1.33	0.95	0.43%	1.86%	2.6%	26.58
			∞	1.24%	2.22%	3.5%	21.02
		1.67	0.95	0.1%	1.95%	4.18%	28.24
			∞	0.31%	2.39%	4.78%	22.22
	0.5	1	0.95	2.17% *	4.54% *	7.08% *	49.3
			∞	0.86% *	3.97% *	6.04% *	49.91
		1.33	0.95	1.98%	3.23%	4.47%	29.38

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.5	1.33	∞	2.41%	2.96%	4.07%	35.78
		1.67	0.95	0.03%	2.66%	4.05%	21.35
			∞	0.43%	2.71%	3.76%	18.73
ftv170	0.2	1	0.95	—	—	—	—
			∞	6.93% *	11.07% *	17.21% *	65.2
		1.33	0.95	5.73%	6.85%	7.51%	41.91
			∞	3.7%	5.72%	8.59%	48.05
		1.67	0.95	6.56%	8.16%	10.08%	32.82
			∞	4.64%	7.68%	13.34%	46.54
	0.35	1	0.95	9.81% *	12.83% *	15.17% *	91.85
			∞	10.2% *	13.49% *	17.94% *	91.38
		1.33	0.95	3.96%	5.32%	9.11%	66.99
			∞	5.81%	7.49%	9.26%	49.63
		1.67	0.95	2.79%	5.51%	8.13%	45.05
			∞	1.63%	4.65%	7.84%	67.38
	0.5	1	0.95	8.09% *	11.59% *	16.25% *	138.39
			∞	8.74% *	9.84% *	13.03% *	113.75
		1.33	0.95	9.76%	10.67%	12.73%	74.28
			∞	7.85%	9.19%	12.73%	72.19
		1.67	0.95	3.34%	5.74%	7.01%	44.33
			∞	2.9%	3.46%	4.1%	39.55

ARI as the constructive heuristic

Table B.6: Computational results, for $\omega = 1/3$ and $MaxIt = 2500$, using the solution obtained through the ARI heuristic as the initial solution.

Symmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	10.74
		1.33	0.95	4.27% *	4.95% *	6.62% *	10.65
			∞	5.38% *	5.46% *	5.74% *	8.25
		1.67	0.95	0.0%	0.89%	1.99%	17.08
			∞	1.99%	2.49%	3.26%	11.2
	0.35	1	0.95	0.0%	0.27%	0.87%	14.85
			∞	0.37%	1.01%	1.17%	9.91
		1.33	0.95	0.0%	0.72%	2.36%	9.28
			∞	0.0%	1.43%	2.93%	7.6

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.35	1.67	0.95	0.0%	1.28%	2.46%	7.64
			∞	0.0%	0.09%	0.46%	10.03
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	8.61
		1.33	0.95	1.34%	1.34%	1.34%	10.51
			∞	0.0%	0.45%	1.37%	7.47
		1.67	0.95	0.51%	1.96%	3.26%	9.5
			∞	0.0%	0.37%	1.33%	12.64
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.64% *	1.77% *	18.52
		1.33	0.95	1.03% *	1.03% *	1.03% *	26.87
			∞	1.55% *	1.99% *	3.25% *	17.07
		1.67	0.95	0.5% *	1.21% *	2.7% *	15.88
			∞	0.43% *	0.75% *	1.6% *	17.45
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.78% *	5.86% *	25.47
		1.33	0.95	0.0%	0.61%	1.44%	12.77
			∞	0.0%	0.27%	0.97%	13.62
		1.67	0.95	0.0%	0.72%	1.02%	19.7
			∞	0.0%	0.44%	1.21%	14.4
	0.5	1	0.95	—	—	—	—
			∞	0.07%	0.85%	2.22%	22.36
		1.33	0.95	0.0%	1.79%	2.74%	18.32
			∞	0.0%	1.24%	2.74%	26.0
		1.67	0.95	0.55%	0.59%	0.62%	19.86
			∞	0.0%	0.36%	0.83%	15.78
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.25% *	9.38% *	11.09% *	109.47
		1.33	0.95	0.0%	0.0%	0.0%	30.59
			∞	0.0%	0.46%	1.83%	34.48
		1.67	0.95	0.0%	0.03%	0.09%	28.5
			∞	0.0%	0.0%	0.0%	33.08
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.92% *	7.8% *	109.73
		1.33	0.95	2.44% *	4.08% *	6.05% *	64.74
			∞	1.56% *	2.27% *	3.6% *	69.18
		1.67	0.95	0.0%	0.06%	0.09%	39.21
			∞	0.0%	0.04%	0.09%	32.15
	0.5	1	0.95	—	—	—	—
			∞	2.62% *	3.07% *	3.74% *	94.1
		1.33	0.95	1.76% *	3.04% *	4.15% *	74.91

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kroA100	0.5	1.33	∞	1.78% *	2.02% *	3.0% *	79.46
		1.67	0.95	0.0%	1.05%	3.98%	40.11
			∞	0.0%	0.17%	0.8%	39.87
pr124	0.2	1	0.95	—	—	—	—
			∞	5.04% *	6.26% *	7.85% *	76.42
		1.33	0.95	9.44% *	9.55% *	9.97% *	40.13
			∞	9.3% *	10.24% *	10.86% *	46.45
		1.67	0.95	10.93% *	10.93% *	10.94% *	45.37
			∞	11.0% *	11.68% *	12.8% *	45.99
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	39.69
		1.33	0.95	—	—	—	—
			∞	0.0%	0.51%	0.88%	34.99
		1.67	0.95	7.08% *	7.87% *	9.09% *	35.82
			∞	5.92% *	7.11% *	8.37% *	64.24
	0.5	1	0.95	—	—	—	—
			∞	5.27% *	5.72% *	6.87% *	50.39
		1.33	0.95	1.98% *	2.76% *	3.51% *	56.88
			∞	4.33% *	4.65% *	5.25% *	43.7
		1.67	0.95	0.0%	0.0%	0.0%	32.88
			∞	0.0%	0.87%	2.32%	40.03
pr152	0.2	1	0.95	—	—	—	—
			∞	14.02% *	14.52% *	16.19% *	128.94
		1.33	0.95	13.69% *	13.69% *	13.69% *	194.3
			∞	13.17% *	13.98% *	15.8% *	108.67
		1.67	0.95	12.8% *	13.81% *	15.21% *	57.68
			∞	12.36% *	13.23% *	15.35% *	58.47
	0.35	1	0.95	—	—	—	—
			∞	23.95% *	24.36% *	24.97% *	115.21
		1.33	0.95	18.38% *	19.19% *	19.7% *	50.01
			∞	19.85% *	20.54% *	21.21% *	61.8
		1.67	0.95	—	—	—	—
			∞	13.89% *	14.52% *	14.84% *	67.27
	0.5	1	0.95	—	—	—	—
			∞	9.33% *	10.24% *	12.25% *	121.77
		1.33	0.95	8.44% *	9.44% *	10.09% *	97.89
			∞	8.43% *	9.38% *	9.85% *	81.05
		1.67	0.95	—	—	—	—
			∞	22.95% *	23.5% *	23.97% *	73.86
rat195	0.2	1	0.95	—	—	—	—
			∞	3.95% *	7.48% *	8.88% *	613.16

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
rat195	0.2	1.33	0.95	—	—	—	—
			∞	2.86% *	3.66% *	5.24% *	329.79
		1.67	0.95	3.09% *	3.57% *	3.99% *	255.28
			∞	2.24% *	4.7% *	8.91% *	196.28
	0.35	1	0.95	—	—	—	—
			∞	3.72% *	4.6% *	5.27% *	250.09
		1.33	0.95	3.2% *	4.37% *	6.1% *	255.49
			∞	3.52% *	4.08% *	4.99% *	276.2
		1.67	0.95	—	—	—	—
			∞	2.96% *	4.13% *	5.47% *	270.79
	0.5	1	0.95	—	—	—	—
			∞	1.72% *	3.32% *	4.8% *	371.02
		1.33	0.95	1.28% *	2.78% *	3.92% *	275.97
			∞	1.15% *	2.81% *	4.93% *	250.7
		1.67	0.95	2.58% *	3.83% *	5.03% *	233.0
			∞	2.09%	3.63%	4.72%	234.86
pr226	0.2	1	0.95	23.43% *	26.18% *	28.93% *	563.23
			∞	22.41% *	25.38% *	28.44% *	400.92
		1.33	0.95	11.48% *	11.7% *	11.93% *	304.15
			∞	9.17% *	12.22% *	17.34% *	406.13
		1.67	0.95	6.72% *	7.04% *	7.58% *	272.89
			∞	6.86% *	8.54% *	11.24% *	222.5
	0.35	1	0.95	14.04% *	15.17% *	17.38% *	486.58
			∞	14.52% *	14.7% *	14.99% *	317.47
		1.33	0.95	11.34% *	12.0% *	12.56% *	183.08
			∞	10.48% *	11.51% *	14.71% *	254.29
		1.67	0.95	9.07% *	9.07% *	9.07% *	217.8
			∞	8.79% *	9.45% *	11.92% *	241.13
	0.5	1	0.95	—	—	—	—
			∞	18.98% *	19.46% *	20.14% *	490.25
		1.33	0.95	10.98% *	13.05% *	16.95% *	229.34
			∞	10.54% *	10.74% *	11.09% *	229.44
		1.67	0.95	11.0% *	12.1% *	14.65% *	223.33
			∞	10.52% *	10.7% *	11.1% *	279.47

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	13.04% *	14.64% *	17.32% *	21.6
			∞	11.09% *	14.42% *	17.17% *	30.17
		1.33	0.95	3.2%	5.33%	8.58%	15.0
			∞	4.43%	6.32%	6.98%	10.45

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1.67	0.95	0.0%	4.25%	10.21%	8.84
			∞	0.2%	1.07%	2.65%	7.28
	0.35	1	0.95	4.67%	6.41%	7.99%	14.15
			∞	1.98%	4.1%	7.92%	15.77
		1.33	0.95	4.5%	6.97%	8.55%	11.33
			∞	3.19%	7.94%	11.63%	14.64
		1.67	0.95	0.88%	3.13%	6.02%	8.67
			∞	0.88%	3.88%	9.51%	7.55
	0.5	1	0.95	4.32%	6.39%	8.47%	10.01
			∞	2.61%	4.87%	7.18%	6.79
		1.33	0.95	1.71%	4.55%	9.92%	6.98
			∞	1.12%	4.88%	9.12%	9.5
		1.67	0.95	0.06%	4.37%	11.17%	6.95
			∞	3.43%	5.05%	7.14%	8.06
ftv64	0.2	1	0.95	4.62%	6.49%	10.25%	20.33
			∞	3.56%	7.71%	13.01%	18.6
		1.33	0.95	4.91%	6.25%	8.41%	11.87
			∞	0.0%	5.53%	9.27%	12.35
		1.67	0.95	3.86%	5.4%	7.83%	8.21
			∞	1.03%	3.33%	5.33%	9.4
	0.35	1	0.95	0.98%	7.89%	13.06%	15.62
			∞	2.76%	5.21%	7.53%	16.27
		1.33	0.95	2.38%	3.59%	5.31%	7.87
			∞	2.87%	4.46%	8.23%	8.82
		1.67	0.95	3.15%	4.71%	7.88%	5.49
			∞	2.28%	3.81%	6.25%	9.55
	0.5	1	0.95	7.78%	8.54%	9.14%	16.82
			∞	1.19%	4.18%	7.19%	13.19
		1.33	0.95	0.81%	2.65%	4.78%	9.5
			∞	0.98%	2.27%	4.18%	10.82
		1.67	0.95	2.01%	5.04%	8.85%	7.73
			∞	2.66%	4.26%	6.13%	9.98
ft70	0.2	1	0.95	4.84% *	4.84% *	4.84% *	48.75
			∞	3.52% *	4.33% *	5.38% *	46.52
		1.33	0.95	2.13%	3.53%	6.01%	51.67
			∞	1.25%	1.95%	2.63%	37.83
		1.67	0.95	1.65%	3.29%	5.3%	37.28
			∞	2.05%	2.38%	2.9%	37.06
	0.35	1	0.95	1.56%	2.71%	3.85%	25.8
			∞	1.76%	2.52%	3.43%	23.62
		1.33	0.95	1.98%	2.66%	3.66%	33.02

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft70	0.35	1.33	∞	2.36%	2.91%	4.45%	32.67
		1.67	0.95	1.81%	2.12%	2.65%	31.57
			∞	2.08%	3.04%	4.98%	22.7
	0.5	1	0.95	1.53%	2.69%	4.01%	49.71
			∞	1.06%	2.31%	2.81%	34.88
		1.33	0.95	2.12%	2.14%	2.17%	26.53
			∞	0.94%	2.48%	3.93%	28.42
		1.67	0.95	1.53%	1.53%	1.53%	22.78
			∞	1.16%	2.23%	3.55%	26.14
kro124p	0.2	1	0.95	6.42% *	7.83% *	8.46% *	91.71
			∞	4.11% *	7.28% *	10.01% *	80.91
		1.33	0.95	0.53%	2.39%	5.48%	42.18
			∞	0.1%	0.59%	1.71%	39.95
		1.67	0.95	1.4%	3.09%	4.74%	54.52
			∞	0.1%	2.61%	5.52%	43.76
	0.35	1	0.95	—	—	—	—
			∞	7.86% *	11.52% *	14.42% *	71.66
		1.33	0.95	0.86%	1.17%	1.55%	48.15
			∞	0.36%	1.76%	5.06%	63.77
		1.67	0.95	0.31%	2.41%	5.51%	50.45
			∞	0.18%	0.74%	1.85%	30.23
	0.5	1	0.95	0.78% *	4.03% *	6.88% *	104.61
			∞	1.64% *	3.52% *	6.52% *	90.43
		1.33	0.95	0.03%	1.83%	4.94%	64.83
			∞	0.93%	3.11%	6.68%	49.4
		1.67	0.95	0.1%	1.89%	2.69%	74.86
			∞	0.91%	2.73%	5.31%	30.52
ftv170	0.2	1	0.95	—	—	—	—
			∞	10.28% *	16.55% *	21.99% *	138.52
		1.33	0.95	5.95%	8.03%	10.26%	95.82
			∞	1.56%	5.79%	8.99%	107.82
		1.67	0.95	2.76%	5.9%	11.17%	86.15
			∞	3.95%	7.08%	8.96%	66.98
	0.35	1	0.95	9.26% *	9.26% *	9.26% *	268.76
			∞	8.57% *	11.68% *	14.52% *	114.17
		1.33	0.95	3.01%	6.74%	14.23%	92.58
			∞	3.52%	6.63%	9.15%	94.12
		1.67	0.95	3.59%	7.08%	9.22%	82.43
			∞	3.96%	7.34%	10.31%	110.95
	0.5	1	0.95	5.55% *	5.55% *	5.55% *	131.17
			∞	6.95% *	11.96% *	15.03% *	143.24

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv170	0.5	1.33	0.95	5.82%	8.98%	11.97%	105.14
			∞	8.35%	9.18%	11.86%	91.24
		1.67	0.95	7.11%	7.91%	8.53%	75.17
			∞	3.59%	6.9%	10.31%	99.05

B.1.3 Results for a maximum of 5000 iterations

ANN as the constructive heuristic

Table B.7: Computational results, for $\omega = 1/3$ and $MaxIt = 5000$, using the solution obtained through the ANN heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	11.89
		1.33	0.95	4.4% *	4.4% *	4.4% *	8.34
			∞	5.38% *	5.58% *	5.74% *	9.16
		1.67	0.95	—	—	—	—
			∞	0.0%	1.4%	3.26%	14.72
	0.35	1	0.95	0.0%	0.6%	1.17%	11.09
			∞	0.0%	0.41%	1.17%	11.06
		1.33	0.95	0.0%	0.0%	0.0%	10.11
			∞	0.0%	1.48%	2.36%	7.87
		1.67	0.95	0.0%	0.23%	1.14%	6.99
			∞	0.0%	0.09%	0.46%	7.25
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.53%	2.64%	6.72
		1.33	0.95	—	—	—	—
			∞	1.11%	1.11%	1.11%	6.37
		1.67	0.95	1.33%	1.33%	1.33%	7.64
			∞	0.0%	0.81%	2.74%	8.21
pr76	0.2	1	0.95	—	—	—	—
			∞	1.38% *	1.44% *	1.45% *	17.34
		1.33	0.95	1.54% *	1.88% *	3.25% *	19.31
			∞	1.04% *	1.75% *	3.05% *	26.25
		1.67	0.95	0.39% *	0.73% *	1.27% *	19.93
			∞	0.52% *	0.89% *	2.4% *	22.27
	0.35	1	0.95	—	—	—	—

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.35	1	∞	4.13% *	4.43% *	4.89% *	19.77
		1.33	0.95	0.01%	0.04%	0.08%	11.34
			∞	0.0%	0.03%	0.08%	13.52
		1.67	0.95	0.0%	0.79%	1.08%	13.25
			∞	0.82%	0.84%	0.91%	15.33
	0.5	1	0.95	—	—	—	—
			∞	0.08%	1.49%	2.74%	28.01
		1.33	0.95	0.0%	0.01%	0.02%	13.47
			∞	0.08%	1.47%	2.74%	16.96
		1.67	0.95	0.55%	0.55%	0.55%	13.41
			∞	0.26%	0.76%	1.08%	14.57
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.45% *	9.06% *	9.73% *	136.74
		1.33	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	31.22
		1.67	0.95	—	—	—	—
			∞	0.0%	0.04%	0.09%	24.97
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	7.01% *	8.25% *	86.79
		1.33	0.95	—	—	—	—
			∞	1.56% *	1.77% *	2.62% *	68.51
		1.67	0.95	0.0%	0.0%	0.0%	30.2
			∞	0.0%	0.04%	0.09%	31.71
	0.5	1	0.95	—	—	—	—
			∞	2.52% *	2.77% *	3.33% *	107.23
		1.33	0.95	1.76% *	1.88% *	2.38% *	104.94
			∞	1.78% *	1.79% *	1.81% *	66.24
		1.67	0.95	0.0%	0.29%	0.71%	31.74
			∞	0.0%	0.0%	0.0%	35.67
pr124	0.2	1	0.95	—	—	—	—
			∞	5.04% *	5.86% *	6.06% *	63.3
		1.33	0.95	9.44% *	10.07% *	11.0% *	34.78
			∞	9.31% *	9.98% *	11.08% *	35.99
		1.67	0.95	10.93% *	10.93% *	10.94% *	36.35
			∞	10.92% *	11.0% *	11.1% *	29.57
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	33.82
		1.33	0.95	—	—	—	—
			∞	0.78%	0.84%	0.88%	36.35
		1.67	0.95	—	—	—	—
			∞	6.56% *	6.63% *	6.75% *	39.8

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr124	0.5	1	0.95	—	—	—	—
			∞	5.05% *	5.45% *	6.18% *	78.11
		1.33	0.95	—	—	—	—
			∞	4.95% *	5.11% *	5.24% *	40.91
		1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	28.24
pr152	0.2	1	0.95	13.48% *	13.96% *	14.67% *	113.64
			∞	14.02% *	14.62% *	15.19% *	204.19
		1.33	0.95	13.19% *	13.39% *	13.69% *	162.68
			∞	13.21% *	13.98% *	15.18% *	111.48
		1.67	0.95	13.07% *	13.53% *	15.0% *	83.0
			∞	12.36% *	13.04% *	13.95% *	101.32
	0.35	1	0.95	22.92% *	23.7% *	24.33% *	120.46
			∞	24.39% *	27.11% *	30.17% *	106.79
		1.33	0.95	19.46% *	19.65% *	19.7% *	78.63
			∞	20.94% *	21.14% *	21.19% *	76.35
		1.67	0.95	14.64% *	15.1% *	16.89% *	58.13
			∞	14.1% *	15.07% *	16.67% *	58.77
	0.5	1	0.95	—	—	—	—
			∞	9.56% *	9.56% *	9.56% *	64.0
		1.33	0.95	8.64% *	10.29% *	11.09% *	85.93
			∞	9.34% *	9.5% *	9.56% *	73.0
		1.67	0.95	—	—	—	—
			∞	23.61% *	23.61% *	23.61% *	64.53
rat195	0.2	1	0.95	—	—	—	—
			∞	5.07% *	6.1% *	7.31% *	361.05
		1.33	0.95	—	—	—	—
			∞	2.86% *	3.43% *	3.98% *	227.83
		1.67	0.95	2.73% *	3.27% *	3.77% *	146.23
			∞	2.55% *	2.76% *	2.96% *	130.29
	0.35	1	0.95	—	—	—	—
			∞	2.08% *	3.18% *	5.14% *	289.48
		1.33	0.95	1.47% *	2.18% *	2.58% *	187.32
			∞	1.65% *	2.2% *	2.72% *	183.95
		1.67	0.95	—	—	—	—
			∞	2.2% *	2.77% *	3.63% *	198.27
	0.5	1	0.95	—	—	—	—
			∞	1.32% *	3.11% *	4.84% *	274.86
		1.33	0.95	1.19% *	1.81% *	3.04% *	229.81
			∞	1.86% *	2.28% *	2.89% *	172.68

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
rat195	0.5	1.67	0.95	1.74% *	2.53% *	2.94% *	183.09
			∞	1.73%	1.88%	2.05%	138.13
pr226	0.2	1	0.95	22.2% *	22.41% *	22.8% *	615.85
			∞	20.98% *	23.08% *	26.16% *	434.57
		1.33	0.95	—	—	—	—
			∞	10.61% *	11.18% *	11.74% *	254.31
		1.67	0.95	6.71% *	7.02% *	7.57% *	258.67
			∞	6.83% *	7.51% *	9.88% *	282.82
	0.35	1	0.95	17.93% *	18.46% *	20.4% *	342.82
			∞	17.73% *	18.0% *	18.31% *	335.91
		1.33	0.95	—	—	—	—
			∞	10.48% *	10.94% *	11.21% *	248.01
		1.67	0.95	—	—	—	—
			∞	8.76% *	9.16% *	9.38% *	207.57
	0.5	1	0.95	14.22% *	15.36% *	18.35% *	1010.99
			∞	14.14% *	22.66% *	32.84% *	326.29
		1.33	0.95	10.46% *	10.73% *	11.24% *	380.99
			∞	10.52% *	10.76% *	11.21% *	315.25
		1.67	0.95	—	—	—	—
			∞	10.51% *	10.66% *	11.19% *	217.56

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	12.94% *	14.35% *	16.0% *	30.17
			∞	11.34% *	14.86% *	16.96% *	25.17
		1.33	0.95	3.0%	5.15%	7.33%	10.09
			∞	1.74%	4.55%	6.75%	8.03
		1.67	0.95	0.14%	2.91%	6.15%	10.82
			∞	1.47%	5.38%	10.78%	9.53
	0.35	1	0.95	3.39%	6.04%	9.3%	17.54
			∞	0.85%	5.94%	12.48%	19.9
		1.33	0.95	3.05%	6.27%	10.89%	13.78
			∞	0.48%	1.76%	3.12%	19.92
		1.67	0.95	0.88%	4.95%	8.33%	6.17
			∞	0.0%	5.34%	9.41%	6.16
	0.5	1	0.95	0.0%	3.91%	8.81%	13.36
			∞	0.06%	3.43%	5.56%	8.61
		1.33	0.95	0.0%	5.34%	9.27%	6.87
			∞	0.0%	4.78%	7.7%	6.74
		1.67	0.95	1.03%	4.05%	6.52%	8.56
			∞	0.0%	3.63%	7.83%	7.82

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv64	0.2	1	0.95	—	—	—	—
			∞	6.05%	7.24%	9.13%	14.31
		1.33	0.95	—	—	—	—
			∞	0.38%	2.96%	6.15%	19.12
		1.67	0.95	3.37%	5.39%	10.28%	12.59
			∞	1.31%	4.72%	8.43%	11.19
	0.35	1	0.95	0.05%	1.54%	4.39%	14.25
			∞	0.05%	5.11%	11.32%	15.93
		1.33	0.95	1.68%	3.27%	5.42%	11.3
			∞	3.74%	5.32%	6.99%	15.22
		1.67	0.95	1.31%	3.2%	6.36%	11.31
			∞	1.25%	4.14%	5.76%	11.5
	0.5	1	0.95	1.62%	4.39%	7.03%	24.24
			∞	1.84%	4.91%	8.05%	15.75
		1.33	0.95	2.06%	3.57%	4.29%	8.97
			∞	1.19%	5.12%	11.56%	11.03
		1.67	0.95	0.87%	4.77%	10.04%	12.46
			∞	0.71%	1.86%	3.09%	7.11
ft70	0.2	1	0.95	4.05% *	5.27% *	6.63% *	29.67
			∞	2.92% *	3.91% *	4.86% *	44.63
		1.33	0.95	2.78%	3.82%	6.14%	43.61
			∞	2.19%	2.87%	3.57%	28.08
		1.67	0.95	1.1%	2.25%	3.02%	38.0
			∞	0.96%	2.78%	4.29%	35.41
	0.35	1	0.95	1.77%	2.25%	2.92%	39.79
			∞	2.58%	3.02%	3.29%	29.43
		1.33	0.95	1.36%	2.19%	3.07%	29.97
			∞	1.13%	2.18%	2.8%	33.17
		1.67	0.95	—	—	—	—
			∞	1.63%	2.87%	3.46%	26.87
	0.5	1	0.95	—	—	—	—
			∞	1.55%	2.2%	2.77%	25.17
		1.33	0.95	—	—	—	—
			∞	0.64%	2.19%	3.39%	36.73
		1.67	0.95	—	—	—	—
			∞	1.45%	2.0%	3.15%	31.41
kro124p	0.2	1	0.95	—	—	—	—
			∞	6.0% *	8.83% *	11.0% *	116.3
		1.33	0.95	5.52%	6.12%	7.69%	62.76
			∞	0.91%	4.96%	7.08%	43.56
		1.67	0.95	3.44%	5.67%	10.3%	55.7

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.2	1.67	∞	4.12%	7.31%	10.23%	83.13
	0.35	1	0.95	—	—	—	—
			∞	5.91% *	6.39% *	7.18% *	103.27
		1.33	0.95	2.96%	4.66%	6.02%	86.05
			∞	3.75%	5.43%	6.65%	63.12
		1.67	0.95	7.09%	8.2%	9.28%	63.59
			∞	7.23%	9.1%	10.41%	52.4
	0.5	1	0.95	—	—	—	—
			∞	4.13% *	6.93% *	8.96% *	104.38
		1.33	0.95	8.13%	9.13%	10.46%	44.32
			∞	6.86%	8.74%	10.79%	58.86
		1.67	0.95	3.18%	5.47%	8.23%	73.9
			∞	3.2%	7.28%	9.94%	49.22
ftv170	0.2	1	0.95	—	—	—	—
			∞	10.58% *	13.92% *	20.57% *	145.3
		1.33	0.95	6.49%	13.01%	17.4%	98.37
			∞	4.82%	8.83%	10.22%	96.85
		1.67	0.95	5.8%	8.49%	11.93%	79.69
			∞	6.06%	9.01%	11.82%	53.41
	0.35	1	0.95	—	—	—	—
			∞	7.37% *	9.53% *	15.21% *	108.68
		1.33	0.95	—	—	—	—
			∞	4.61%	6.23%	8.46%	112.17
		1.67	0.95	5.84%	8.2%	10.82%	80.95
			∞	5.19%	7.45%	10.85%	87.56
	0.5	1	0.95	—	—	—	—
			∞	7.43% *	8.42% *	10.67% *	119.3
		1.33	0.95	—	—	—	—
			∞	7.05%	9.23%	12.4%	70.5
		1.67	0.95	4.68%	7.84%	10.45%	82.34
			∞	5.74%	8.23%	11.14%	94.64

AFI as the constructive heuristic

Table B.8: Computational results, for $\omega = 1/3$ and $MaxIt = 5000$, using the solution obtained through the AFI heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	8.53
		1.33	0.95	4.4% *	4.41% *	4.45% *	9.49
			∞	5.24% *	5.52% *	5.74% *	7.51
		1.67	0.95	0.0%	0.64%	1.99%	12.82
			∞	1.99%	1.99%	1.99%	8.56
	0.35	1	0.95	0.0%	0.27%	0.42%	9.31
			∞	0.0%	0.33%	0.78%	10.84
		1.33	0.95	0.0%	0.0%	0.0%	12.16
			∞	0.0%	0.0%	0.0%	7.92
		1.67	0.95	0.0%	1.07%	2.46%	7.86
			∞	0.0%	0.98%	2.46%	6.37
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.55%	2.74%	5.89
		1.33	0.95	0.0%	0.66%	1.34%	11.8
			∞	0.29%	0.29%	0.29%	5.88
		1.67	0.95	0.0%	1.24%	2.74%	7.37
			∞	0.91%	1.53%	2.74%	6.97
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.74% *	2.23% *	14.32
		1.33	0.95	1.03% *	2.05% *	3.84% *	20.45
			∞	1.04% *	1.62% *	3.84% *	19.76
		1.67	0.95	0.39% *	0.53% *	0.59% *	15.39
			∞	0.33% *	0.38% *	0.46% *	18.06
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.13% *	4.13% *	18.32
		1.33	0.95	0.0%	0.0%	0.01%	15.25
			∞	0.0%	0.29%	0.97%	22.54
		1.67	0.95	0.0%	0.12%	0.26%	27.22
			∞	0.0%	0.11%	0.26%	21.51
	0.5	1	0.95	—	—	—	—
			∞	3.97%	4.05%	4.07%	15.11
		1.33	0.95	0.08%	2.21%	2.74%	20.5
			∞	0.0%	2.19%	2.74%	22.19

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.5	1.67	0.95	—	—	—	—
			∞	0.07%	0.74%	2.82%	14.74
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.25% *	8.52% *	9.09% *	107.26
		1.33	0.95	0.0%	0.0%	0.0%	30.56
			∞	0.0%	0.0%	0.0%	27.67
		1.67	0.95	0.0%	0.18%	0.46%	24.29
			∞	0.0%	0.18%	0.46%	28.42
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	7.14% *	8.88% *	59.17
		1.33	0.95	2.44% *	2.99% *	4.5% *	63.78
			∞	1.56% *	2.45% *	3.95% *	59.21
		1.67	0.95	0.0%	0.0%	0.0%	30.46
			∞	0.0%	0.12%	0.52%	26.23
	0.5	1	0.95	—	—	—	—
			∞	2.77% *	3.49% *	4.52% *	105.85
		1.33	0.95	1.76% *	2.21% *	4.02% *	69.99
			∞	1.78% *	2.75% *	4.8% *	85.55
		1.67	0.95	0.0%	0.03%	0.17%	32.91
			∞	0.0%	0.0%	0.0%	28.79
pr124	0.2	1	0.95	6.84% *	7.45% *	8.93% *	81.59
			∞	5.58% *	5.94% *	6.06% *	67.02
		1.33	0.95	9.44% *	9.44% *	9.44% *	38.71
			∞	9.23% *	9.6% *	10.86% *	47.99
		1.67	0.95	10.86% *	11.19% *	12.44% *	37.14
			∞	11.0% *	11.0% *	11.0% *	34.05
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	36.76
		1.33	0.95	—	—	—	—
			∞	0.0%	0.47%	0.88%	30.96
		1.67	0.95	7.25% *	8.11% *	8.78% *	47.91
			∞	5.92% *	6.54% *	6.75% *	32.89
	0.5	1	0.95	—	—	—	—
			∞	5.0% *	5.21% *	5.27% *	33.41
		1.33	0.95	1.98% *	2.33% *	2.87% *	53.15
			∞	4.33% *	4.94% *	5.24% *	39.89
		1.67	0.95	0.0%	0.63%	0.78%	31.62
			∞	0.0%	0.18%	0.88%	34.58
pr152	0.2	1	0.95	—	—	—	—
			∞	13.95% *	14.79% *	15.86% *	134.01
		1.33	0.95	13.21% *	14.28% *	15.17% *	209.25

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr152	0.2	1.33	∞	13.5% *	14.23% *	15.69% *	92.4
pr152	0.2	1.67	0.95	12.48% *	12.48% *	12.48% *	67.76
pr152	0.2	1.67	∞	12.36% *	12.66% *	13.07% *	60.12
pr152	0.35	1	0.95	—	—	—	—
pr152	0.35	1	∞	23.56% *	24.18% *	25.08% *	107.65
pr152	0.35	1.33	0.95	18.38% *	18.77% *	19.03% *	81.1
pr152	0.35	1.33	∞	20.51% *	20.54% *	20.6% *	88.94
pr152	0.35	1.67	0.95	—	—	—	—
pr152	0.35	1.67	∞	14.19% *	14.62% *	14.73% *	84.22
pr152	0.5	1	0.95	—	—	—	—
pr152	0.5	1	∞	9.25% *	9.25% *	9.25% *	87.52
pr152	0.5	1.33	0.95	9.04% *	9.18% *	9.33% *	103.44
pr152	0.5	1.33	∞	8.72% *	8.98% *	9.41% *	73.44
pr152	0.5	1.67	0.95	—	—	—	—
pr152	0.5	1.67	∞	22.24% *	23.01% *	24.0% *	76.1
rat195	0.2	1	0.95	—	—	—	—
rat195	0.2	1	∞	5.7% *	6.3% *	7.09% *	474.05
rat195	0.2	1.33	0.95	—	—	—	—
rat195	0.2	1.33	∞	3.49% *	4.29% *	5.15% *	307.67
rat195	0.2	1.67	0.95	3.72% *	4.69% *	5.92% *	165.5
rat195	0.2	1.67	∞	2.51% *	3.61% *	4.34% *	239.07
rat195	0.35	1	0.95	—	—	—	—
rat195	0.35	1	∞	2.79% *	4.16% *	5.63% *	321.04
rat195	0.35	1.33	0.95	2.63% *	3.92% *	4.98% *	215.03
rat195	0.35	1.33	∞	3.12% *	4.42% *	5.08% *	261.47
rat195	0.35	1.67	0.95	—	—	—	—
rat195	0.35	1.67	∞	2.69% *	3.87% *	5.2% *	246.57
rat195	0.5	1	0.95	—	—	—	—
rat195	0.5	1	∞	1.67% *	2.19% *	3.21% *	335.04
rat195	0.5	1.33	0.95	0.62% *	1.86% *	2.78% *	295.32
rat195	0.5	1.33	∞	1.46% *	2.87% *	4.08% *	264.28
rat195	0.5	1.67	0.95	1.6% *	2.66% *	3.96% *	278.85
rat195	0.5	1.67	∞	1.82%	3.21%	4.85%	179.14
pr226	0.2	1	0.95	—	—	—	—
pr226	0.2	1	∞	22.36% *	24.46% *	26.01% *	511.54
pr226	0.2	1.33	0.95	9.12% *	9.84% *	10.95% *	428.2
pr226	0.2	1.33	∞	9.43% *	9.66% *	10.24% *	599.85
pr226	0.2	1.67	0.95	6.71% *	6.76% *	6.91% *	250.16
pr226	0.2	1.67	∞	6.83% *	6.98% *	7.51% *	249.06
pr226	0.35	1	0.95	—	—	—	—
pr226	0.35	1	∞	14.45% *	14.53% *	14.67% *	316.27

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr226	0.35	1.33	0.95	11.32% *	11.4% *	11.54% *	231.12
			∞	10.49% *	10.55% *	10.63% *	243.66
		1.67	0.95	9.07% *	9.11% *	9.14% *	243.86
			∞	8.76% *	8.77% *	8.81% *	217.34
	0.5	1	0.95	—	—	—	—
			∞	18.91% *	19.1% *	19.5% *	601.77
		1.33	0.95	11.12% *	11.2% *	11.36% *	361.21
			∞	10.51% *	10.55% *	10.62% *	267.21
		1.67	0.95	10.96% *	11.01% *	11.13% *	214.54
			∞	10.52% *	10.56% *	10.61% *	224.76

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	—	—	—	—
			∞	11.09% *	14.86% *	19.75% *	26.89
		1.33	0.95	1.92%	4.15%	7.78%	14.25
			∞	3.42%	4.75%	7.35%	15.69
		1.67	0.95	0.0%	2.14%	9.07%	7.41
			∞	0.0%	3.98%	9.61%	8.05
	0.35	1	0.95	—	—	—	—
			∞	3.08%	5.44%	8.3%	19.56
		1.33	0.95	0.17%	4.42%	8.0%	22.38
			∞	2.64%	5.74%	10.54%	17.37
		1.67	0.95	0.0%	2.81%	6.65%	8.63
			∞	1.01%	2.35%	4.97%	8.0
	0.5	1	0.95	4.43%	6.62%	9.75%	11.45
			∞	3.35%	4.59%	7.1%	9.87
		1.33	0.95	3.98%	5.34%	5.78%	6.99
			∞	0.06%	2.37%	3.98%	8.72
		1.67	0.95	1.56%	2.72%	3.43%	5.44
			∞	1.03%	3.18%	5.17%	5.96
ftv64	0.2	1	0.95	2.76%	5.44%	8.87%	18.82
			∞	5.31%	8.34%	12.59%	20.27
		1.33	0.95	1.67%	4.28%	7.71%	15.71
			∞	3.07%	5.6%	10.89%	11.04
		1.67	0.95	—	—	—	—
			∞	2.28%	5.13%	7.5%	7.86
	0.35	1	0.95	7.69%	8.48%	9.91%	12.23
			∞	2.44%	6.53%	11.48%	23.85
		1.33	0.95	1.57%	3.48%	6.18%	12.27
			∞	1.57%	2.6%	3.68%	11.51

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.35	1.67	0.95	1.96%	3.55%	4.57%	6.93
			∞	2.28%	3.53%	4.57%	8.09
	0.5	1	0.95	—	—	—	—
			∞	0.27%	3.31%	10.76%	13.18
		1.33	0.95	0.49%	2.14%	3.15%	10.5
			∞	0.49%	1.41%	3.15%	8.56
		1.67	0.95	—	—	—	—
			∞	0.71%	2.1%	4.67%	7.82
ft70	0.2	1	0.95	3.57% *	4.67% *	5.47% *	44.93
			∞	2.6% *	3.96% *	5.16% *	44.38
		1.33	0.95	1.47%	2.32%	3.13%	35.23
			∞	0.93%	1.77%	2.52%	32.02
		1.67	0.95	1.29%	2.2%	2.7%	52.56
			∞	1.6%	2.22%	2.99%	32.38
	0.35	1	0.95	1.23%	2.36%	3.55%	31.79
			∞	1.24%	2.46%	3.96%	31.98
		1.33	0.95	1.39%	2.07%	2.93%	25.44
			∞	1.43%	1.89%	3.05%	32.9
		1.67	0.95	0.82%	2.21%	3.81%	24.24
			∞	0.97%	1.89%	3.21%	33.1
	0.5	1	0.95	1.44%	2.09%	2.59%	55.37
			∞	1.4%	2.14%	2.86%	37.53
		1.33	0.95	1.87%	2.52%	3.04%	28.77
			∞	0.79%	1.28%	1.75%	44.5
		1.67	0.95	1.28%	2.33%	3.05%	26.33
			∞	0.83%	2.57%	4.04%	29.22
kro124p	0.2	1	0.95	4.21% *	5.95% *	7.43% *	85.66
			∞	4.5% *	5.44% *	6.37% *	76.35
		1.33	0.95	2.8%	3.74%	5.42%	41.95
			∞	1.16%	2.77%	4.12%	46.17
		1.67	0.95	0.36%	0.72%	1.53%	41.58
			∞	0.03%	0.3%	0.46%	46.67
	0.35	1	0.95	—	—	—	—
			∞	7.01% *	8.28% *	10.3% *	134.12
		1.33	0.95	1.15%	1.5%	2.01%	42.52
			∞	1.08%	1.54%	2.34%	64.03
		1.67	0.95	2.24%	2.77%	4.12%	33.97
			∞	0.15%	2.09%	3.08%	39.04
	0.5	1	0.95	5.1% *	6.18% *	8.14% *	81.1
			∞	1.84% *	3.49% *	5.19% *	89.42
		1.33	0.95	2.2%	2.46%	3.0%	72.74

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.5	1.33	∞	2.27%	2.72%	3.65%	67.8
		1.67	0.95	2.24%	2.83%	3.56%	39.58
			∞	2.31%	2.9%	3.67%	45.45
ftv170	0.2	1	0.95	—	—	—	—
			∞	5.07% *	9.42% *	14.81% *	136.01
		1.33	0.95	3.99%	5.45%	6.96%	101.75
			∞	3.52%	5.32%	7.11%	84.68
		1.67	0.95	3.05%	5.54%	7.51%	86.12
			∞	3.84%	5.63%	9.61%	93.2
	0.35	1	0.95	12.22% *	13.54% *	14.55% *	174.94
			∞	9.77% *	11.57% *	12.6% *	165.67
		1.33	0.95	4.17%	6.74%	9.58%	68.02
			∞	1.92%	4.05%	8.38%	119.2
		1.67	0.95	4.1%	5.36%	6.5%	80.07
			∞	4.17%	6.35%	8.13%	57.1
	0.5	1	0.95	6.31% *	9.82% *	13.06% *	232.52
			∞	9.17% *	10.86% *	15.58% *	166.85
		1.33	0.95	7.23%	9.67%	13.3%	139.42
			∞	5.46%	8.64%	11.64%	90.16
		1.67	0.95	1.02%	2.88%	3.77%	63.49
			∞	2.4%	3.85%	4.94%	88.97

ARI as the constructive heuristic

Table B.9: Computational results, for $\omega = 1/3$ and $MaxIt = 5000$, using the solution obtained through the ARI heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	23.26
		1.33	0.95	4.27% *	4.48% *	4.96% *	15.09
			∞	5.38% *	5.64% *	5.94% *	16.4
		1.67	0.95	0.0%	0.41%	0.83%	24.54
			∞	1.99%	2.25%	3.26%	19.66
	0.35	1	0.95	0.14%	0.52%	1.17%	15.92
			∞	0.0%	0.11%	0.33%	20.55
		1.33	0.95	0.0%	0.0%	0.0%	26.24
			∞	0.0%	1.7%	2.36%	14.01

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.35	1.67	0.95	0.0%	0.73%	3.63%	13.27
			∞	0.0%	0.79%	2.36%	15.98
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.22%	1.08%	16.47
		1.33	0.95	0.0%	0.84%	1.44%	21.61
			∞	0.29%	0.45%	1.11%	14.34
		1.67	0.95	0.0%	0.74%	1.33%	21.48
			∞	0.0%	0.37%	1.33%	17.14
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.8% *	2.16% *	35.14
		1.33	0.95	1.12% *	1.56% *	2.05% *	38.27
			∞	1.04% *	1.68% *	3.25% *	38.14
		1.67	0.95	0.39% *	0.74% *	1.27% *	39.25
			∞	0.43% *	0.51% *	0.61% *	40.95
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.21% *	4.52% *	38.38
		1.33	0.95	0.0%	0.11%	0.21%	29.25
			∞	0.0%	0.04%	0.21%	23.15
		1.67	0.95	0.0%	0.79%	1.08%	31.95
			∞	0.07%	0.64%	0.95%	33.68
	0.5	1	0.95	—	—	—	—
			∞	0.0%	1.65%	3.97%	38.79
		1.33	0.95	0.0%	0.62%	2.8%	41.25
			∞	0.0%	1.4%	2.74%	35.02
		1.67	0.95	0.55%	0.65%	0.82%	39.46
			∞	0.0%	0.36%	0.95%	27.7
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.85% *	9.78% *	11.56% *	196.66
		1.33	0.95	0.0%	0.01%	0.04%	41.37
			∞	0.0%	0.01%	0.04%	56.76
		1.67	0.95	0.0%	0.02%	0.09%	59.72
			∞	0.0%	0.01%	0.04%	51.33
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.51% *	6.88% *	158.77
		1.33	0.95	2.44% *	2.8% *	3.51% *	92.51
			∞	1.56% *	2.5% *	4.24% *	126.76
		1.67	0.95	0.0%	0.05%	0.09%	74.46
			∞	0.0%	0.01%	0.04%	52.96
	0.5	1	0.95	—	—	—	—
			∞	2.62% *	2.92% *	3.3% *	204.61
		1.33	0.95	1.76% *	1.95% *	2.72% *	134.91

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kroA100	0.5	1.33	∞	1.78% *	1.79% *	1.81% *	110.47
		1.67	0.95	0.0%	0.0%	0.0%	62.4
			∞	0.0%	0.12%	0.61%	60.26
pr124	0.2	1	0.95	—	—	—	—
			∞	5.04% *	5.66% *	6.54% *	128.92
		1.33	0.95	9.37% *	10.05% *	11.0% *	67.59
			∞	9.3% *	9.68% *	11.17% *	65.43
		1.67	0.95	10.86% *	11.77% *	12.51% *	85.55
			∞	11.0% *	11.1% *	11.53% *	78.15
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	9.38% *	13.28% *	65.26
		1.33	0.95	—	—	—	—
			∞	0.0%	0.28%	0.78%	61.27
		1.67	0.95	6.69% *	7.59% *	8.78% *	81.9
			∞	5.92% *	7.02% *	8.37% *	75.26
	0.5	1	0.95	—	—	—	—
			∞	4.32% *	5.56% *	7.08% *	78.89
		1.33	0.95	2.87% *	3.61% *	4.42% *	74.28
			∞	4.33% *	4.64% *	5.9% *	66.12
		1.67	0.95	0.0%	0.26%	0.78%	72.21
			∞	0.0%	0.41%	1.44%	84.09
pr152	0.2	1	0.95	—	—	—	—
			∞	14.02% *	14.79% *	15.99% *	291.31
		1.33	0.95	13.21% *	13.21% *	13.21% *	371.82
			∞	13.15% *	13.87% *	15.22% *	229.49
		1.67	0.95	12.57% *	13.13% *	14.15% *	107.32
			∞	12.36% *	13.1% *	15.09% *	120.19
	0.35	1	0.95	—	—	—	—
			∞	23.74% *	24.27% *	24.92% *	148.8
		1.33	0.95	18.71% *	19.26% *	19.46% *	145.08
			∞	19.85% *	20.61% *	21.21% *	112.64
		1.67	0.95	—	—	—	—
			∞	13.89% *	14.22% *	14.52% *	136.47
	0.5	1	0.95	—	—	—	—
			∞	8.65% *	9.09% *	9.64% *	147.32
		1.33	0.95	8.64% *	9.65% *	10.19% *	100.2
			∞	8.43% *	9.17% *	9.63% *	130.98
		1.67	0.95	—	—	—	—
			∞	22.24% *	22.62% *	23.26% *	121.8
rat195	0.2	1	0.95	—	—	—	—
			∞	5.38% *	6.3% *	8.07% *	805.57

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
rat195	0.2	1.33	0.95	—	—	—	—
			∞	2.64% *	3.65% *	4.83% *	384.77
		1.67	0.95	2.55% *	4.23% *	6.54% *	438.46
			∞	2.64% *	3.66% *	4.43% *	296.02
	0.35	1	0.95	—	—	—	—
			∞	1.68% *	3.34% *	4.21% *	516.25
		1.33	0.95	2.71% *	3.31% *	4.63% *	360.05
			∞	2.76% *	3.51% *	4.28% *	438.76
		1.67	0.95	—	—	—	—
			∞	3.18% *	3.93% *	4.66% *	301.19
	0.5	1	0.95	—	—	—	—
			∞	1.8% *	2.98% *	4.49% *	607.38
		1.33	0.95	2.56% *	3.45% *	4.63% *	385.43
			∞	1.42% *	3.84% *	5.46% *	469.13
		1.67	0.95	2.67% *	3.4% *	3.92% *	423.09
			∞	2.09%	2.5%	3.29%	473.49
pr226	0.2	1	0.95	28.79% *	28.79% *	28.79% *	797.45
			∞	22.31% *	22.86% *	23.29% *	1029.84
		1.33	0.95	9.49% *	11.2% *	12.81% *	426.8
			∞	9.55% *	11.27% *	12.86% *	493.35
		1.67	0.95	6.72% *	6.77% *	6.93% *	370.65
			∞	6.84% *	6.88% *	7.04% *	510.86
	0.35	1	0.95	17.66% *	17.69% *	17.71% *	680.79
			∞	14.45% *	16.37% *	20.12% *	725.06
		1.33	0.95	11.32% *	11.63% *	12.4% *	429.53
			∞	10.47% *	11.24% *	14.02% *	374.75
		1.67	0.95	9.75% *	9.75% *	9.75% *	402.66
			∞	8.75% *	9.54% *	11.84% *	426.1
	0.5	1	0.95	—	—	—	—
			∞	18.96% *	19.26% *	19.54% *	895.39
		1.33	0.95	10.46% *	10.77% *	11.3% *	723.91
			∞	10.53% *	10.96% *	11.36% *	411.6
		1.67	0.95	10.97% *	11.05% *	11.36% *	495.28
			∞	10.52% *	10.53% *	10.56% *	477.99

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	10.56% *	14.0% *	20.05% *	51.17
			∞	11.34% *	13.61% *	16.22% *	48.27
		1.33	0.95	1.23%	4.94%	7.52%	26.16
			∞	1.14%	3.52%	7.24%	23.35

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1.67	0.95	0.01%	1.12%	2.45%	12.36
			∞	0.0%	2.09%	7.11%	14.91
	0.35	1	0.95	1.27%	3.91%	6.08%	32.83
			∞	3.98%	7.74%	12.57%	29.63
		1.33	0.95	3.44%	7.81%	10.64%	15.23
			∞	1.1%	6.05%	10.53%	24.92
		1.67	0.95	2.46%	4.65%	9.82%	16.73
			∞	4.61%	6.91%	10.34%	19.28
	0.5	1	0.95	9.98%	9.98%	9.98%	51.37
			∞	0.0%	5.36%	7.6%	11.16
		1.33	0.95	0.06%	5.83%	9.66%	14.97
			∞	0.0%	2.91%	6.57%	12.18
		1.67	0.95	0.0%	3.29%	7.33%	16.69
			∞	0.0%	2.2%	3.98%	14.33
ftv64	0.2	1	0.95	5.47%	7.66%	9.82%	35.74
			∞	3.98%	6.48%	9.13%	31.05
		1.33	0.95	1.35%	5.07%	8.68%	24.25
			∞	3.45%	5.24%	7.92%	23.76
		1.67	0.95	1.14%	4.39%	9.03%	30.28
			∞	4.79%	5.69%	8.48%	16.47
	0.35	1	0.95	0.05%	2.33%	5.09%	25.68
			∞	0.98%	6.02%	10.13%	33.79
		1.33	0.95	2.87%	3.02%	3.3%	27.2
			∞	2.82%	4.36%	6.72%	17.1
		1.67	0.95	2.28%	3.74%	5.98%	14.24
			∞	3.26%	4.56%	7.23%	13.07
	0.5	1	0.95	4.22%	5.0%	5.78%	37.3
			∞	0.38%	3.05%	5.89%	21.43
		1.33	0.95	2.06%	4.07%	5.97%	15.95
			∞	0.98%	3.76%	7.0%	19.23
		1.67	0.95	2.88%	4.04%	4.72%	13.18
			∞	2.12%	3.63%	4.89%	11.91
ft70	0.2	1	0.95	3.89% *	3.89% *	3.89% *	57.97
			∞	3.21% *	4.23% *	5.06% *	102.92
		1.33	0.95	2.03%	2.67%	3.57%	125.4
			∞	1.95%	2.47%	3.65%	63.69
		1.67	0.95	1.52%	2.56%	4.17%	61.86
			∞	0.85%	1.6%	2.66%	50.64
	0.35	1	0.95	1.63%	2.3%	2.92%	75.09
			∞	1.34%	2.09%	2.82%	89.08
		1.33	0.95	1.24%	1.81%	2.34%	73.53

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft70	0.35	1.33	∞	1.06%	2.34%	3.48%	57.29
		1.67	0.95	1.71%	2.51%	2.9%	45.9
			∞	1.08%	2.22%	3.46%	48.58
	0.5	1	0.95	1.85%	1.94%	2.04%	62.57
			∞	0.85%	2.14%	3.73%	65.91
		1.33	0.95	2.23%	2.6%	2.96%	54.92
			∞	1.03%	2.25%	3.33%	41.85
		1.67	0.95	1.03%	1.52%	2.16%	65.29
			∞	1.75%	2.25%	3.08%	44.12
kro124p	0.2	1	0.95	6.68% *	6.68% *	6.68% *	162.11
			∞	5.27% *	6.8% *	11.66% *	159.43
		1.33	0.95	0.03%	2.04%	4.08%	83.89
			∞	0.24%	2.34%	4.63%	102.32
		1.67	0.95	0.03%	1.7%	3.87%	98.25
			∞	0.23%	2.42%	5.22%	84.79
	0.35	1	0.95	—	—	—	—
			∞	7.07% *	9.46% *	10.97% *	180.77
		1.33	0.95	0.16%	1.86%	2.93%	88.5
			∞	0.16%	0.98%	2.08%	91.22
		1.67	0.95	0.26%	0.53%	1.08%	64.1
			∞	0.03%	1.56%	4.47%	80.42
	0.5	1	0.95	2.0% *	4.8% *	7.5% *	179.39
			∞	2.48% *	5.14% *	10.39% *	154.43
		1.33	0.95	0.62%	1.73%	3.43%	136.19
			∞	1.33%	2.12%	3.38%	110.08
		1.67	0.95	0.03%	1.14%	1.79%	106.89
			∞	0.33%	1.95%	3.69%	121.76
ftv170	0.2	1	0.95	—	—	—	—
			∞	8.28% *	11.74% *	13.93% *	257.09
		1.33	0.95	2.68%	5.18%	7.51%	166.01
			∞	5.22%	7.29%	9.03%	142.09
		1.67	0.95	4.02%	5.74%	7.03%	147.06
			∞	1.2%	4.42%	7.47%	141.6
	0.35	1	0.95	8.32% *	11.07% *	14.33% *	172.67
			∞	9.48% *	13.68% *	16.19% *	244.53
		1.33	0.95	1.67%	6.21%	9.84%	142.13
			∞	3.38%	7.19%	8.86%	220.65
		1.67	0.95	3.41%	6.64%	9.98%	126.72
			∞	4.72%	6.26%	9.07%	138.92
	0.5	1	0.95	16.43% *	16.43% *	16.43% *	967.64
			∞	9.06% *	11.29% *	14.89% *	244.25

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv170	0.5	1.33	0.95	5.35%	7.74%	9.83%	128.79
			∞	9.07%	10.07%	11.1%	190.91
		1.67	0.95	2.61%	6.79%	8.86%	189.01
			∞	2.47%	7.13%	11.07%	115.25

B.2 Results for $\omega = 1/2$

B.2.1 Results for a maximum of 1000 iterations

ANN as the constructive heuristic

Table B.10: Computational results, for $\omega = 1/2$ and $MaxIt = 1000$, using the solution obtained through the ANN heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	——	——	——	——
			∞	4.49% *	4.49% *	4.49% *	3.57
		1.33	0.95	4.4% *	4.41% *	4.45% *	3.74
			∞	5.43% *	5.73% *	6.02% *	1.97
		1.67	0.95	——	——	——	——
			∞	0.0%	0.89%	1.99%	3.34
	0.35	1	0.95	0.23%	0.76%	1.17%	2.78
			∞	0.33%	0.94%	1.32%	1.85
		1.33	0.95	0.0%	0.0%	0.0%	3.17
			∞	0.0%	1.2%	2.36%	2.49
		1.67	0.95	0.0%	0.17%	0.52%	2.12
			∞	0.0%	0.15%	0.46%	2.61
	0.5	1	0.95	——	——	——	——
			∞	0.0%	0.12%	0.36%	1.88
		1.33	0.95	——	——	——	——
			∞	0.29%	0.69%	1.11%	1.63
		1.67	0.95	0.52%	1.53%	2.74%	2.24
			∞	0.0%	1.04%	3.12%	2.22
pr76	0.2	1	0.95	——	——	——	——
			∞	1.38% *	1.66% *	2.13% *	6.08
		1.33	0.95	1.12% *	1.4% *	1.54% *	11.24
			∞	1.13% *	2.12% *	3.62% *	7.54
		1.67	0.95	0.52% *	0.75% *	1.16% *	5.43
			∞	0.52% *	0.85% *	1.2% *	4.61

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.51% *	5.27% *	6.84
		1.33	0.95	0.0%	0.03%	0.08%	3.9
			∞	0.0%	0.03%	0.08%	3.95
		1.67	0.95	0.07%	0.63%	0.95%	4.13
			∞	0.83%	0.93%	1.08%	4.44
	0.5	1	0.95	—	—	—	—
			∞	0.01%	0.11%	0.24%	6.31
		1.33	0.95	0.0%	0.03%	0.09%	4.82
			∞	0.0%	0.4%	1.12%	6.02
		1.67	0.95	0.55%	0.66%	0.87%	3.7
			∞	0.82%	0.84%	0.9%	6.43
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.85% *	9.76% *	11.01% *	32.9
		1.33	0.95	—	—	—	—
			∞	0.0%	0.01%	0.04%	8.36
		1.67	0.95	—	—	—	—
			∞	0.0%	0.03%	0.09%	8.36
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.49% *	6.61% *	20.64
		1.33	0.95	—	—	—	—
			∞	1.56% *	2.08% *	3.11% *	25.46
		1.67	0.95	0.0%	0.0%	0.0%	9.65
			∞	0.09%	0.09%	0.09%	7.99
	0.5	1	0.95	—	—	—	—
			∞	2.78% *	3.28% *	4.26% *	25.72
		1.33	0.95	1.76% *	2.12% *	2.8% *	19.21
			∞	1.78% *	1.79% *	1.81% *	19.07
		1.67	0.95	0.04%	0.04%	0.04%	11.01
			∞	0.0%	0.3%	0.46%	7.56
pr124	0.2	1	0.95	—	—	—	—
			∞	6.06% *	6.63% *	7.36% *	22.86
		1.33	0.95	9.44% *	9.96% *	11.0% *	13.34
			∞	9.23% *	9.46% *	9.83% *	9.66
		1.67	0.95	10.93% *	10.93% *	10.93% *	7.14
			∞	11.0% *	11.0% *	11.0% *	7.24
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	11.65% *	13.28% *	9.13
		1.33	0.95	—	—	—	—
			∞	0.6%	0.69%	0.88%	13.04
		1.67	0.95	—	—	—	—

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr124	0.35	1.67	∞	6.56% *	6.56% *	6.56% *	11.32
	0.5	1	0.95	—	—	—	—
			∞	4.81% *	5.54% *	6.54% *	20.43
		1.33	0.95	—	—	—	—
			∞	4.33% *	4.94% *	5.24% *	12.14
		1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	6.7
pr152	0.2	1	0.95	14.06% *	14.56% *	14.82% *	51.93
			∞	13.98% *	14.48% *	15.23% *	64.44
		1.33	0.95	13.55% *	13.96% *	14.63% *	36.01
			∞	13.62% *	14.17% *	15.18% *	31.59
		1.67	0.95	13.07% *	13.07% *	13.07% *	27.22
			∞	12.95% *	13.71% *	14.89% *	18.41
	0.35	1	0.95	26.55% *	26.86% *	27.38% *	26.79
			∞	24.51% *	25.51% *	27.08% *	42.5
		1.33	0.95	18.71% *	19.41% *	19.8% *	22.6
			∞	20.94% *	21.14% *	21.29% *	18.32
		1.67	0.95	14.64% *	14.79% *	14.93% *	19.4
			∞	14.8% *	14.97% *	15.05% *	16.18
	0.5	1	0.95	—	—	—	—
			∞	9.56% *	9.56% *	9.56% *	22.71
		1.33	0.95	8.64% *	10.33% *	11.18% *	32.56
			∞	9.34% *	10.09% *	11.08% *	15.62
		1.67	0.95	—	—	—	—
			∞	22.24% *	23.07% *	23.61% *	27.55
rat195	0.2	1	0.95	—	—	—	—
			∞	4.3% *	5.65% *	7.58% *	175.33
		1.33	0.95	—	—	—	—
			∞	3.8% *	3.92% *	4.03% *	88.97
		1.67	0.95	3.85% *	3.96% *	4.03% *	52.67
			∞	3.49% *	3.7% *	3.94% *	35.86
	0.35	1	0.95	—	—	—	—
			∞	3.86% *	4.55% *	5.59% *	99.01
		1.33	0.95	2.23% *	2.6% *	2.8% *	52.93
			∞	2.54% *	3.03% *	3.92% *	63.19
		1.67	0.95	—	—	—	—
			∞	2.69% *	3.02% *	3.36% *	50.37
	0.5	1	0.95	—	—	—	—
			∞	2.64% *	3.06% *	3.56% *	100.79
		1.33	0.95	2.38% *	2.92% *	3.7% *	83.8
			∞	2.71% *	2.9% *	3.2% *	91.81

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
rat195	0.5	1.67	0.95	2.85% *	3.22% *	3.88% *	68.49
			∞	2.22%	2.4%	2.62%	31.86
pr226	0.2	1	0.95	22.47% *	22.53% *	22.6% *	136.55
			∞	22.82% *	23.16% *	23.53% *	101.22
		1.33	0.95	———	———	———	———
			∞	11.58% *	12.03% *	12.6% *	50.92
		1.67	0.95	6.89% *	7.26% *	7.5% *	47.42
			∞	6.92% *	7.39% *	7.64% *	76.98
	0.35	1	0.95	18.11% *	18.92% *	20.54% *	98.45
			∞	18.23% *	19.36% *	21.59% *	93.09
		1.33	0.95	———	———	———	———
			∞	10.87% *	11.05% *	11.36% *	59.99
		1.67	0.95	———	———	———	———
			∞	8.83% *	9.07% *	9.41% *	82.48
	0.5	1	0.95	14.3% *	16.8% *	18.05% *	273.81
			∞	17.9% *	19.16% *	19.8% *	100.8
		1.33	0.95	11.26% *	13.68% *	16.0% *	71.2
			∞	10.63% *	11.99% *	14.18% *	69.96
		1.67	0.95	———	———	———	———
			∞	10.59% *	10.63% *	10.69% *	56.86

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	15.93% *	19.96% *	25.88% *	6.97
			∞	12.33% *	16.7% *	20.69% *	6.82
		1.33	0.95	4.47%	6.71%	10.76%	3.81
			∞	3.3%	4.27%	5.03%	5.6
		1.67	0.95	3.13%	4.98%	6.44%	2.25
			∞	0.0%	2.62%	5.33%	3.0
	0.35	1	0.95	2.16%	5.32%	7.48%	5.2
			∞	7.27%	8.14%	9.78%	5.38
		1.33	0.95	3.37%	3.66%	4.15%	4.86
			∞	0.48%	4.22%	7.07%	5.43
		1.67	0.95	1.64%	3.0%	4.88%	4.39
			∞	1.01%	4.69%	6.53%	1.67
	0.5	1	0.95	3.07%	5.64%	9.89%	5.47
			∞	0.0%	0.8%	2.35%	3.22
		1.33	0.95	4.87%	7.63%	10.76%	3.0
			∞	0.0%	1.77%	3.75%	2.47
		1.67	0.95	0.06%	2.78%	5.66%	1.39
			∞	1.03%	6.62%	12.24%	1.97

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.2	1	0.95	—	—	—	—
			∞	4.78%	5.98%	7.65%	6.61
		1.33	0.95	—	—	—	—
			∞	3.77%	6.47%	11.7%	4.08
		1.67	0.95	5.6%	7.07%	8.16%	3.4
			∞	2.77%	6.8%	9.57%	2.96
	0.35	1	0.95	0.05%	3.05%	9.05%	4.31
			∞	0.05%	0.65%	1.84%	5.16
		1.33	0.95	2.87%	3.47%	4.66%	2.75
			∞	1.68%	3.34%	4.28%	2.25
		1.67	0.95	2.61%	3.92%	5.22%	2.05
			∞	3.1%	4.21%	5.87%	2.28
	0.5	1	0.95	6.32%	8.0%	10.32%	4.55
			∞	1.19%	2.5%	3.51%	4.7
		1.33	0.95	4.29%	5.05%	6.08%	2.08
			∞	4.29%	4.87%	5.16%	2.12
		1.67	0.95	0.87%	2.95%	4.89%	3.1
			∞	1.85%	4.92%	9.45%	2.38
ft70	0.2	1	0.95	5.15% *	6.07% *	7.05% *	9.86
			∞	4.8% *	5.95% *	7.09% *	12.57
		1.33	0.95	3.45%	4.08%	4.65%	9.92
			∞	3.81%	4.0%	4.12%	11.58
		1.67	0.95	4.49%	5.77%	6.68%	6.29
			∞	3.67%	4.0%	4.4%	6.5
	0.35	1	0.95	3.61%	4.24%	4.86%	9.47
			∞	2.04%	3.06%	4.04%	9.64
		1.33	0.95	2.84%	3.3%	4.22%	9.07
			∞	3.16%	3.91%	4.97%	7.25
		1.67	0.95	—	—	—	—
			∞	3.22%	3.91%	4.38%	6.84
	0.5	1	0.95	—	—	—	—
			∞	3.03%	3.21%	3.4%	7.65
		1.33	0.95	—	—	—	—
			∞	2.86%	3.14%	3.57%	5.77
		1.67	0.95	—	—	—	—
			∞	2.59%	2.89%	3.2%	7.59
kro124p	0.2	1	0.95	—	—	—	—
			∞	10.19% *	11.58% *	13.01% *	23.84
		1.33	0.95	0.9%	4.19%	6.96%	18.53
			∞	4.02%	7.26%	9.68%	13.9
		1.67	0.95	9.48%	11.66%	14.91%	11.15

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kro124p	0.2	1.67	∞	6.31%	9.29%	11.5%	13.64
	0.35	1	0.95	—	—	—	—
			∞	7.58% *	9.82% *	12.13% *	14.21
		1.33	0.95	4.75%	6.47%	8.4%	14.39
			∞	5.07%	5.93%	7.62%	12.14
		1.67	0.95	9.67%	11.73%	12.78%	12.34
			∞	6.2%	9.25%	11.86%	12.59
	0.5	1	0.95	—	—	—	—
			∞	10.37% *	11.3% *	12.97% *	18.99
		1.33	0.95	7.55%	8.46%	9.81%	16.5
			∞	10.28%	10.75%	11.59%	12.89
		1.67	0.95	4.74%	6.21%	8.41%	25.18
			∞	4.76%	6.82%	9.91%	18.09
ftv170	0.2	1	0.95	—	—	—	—
			∞	9.48% *	13.88% *	17.51% *	44.51
		1.33	0.95	8.41%	9.37%	9.97%	25.51
			∞	8.48%	9.21%	10.33%	23.98
		1.67	0.95	5.18%	7.88%	9.64%	29.04
			∞	10.7%	12.59%	15.92%	21.19
	0.35	1	0.95	—	—	—	—
			∞	8.61% *	10.28% *	12.78% *	61.11
		1.33	0.95	—	—	—	—
			∞	6.57%	7.99%	10.42%	35.88
		1.67	0.95	7.01%	8.22%	9.8%	30.57
			∞	5.08%	7.89%	11.8%	24.01
	0.5	1	0.95	—	—	—	—
			∞	10.19% *	11.64% *	13.65% *	30.78
		1.33	0.95	—	—	—	—
			∞	6.04%	7.66%	9.15%	31.19
		1.67	0.95	8.35%	10.74%	15.21%	28.07
			∞	5.23%	7.38%	9.36%	27.94

AFI as the constructive heuristic

Table B.11: Computational results, for $\omega = 1/2$ and $MaxIt = 1000$, using the solution obtained through the AFI heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	4.73
		1.33	0.95	4.4% *	4.54% *	4.84% *	2.53
			∞	5.24% *	5.58% *	5.74% *	3.41
		1.67	0.95	1.99%	1.99%	1.99%	2.53
			∞	1.99%	2.05%	2.16%	2.61
	0.35	1	0.95	0.0%	0.21%	0.42%	3.68
			∞	0.0%	0.14%	0.42%	2.84
		1.33	0.95	0.0%	0.41%	1.24%	3.05
			∞	0.0%	0.83%	1.24%	2.67
		1.67	0.95	0.0%	0.15%	0.46%	3.38
			∞	0.51%	1.78%	2.46%	1.92
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	2.1
		1.33	0.95	1.34%	1.37%	1.39%	2.88
			∞	0.29%	0.42%	0.69%	1.66
		1.67	0.95	0.0%	0.89%	1.33%	3.43
			∞	1.33%	1.33%	1.33%	1.44
pr76	0.2	1	0.95	—	—	—	—
			∞	1.77% *	2.13% *	2.4% *	4.87
		1.33	0.95	1.05% *	1.76% *	3.1% *	7.18
			∞	1.06% *	2.0% *	3.84% *	6.49
		1.67	0.95	0.59% *	0.83% *	1.16% *	5.63
			∞	0.33% *	0.33% *	0.33% *	5.1
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.38% *	4.89% *	7.15
		1.33	0.95	0.0%	0.03%	0.08%	4.36
			∞	0.0%	0.14%	0.21%	4.47
		1.67	0.95	0.0%	0.09%	0.26%	7.06
			∞	0.0%	0.18%	0.26%	5.16
	0.5	1	0.95	—	—	—	—
			∞	0.0%	2.27%	4.07%	8.91
		1.33	0.95	2.74%	2.74%	2.74%	5.38
			∞	2.74%	2.74%	2.74%	6.68

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.5	1.67	0.95	—	—	—	—
			∞	0.0%	0.09%	0.26%	4.36
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.85% *	9.9% *	11.59% *	31.07
		1.33	0.95	0.0%	0.0%	0.0%	7.95
			∞	0.0%	0.01%	0.04%	6.65
		1.67	0.95	0.0%	0.15%	0.46%	8.58
			∞	0.0%	0.06%	0.17%	8.29
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.55% *	6.99% *	32.23
		1.33	0.95	2.56% *	3.51% *	4.71% *	15.37
			∞	3.15% *	3.6% *	4.06% *	8.78
		1.67	0.95	0.0%	0.03%	0.09%	10.11
			∞	0.0%	0.0%	0.0%	10.38
	0.5	1	0.95	—	—	—	—
			∞	2.52% *	3.12% *	4.29% *	40.49
		1.33	0.95	1.76% *	2.85% *	4.15% *	22.21
			∞	1.78% *	1.83% *	1.93% *	21.67
		1.67	0.95	0.0%	0.01%	0.04%	9.09
			∞	0.0%	0.27%	0.8%	9.67
pr124	0.2	1	0.95	7.32% *	7.86% *	8.69% *	23.09
			∞	6.06% *	6.06% *	6.06% *	14.14
		1.33	0.95	9.44% *	9.44% *	9.44% *	9.92
			∞	9.3% *	9.3% *	9.3% *	13.65
		1.67	0.95	10.86% *	11.09% *	11.47% *	12.57
			∞	11.0% *	11.0% *	11.0% *	12.37
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	11.82
		1.33	0.95	—	—	—	—
			∞	0.0%	0.49%	0.88%	9.7
		1.67	0.95	8.14% *	8.17% *	8.23% *	10.53
			∞	6.0% *	6.37% *	6.56% *	12.66
	0.5	1	0.95	—	—	—	—
			∞	4.81% *	5.12% *	5.27% *	18.15
		1.33	0.95	1.98% *	1.98% *	1.98% *	13.6
			∞	4.33% *	4.84% *	5.24% *	9.92
		1.67	0.95	0.0%	0.49%	0.88%	18.03
			∞	0.0%	0.0%	0.0%	9.48
pr152	0.2	1	0.95	—	—	—	—
			∞	14.02% *	15.16% *	16.87% *	41.15
		1.33	0.95	14.38% *	14.89% *	15.79% *	37.86

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr152	0.2	1.33	∞	14.07% *	14.27% *	14.38% *	26.61
		1.67	0.95	12.48% *	12.92% *	13.19% *	18.3
			∞	12.45% *	12.83% *	13.07% *	19.08
	0.35	1	0.95	—	—	—	—
			∞	24.35% *	24.63% *	25.08% *	22.93
		1.33	0.95	19.12% *	19.2% *	19.36% *	18.85
			∞	20.6% *	20.64% *	20.7% *	18.53
		1.67	0.95	—	—	—	—
			∞	14.19% *	14.66% *	15.05% *	21.26
	0.5	1	0.95	—	—	—	—
			∞	9.25% *	9.28% *	9.33% *	15.44
		1.33	0.95	9.33% *	9.62% *	10.21% *	20.91
			∞	9.03% *	9.19% *	9.32% *	23.99
		1.67	0.95	—	—	—	—
			∞	23.01% *	23.34% *	24.0% *	15.63
rat195	0.2	1	0.95	—	—	—	—
			∞	5.52% *	7.13% *	8.92% *	136.5
		1.33	0.95	—	—	—	—
			∞	3.13% *	3.55% *	3.76% *	112.69
		1.67	0.95	2.73% *	3.35% *	3.77% *	82.91
			∞	3.9% *	4.55% *	4.88% *	75.7
	0.35	1	0.95	—	—	—	—
			∞	2.13% *	2.9% *	3.37% *	80.68
		1.33	0.95	5.47% *	5.64% *	5.79% *	48.34
			∞	3.66% *	4.93% *	6.02% *	78.79
		1.67	0.95	—	—	—	—
			∞	3.14% *	4.65% *	5.7% *	71.61
	0.5	1	0.95	—	—	—	—
			∞	2.46% *	3.18% *	4.58% *	94.99
		1.33	0.95	2.16% *	3.1% *	3.66% *	79.09
			∞	3.51% *	4.85% *	6.75% *	73.02
		1.67	0.95	4.41% *	5.29% *	5.88% *	71.66
			∞	3.69%	4.42%	5.2%	59.01
pr226	0.2	1	0.95	—	—	—	—
			∞	27.08% *	27.66% *	28.7% *	135.45
		1.33	0.95	10.23% *	11.31% *	12.15% *	66.46
			∞	9.67% *	11.81% *	16.02% *	90.94
		1.67	0.95	6.79% *	6.83% *	6.87% *	65.58
			∞	6.91% *	7.06% *	7.18% *	55.73
	0.35	1	0.95	—	—	—	—
			∞	14.57% *	14.73% *	14.85% *	111.61

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr226	0.35	1.33	0.95	11.38% *	11.63% *	11.77% *	78.8
			∞	10.48% *	10.54% *	10.6% *	123.3
		1.67	0.95	9.17% *	9.34% *	9.61% *	111.29
			∞	8.83% *	8.85% *	8.87% *	81.7
	0.5	1	0.95	—	—	—	—
			∞	19.51% *	19.54% *	19.56% *	169.15
		1.33	0.95	11.3% *	11.38% *	11.47% *	63.29
			∞	10.58% *	10.6% *	10.61% *	90.67
		1.67	0.95	10.96% *	11.39% *	11.98% *	89.39
			∞	10.59% *	10.61% *	10.63% *	80.23

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	—	—	—	—
			∞	16.61% *	16.98% *	17.5% *	5.82
		1.33	0.95	2.36%	4.98%	9.45%	3.87
			∞	0.74%	2.8%	4.88%	6.24
		1.67	0.95	0.71%	3.79%	9.24%	3.41
			∞	0.0%	6.17%	9.84%	2.99
	0.35	1	0.95	—	—	—	—
			∞	6.18%	7.75%	8.89%	3.78
		1.33	0.95	2.55%	3.9%	5.56%	3.01
			∞	3.33%	3.51%	3.66%	5.02
		1.67	0.95	1.25%	4.47%	8.3%	1.99
			∞	0.0%	0.77%	1.38%	2.31
	0.5	1	0.95	3.98%	5.1%	7.33%	3.43
			∞	4.2%	4.83%	5.87%	4.04
		1.33	0.95	2.39%	3.95%	5.78%	1.88
			∞	0.06%	1.57%	3.42%	3.0
		1.67	0.95	1.56%	2.76%	3.43%	2.59
			∞	1.56%	2.26%	3.26%	1.83
ftv64	0.2	1	0.95	11.15%	11.9%	12.37%	5.68
			∞	6.85%	7.59%	8.71%	5.26
		1.33	0.95	3.02%	5.01%	7.12%	3.81
			∞	5.44%	6.06%	6.42%	3.2
		1.67	0.95	—	—	—	—
			∞	2.28%	3.1%	3.53%	6.44
	0.35	1	0.95	0.98%	3.95%	9.05%	4.53
			∞	5.74%	6.01%	6.39%	4.4
		1.33	0.95	1.57%	2.28%	3.41%	3.62
			∞	1.9%	3.83%	4.93%	2.29

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.35	1.67	0.95	1.96%	3.57%	5.44%	2.14
			∞	4.46%	5.13%	5.98%	2.9
	0.5	1	0.95	———	———	———	———
			∞	0.43%	1.26%	2.0%	2.62
		1.33	0.95	0.98%	1.94%	2.77%	4.01
			∞	0.81%	4.65%	8.31%	1.99
		1.67	0.95	———	———	———	———
			∞	1.14%	1.56%	2.06%	2.34
ft70	0.2	1	0.95	6.13% *	6.45% *	6.99% *	7.85
			∞	5.67% *	6.0% *	6.18% *	9.85
		1.33	0.95	3.36%	4.46%	5.2%	7.13
			∞	3.22%	3.49%	3.89%	5.31
		1.67	0.95	4.97%	5.55%	6.69%	8.47
			∞	3.21%	3.9%	4.58%	6.36
	0.35	1	0.95	2.61%	3.3%	4.59%	7.9
			∞	2.15%	3.05%	3.64%	9.1
		1.33	0.95	3.46%	3.67%	3.98%	5.69
			∞	3.23%	4.06%	5.03%	9.11
		1.67	0.95	2.81%	3.6%	4.55%	6.28
			∞	1.6%	2.88%	4.02%	6.71
	0.5	1	0.95	2.9%	3.62%	4.26%	9.18
			∞	2.35%	3.54%	4.73%	8.3
		1.33	0.95	2.44%	3.06%	3.58%	9.34
			∞	2.25%	3.2%	4.25%	8.26
		1.67	0.95	2.94%	4.01%	5.13%	7.1
			∞	2.91%	3.88%	4.78%	7.19
kro124p	0.2	1	0.95	5.86% *	7.42% *	9.09% *	29.12
			∞	5.5% *	7.41% *	9.62% *	26.38
		1.33	0.95	0.68%	2.14%	4.85%	10.66
			∞	0.72%	3.84%	6.01%	8.62
		1.67	0.95	0.7%	1.17%	1.79%	9.33
			∞	0.67%	1.71%	3.16%	12.75
	0.35	1	0.95	———	———	———	———
			∞	7.25% *	10.19% *	15.44% *	22.92
		1.33	0.95	2.14%	3.4%	4.4%	10.02
			∞	2.25%	2.67%	3.27%	10.87
		1.67	0.95	2.88%	4.19%	5.4%	5.31
			∞	2.38%	3.26%	4.03%	7.84
	0.5	1	0.95	5.45% *	7.41% *	8.5% *	16.43
			∞	3.26% *	5.15% *	7.21% *	27.09
		1.33	0.95	2.49%	4.28%	5.91%	15.71

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.5	1.33	∞	1.19%	3.59%	5.81%	13.7
		1.67	0.95	3.07%	3.44%	4.15%	14.23
			∞	2.88%	4.29%	5.48%	17.21
ftv170	0.2	1	0.95	———	———	———	———
			∞	7.59% *	9.6% *	10.76% *	47.75
		1.33	0.95	4.64%	6.93%	9.17%	24.2
			∞	9.97%	10.44%	10.7%	17.64
		1.67	0.95	6.49%	7.84%	10.01%	23.19
			∞	4.39%	6.21%	8.16%	26.58
	0.35	1	0.95	10.76% *	14.69% *	18.27% *	42.87
			∞	14.05% *	17.62% *	19.86% *	35.51
		1.33	0.95	5.19%	7.07%	8.49%	32.0
			∞	4.79%	6.76%	9.95%	32.33
		1.67	0.95	2.47%	5.13%	6.64%	22.01
			∞	4.36%	6.05%	7.51%	17.94
	0.5	1	0.95	6.78% *	8.63% *	10.45% *	44.75
			∞	5.39% *	7.81% *	12.09% *	59.36
		1.33	0.95	10.12%	12.62%	13.99%	21.76
			∞	5.97%	9.01%	11.35%	49.89
		1.67	0.95	1.63%	3.17%	4.43%	27.18
			∞	1.74%	2.67%	3.88%	29.93

ARI as the constructive heuristic

Table B.12: Computational results, for $\omega = 1/2$ and $MaxIt = 1000$, using the solution obtained through the ARI heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	———	———	———	———
			∞	4.49% *	4.81% *	5.45% *	6.32
		1.33	0.95	4.4% *	4.41% *	4.45% *	7.15
			∞	5.24% *	5.47% *	5.74% *	4.03
		1.67	0.95	0.0%	1.37%	3.29%	5.96
			∞	0.88%	2.04%	3.25%	3.96
	0.35	1	0.95	1.17%	1.17%	1.17%	3.62
			∞	0.33%	0.66%	1.25%	6.52
		1.33	0.95	0.0%	1.76%	2.93%	5.08
			∞	0.0%	1.76%	2.91%	3.64

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.35	1.67	0.95	0.0%	0.79%	2.36%	4.26
			∞	0.0%	0.38%	1.14%	4.46
	0.5	1	0.95	0.87%	1.54%	2.2%	6.31
			∞	0.0%	0.0%	0.0%	5.0
		1.33	0.95	1.07%	1.28%	1.44%	7.17
			∞	0.29%	0.69%	1.11%	3.23
		1.67	0.95	0.0%	0.34%	0.52%	4.42
			∞	0.0%	0.0%	0.0%	3.41
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.45% *	1.45% *	12.91
		1.33	0.95	1.05% *	1.05% *	1.05% *	20.77
			∞	1.06% *	1.26% *	1.62% *	10.4
		1.67	0.95	0.74% *	0.96% *	1.27% *	8.09
			∞	0.33% *	0.48% *	0.61% *	10.15
	0.35	1	0.95	8.03% *	8.03% *	8.03% *	18.75
			∞	4.13% *	4.13% *	4.13% *	14.81
		1.33	0.95	0.07%	0.16%	0.21%	7.38
			∞	0.0%	0.07%	0.21%	8.73
		1.67	0.95	0.07%	0.36%	0.95%	12.98
			∞	0.95%	2.14%	2.97%	10.93
	0.5	1	0.95	0.0%	1.59%	2.61%	15.25
			∞	0.07%	1.45%	2.74%	16.07
		1.33	0.95	0.0%	0.03%	0.07%	8.6
			∞	0.0%	0.87%	2.52%	12.14
		1.67	0.95	0.55%	0.56%	0.56%	10.74
			∞	0.82%	0.84%	0.86%	11.09
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.92% *	10.55% *	11.89% *	82.98
		1.33	0.95	0.0%	0.02%	0.04%	15.19
			∞	0.0%	0.15%	0.46%	13.75
		1.67	0.95	0.0%	0.0%	0.0%	15.32
			∞	0.0%	0.03%	0.09%	17.33
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.45% *	6.69% *	48.97
		1.33	0.95	2.44% *	3.71% *	5.88% *	35.9
			∞	1.95% *	2.08% *	2.21% *	35.93
		1.67	0.95	0.09%	0.11%	0.13%	22.34
			∞	0.0%	0.19%	0.45%	17.36
	0.5	1	0.95	—	—	—	—
			∞	2.78% *	3.38% *	3.94% *	41.2
		1.33	0.95	1.76% *	1.81% *	1.93% *	45.49

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kroA100	0.5	1.33	∞	1.78% *	1.78% *	1.79% *	39.95
		1.67	0.95	0.0%	0.0%	0.0%	25.61
			∞	0.0%	0.27%	0.8%	19.13
pr124	0.2	1	0.95	—	—	—	—
			∞	5.04% *	7.01% *	8.7% *	32.08
		1.33	0.95	9.44% *	9.62% *	9.97% *	19.07
			∞	9.3% *	9.3% *	9.3% *	20.84
		1.67	0.95	11.47% *	11.47% *	11.47% *	29.97
			∞	11.0% *	11.13% *	11.41% *	27.2
	0.35	1	0.95	8.73% *	8.73% *	8.73% *	16.22
			∞	8.4% *	8.4% *	8.4% *	24.69
		1.33	0.95	0.88%	0.88%	0.88%	22.42
			∞	0.0%	0.0%	0.0%	28.21
		1.67	0.95	6.69% *	7.78% *	8.52% *	25.93
			∞	5.92% *	6.43% *	7.44% *	22.07
	0.5	1	0.95	—	—	—	—
			∞	4.81% *	5.12% *	5.27% *	39.33
		1.33	0.95	2.87% *	3.3% *	3.51% *	25.35
			∞	4.33% *	5.03% *	6.45% *	26.55
		1.67	0.95	0.0%	0.2%	0.6%	18.88
			∞	0.07%	0.7%	1.44%	17.23
pr152	0.2	1	0.95	13.48% *	14.21% *	14.95% *	96.62
			∞	14.35% *	15.47% *	16.43% *	127.05
		1.33	0.95	13.7% *	13.88% *	14.06% *	56.65
			∞	13.19% *	13.78% *	14.43% *	94.94
		1.67	0.95	12.57% *	13.65% *	15.19% *	48.38
			∞	12.36% *	12.74% *	13.39% *	69.64
	0.35	1	0.95	24.23% *	24.23% *	24.23% *	40.8
			∞	24.03% *	24.63% *	25.39% *	63.05
		1.33	0.95	18.38% *	18.59% *	19.03% *	29.95
			∞	20.51% *	20.85% *	21.19% *	70.8
		1.67	0.95	14.72% *	14.72% *	14.72% *	26.24
			∞	14.1% *	14.66% *	15.05% *	34.65
	0.5	1	0.95	—	—	—	—
			∞	8.65% *	12.99% *	15.41% *	68.68
		1.33	0.95	8.44% *	8.8% *	9.33% *	36.32
			∞	9.27% *	10.04% *	11.31% *	36.73
		1.67	0.95	—	—	—	—
			∞	22.01% *	22.84% *	23.26% *	29.74
rat195	0.2	1	0.95	—	—	—	—
			∞	7.53% *	7.79% *	8.21% *	228.19

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
rat195	0.2	1.33	0.95	5.06% *	5.06% *	5.06% *	205.58
			∞	5.24% *	5.7% *	6.0% *	104.72
		1.67	0.95	4.75% *	5.69% *	6.45% *	140.44
			∞	2.6% *	3.61% *	4.16% *	176.47
	0.35	1	0.95	4.35% *	4.35% *	4.35% *	202.66
			∞	2.22% *	3.61% *	4.43% *	172.91
		1.33	0.95	2.31% *	3.52% *	4.23% *	233.65
			∞	4.73% *	5.41% *	5.97% *	169.35
		1.67	0.95	———	———	———	———
			∞	3.72% *	4.47% *	5.56% *	140.06
	0.5	1	0.95	———	———	———	———
			∞	1.54% *	2.38% *	3.65% *	153.03
		1.33	0.95	2.33% *	2.33% *	2.33% *	139.8
			∞	3.02% *	3.14% *	3.24% *	143.61
		1.67	0.95	3.65% *	4.37% *	5.08% *	117.4
			∞	2.22%	3.45%	4.63%	109.52
pr226	0.2	1	0.95	21.45% *	24.61% *	28.2% *	460.71
			∞	21.97% *	22.93% *	23.52% *	458.94
		1.33	0.95	9.39% *	11.06% *	11.94% *	103.13
			∞	9.34% *	9.88% *	10.42% *	208.54
		1.67	0.95	6.91% *	7.3% *	7.53% *	100.0
			∞	6.89% *	6.96% *	7.05% *	170.25
	0.35	1	0.95	17.62% *	17.62% *	17.62% *	223.39
			∞	14.68% *	14.73% *	14.8% *	243.15
		1.33	0.95	11.4% *	11.71% *	12.28% *	146.92
			∞	11.18% *	12.33% *	14.19% *	167.04
		1.67	0.95	9.07% *	10.59% *	12.11% *	127.2
			∞	8.84% *	9.15% *	9.64% *	130.97
	0.5	1	0.95	———	———	———	———
			∞	19.09% *	19.57% *	19.86% *	291.4
		1.33	0.95	10.51% *	10.81% *	11.14% *	251.97
			∞	10.72% *	10.72% *	10.73% *	106.47
		1.67	0.95	11.03% *	11.56% *	12.04% *	146.6
			∞	10.58% *	11.14% *	11.57% *	127.53
Asymmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	13.13% *	15.59% *	17.55% *	10.27
			∞	14.82% *	16.39% *	18.37% *	12.38
		1.33	0.95	0.0%	1.88%	3.0%	9.2
			∞	4.08%	5.45%	7.75%	6.83

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1.67	0.95	1.57%	2.78%	4.66%	7.43
			∞	1.47%	2.03%	2.52%	5.2
	0.35	1	0.95	3.32%	5.27%	7.58%	8.61
			∞	2.23%	5.41%	8.14%	9.22
		1.33	0.95	3.36%	5.56%	7.69%	6.88
			∞	4.29%	5.75%	6.96%	11.09
		1.67	0.95	2.26%	4.35%	6.2%	4.27
			∞	1.01%	5.09%	10.25%	3.77
	0.5	1	0.95	0.06%	4.15%	6.6%	7.15
			∞	2.93%	4.99%	8.28%	7.15
		1.33	0.95	0.0%	1.39%	2.61%	3.51
			∞	2.55%	4.75%	7.33%	4.12
		1.67	0.95	0.0%	1.82%	4.42%	6.26
			∞	3.07%	4.29%	6.49%	4.34
ftv64	0.2	1	0.95	4.83%	7.65%	9.35%	7.13
			∞	3.19%	5.36%	8.28%	12.13
		1.33	0.95	2.32%	5.89%	10.3%	5.6
			∞	3.29%	5.5%	7.71%	3.77
		1.67	0.95	1.31%	4.4%	7.5%	4.1
			∞	1.25%	4.64%	10.39%	4.88
	0.35	1	0.95	4.93%	10.17%	14.79%	10.45
			∞	4.23%	7.04%	8.88%	7.05
		1.33	0.95	4.71%	8.31%	10.73%	6.92
			∞	1.35%	3.32%	4.71%	6.91
		1.67	0.95	2.18%	3.37%	4.68%	4.15
			∞	2.28%	3.73%	5.44%	4.23
	0.5	1	0.95	0.27%	1.01%	2.49%	14.58
			∞	1.41%	2.99%	5.41%	3.87
		1.33	0.95	0.81%	2.73%	6.57%	6.35
			∞	2.44%	3.44%	4.18%	4.89
		1.67	0.95	3.31%	5.36%	6.68%	5.07
			∞	3.09%	3.49%	4.29%	3.34
ft70	0.2	1	0.95	5.45% *	6.15% *	6.81% *	15.3
			∞	5.81% *	6.11% *	6.53% *	17.12
		1.33	0.95	4.03%	4.79%	5.18%	12.66
			∞	3.02%	3.68%	4.04%	17.11
		1.67	0.95	3.07%	4.28%	5.63%	11.19
			∞	2.71%	3.06%	3.5%	14.4
	0.35	1	0.95	2.77%	3.85%	4.61%	10.29
			∞	3.18%	4.21%	5.09%	17.84
		1.33	0.95	3.07%	3.4%	4.05%	14.03

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft70	0.35	1.33	∞	2.54%	3.39%	4.3%	9.2
ft70	0.35	1.67	0.95	2.88%	3.17%	3.44%	14.96
ft70	0.35	1.67	∞	3.88%	4.18%	4.66%	10.57
ft70	0.5	1	0.95	3.46%	4.02%	4.58%	20.25
ft70	0.5	1	∞	2.0%	3.44%	4.56%	13.71
ft70	0.5	1.33	0.95	3.0%	3.24%	3.48%	16.74
ft70	0.5	1.33	∞	2.64%	3.23%	4.21%	13.32
ft70	0.5	1.67	0.95	2.33%	3.51%	4.7%	17.0
ft70	0.5	1.67	∞	1.71%	2.67%	3.59%	16.09
kro124p	0.2	1	0.95	7.47% *	9.51% *	12.6% *	39.06
kro124p	0.2	1	∞	7.6% *	9.26% *	10.14% *	38.4
kro124p	0.2	1.33	0.95	0.71%	2.89%	5.03%	41.67
kro124p	0.2	1.33	∞	1.92%	4.5%	6.65%	14.24
kro124p	0.2	1.67	0.95	1.46%	1.89%	2.58%	22.11
kro124p	0.2	1.67	∞	4.25%	6.7%	9.08%	17.96
kro124p	0.35	1	0.95	—	—	—	—
kro124p	0.35	1	∞	12.55% *	15.72% *	17.55% *	39.29
kro124p	0.35	1.33	0.95	1.92%	3.4%	5.55%	12.79
kro124p	0.35	1.33	∞	1.29%	1.54%	1.75%	24.07
kro124p	0.35	1.67	0.95	1.29%	1.45%	1.74%	15.8
kro124p	0.35	1.67	∞	1.05%	1.28%	1.54%	21.81
kro124p	0.5	1	0.95	4.03% *	5.51% *	7.63% *	37.43
kro124p	0.5	1	∞	3.67% *	5.59% *	8.59% *	31.5
kro124p	0.5	1.33	0.95	2.29%	4.05%	4.94%	42.96
kro124p	0.5	1.33	∞	1.27%	3.03%	4.33%	22.37
kro124p	0.5	1.67	0.95	2.94%	3.46%	4.03%	21.96
kro124p	0.5	1.67	∞	1.77%	2.51%	3.47%	17.15
ftv170	0.2	1	0.95	18.03% *	18.03% *	18.03% *	78.33
ftv170	0.2	1	∞	16.3% *	17.43% *	18.23% *	76.59
ftv170	0.2	1.33	0.95	6.45%	10.91%	18.35%	49.62
ftv170	0.2	1.33	∞	5.0%	6.04%	6.85%	42.9
ftv170	0.2	1.67	0.95	4.13%	6.08%	8.01%	40.68
ftv170	0.2	1.67	∞	3.92%	5.18%	6.96%	33.95
ftv170	0.35	1	0.95	10.94% *	12.31% *	13.24% *	122.81
ftv170	0.35	1	∞	11.95% *	12.25% *	12.45% *	88.52
ftv170	0.35	1.33	0.95	5.99%	8.26%	10.82%	59.13
ftv170	0.35	1.33	∞	7.26%	7.97%	8.46%	57.38
ftv170	0.35	1.67	0.95	6.86%	7.95%	8.71%	53.56
ftv170	0.35	1.67	∞	6.42%	8.42%	11.29%	43.57
ftv170	0.5	1	0.95	10.3% *	11.28% *	12.08% *	82.13
ftv170	0.5	1	∞	8.63% *	11.99% *	16.93% *	53.53

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv170	0.5	1.33	0.95	8.97%	12.5%	16.09%	55.46
			∞	8.42%	10.0%	12.47%	48.8
		1.67	0.95	7.22%	10.28%	12.99%	55.13
			∞	6.13%	7.49%	8.49%	67.86

B.2.2 Results for a maximum of 2500 iterations

ANN as the constructive heuristic

Table B.13: Computational results, for $\omega = 1/2$ and $MaxIt = 2500$, using the solution obtained through the ANN heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	7.62
		1.33	0.95	4.4% *	4.43% *	4.45% *	9.95
			∞	5.38% *	5.45% *	5.74% *	6.71
		1.67	0.95	—	—	—	—
			∞	0.88%	1.77%	1.99%	7.44
	0.35	1	0.95	0.14%	0.53%	1.17%	7.55
			∞	0.0%	0.41%	1.17%	7.59
		1.33	0.95	0.0%	0.47%	2.36%	7.71
			∞	0.0%	0.75%	2.36%	5.71
		1.67	0.95	0.0%	0.0%	0.0%	6.18
			∞	0.0%	0.0%	0.0%	6.17
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.08%	0.4%	5.47
		1.33	0.95	—	—	—	—
			∞	0.29%	0.94%	1.11%	5.68
		1.67	0.95	1.33%	1.67%	2.74%	6.75
			∞	0.0%	0.96%	1.65%	5.02
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.45% *	1.45% *	13.22
		1.33	0.95	1.54% *	1.54% *	1.54% *	15.84
			∞	1.04% *	1.45% *	1.55% *	15.56
		1.67	0.95	0.5% *	0.78% *	1.27% *	13.01
			∞	0.51% *	0.62% *	0.8% *	12.04
	0.35	1	0.95	—	—	—	—

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.35	1	∞	4.13% *	4.36% *	5.27% *	17.03
		1.33	0.95	0.0%	0.02%	0.08%	11.57
			∞	0.0%	0.05%	0.08%	8.44
		1.67	0.95	0.0%	0.72%	0.95%	13.56
			∞	0.0%	0.54%	0.91%	11.44
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	13.74
		1.33	0.95	—	—	—	—
			∞	0.0%	1.05%	2.8%	8.99
		1.67	0.95	0.55%	0.55%	0.55%	7.28
			∞	0.0%	0.61%	1.08%	9.74
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.25% *	9.05% *	10.52% *	70.24
		1.33	0.95	—	—	—	—
			∞	0.0%	0.02%	0.04%	16.09
		1.67	0.95	—	—	—	—
			∞	0.0%	0.13%	0.44%	18.03
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.48% *	6.76% *	55.48
		1.33	0.95	—	—	—	—
			∞	1.56% *	1.6% *	1.74% *	40.3
		1.67	0.95	0.0%	0.01%	0.04%	14.08
			∞	0.0%	0.02%	0.09%	18.84
	0.5	1	0.95	—	—	—	—
			∞	2.57% *	3.06% *	3.8% *	69.86
		1.33	0.95	1.76% *	2.04% *	2.72% *	56.0
			∞	1.78% *	1.79% *	1.81% *	37.56
		1.67	0.95	0.0%	0.15%	0.71%	16.16
			∞	0.0%	0.0%	0.0%	18.98
pr124	0.2	1	0.95	—	—	—	—
			∞	5.58% *	6.06% *	6.54% *	43.69
		1.33	0.95	9.44% *	10.06% *	11.0% *	20.79
			∞	9.23% *	9.61% *	9.83% *	20.65
		1.67	0.95	10.93% *	10.93% *	10.93% *	17.66
			∞	11.0% *	11.0% *	11.0% *	16.08
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	9.67% *	14.72% *	23.6
		1.33	0.95	—	—	—	—
			∞	0.0%	0.35%	0.88%	23.74
		1.67	0.95	—	—	—	—
			∞	5.92% *	6.53% *	6.85% *	17.9

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr124	0.5	1	0.95	—	—	—	—
			∞	4.81% *	5.31% *	6.18% *	50.89
		1.33	0.95	—	—	—	—
			∞	4.33% *	4.58% *	4.95% *	27.22
		1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	16.2
pr152	0.2	1	0.95	13.54% *	14.18% *	14.67% *	95.32
			∞	13.98% *	14.38% *	15.08% *	83.19
		1.33	0.95	13.55% *	14.0% *	15.17% *	70.05
			∞	13.21% *	13.27% *	13.52% *	90.94
		1.67	0.95	13.07% *	13.14% *	13.42% *	60.19
			∞	12.95% *	12.95% *	12.95% *	46.68
	0.35	1	0.95	23.31% *	24.24% *	26.55% *	88.97
			∞	23.56% *	24.67% *	27.37% *	94.67
		1.33	0.95	19.36% *	19.63% *	19.7% *	34.82
			∞	21.19% *	21.19% *	21.19% *	39.09
		1.67	0.95	14.64% *	14.68% *	14.72% *	49.3
			∞	14.77% *	14.77% *	14.77% *	31.16
	0.5	1	0.95	—	—	—	—
			∞	9.56% *	9.56% *	9.56% *	42.01
		1.33	0.95	8.64% *	9.67% *	11.32% *	40.81
			∞	9.34% *	9.4% *	9.67% *	34.46
		1.67	0.95	—	—	—	—
			∞	23.61% *	23.61% *	23.61% *	36.07
rat195	0.2	1	0.95	—	—	—	—
			∞	4.53% *	5.57% *	7.26% *	259.63
		1.33	0.95	—	—	—	—
			∞	2.82% *	3.55% *	4.79% *	150.65
		1.67	0.95	2.91% *	3.56% *	3.9% *	120.69
			∞	2.6% *	3.32% *	3.76% *	114.44
	0.35	1	0.95	—	—	—	—
			∞	2.44% *	3.62% *	5.54% *	157.75
		1.33	0.95	1.56% *	2.46% *	3.52% *	108.91
			∞	2.14% *	2.77% *	3.52% *	119.97
		1.67	0.95	—	—	—	—
			∞	2.6% *	3.09% *	3.54% *	119.07
	0.5	1	0.95	—	—	—	—
			∞	1.28% *	2.51% *	3.17% *	188.67
		1.33	0.95	0.57% *	1.96% *	2.56% *	181.53
			∞	1.86% *	2.21% *	2.4% *	115.46

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
rat195	0.5	1.67	0.95	2.05% *	2.44% *	3.21% *	155.32
			∞	1.91%	2.36%	2.98%	98.26
pr226	0.2	1	0.95	22.1% *	24.52% *	27.94% *	213.85
			∞	22.31% *	22.76% *	23.35% *	273.37
		1.33	0.95	—	—	—	—
			∞	9.36% *	10.74% *	11.74% *	204.51
		1.67	0.95	6.72% *	7.2% *	7.58% *	161.58
			∞	6.84% *	7.09% *	7.75% *	149.36
	0.35	1	0.95	17.99% *	19.4% *	20.98% *	289.47
			∞	15.23% *	17.41% *	18.42% *	201.95
		1.33	0.95	—	—	—	—
			∞	10.48% *	11.02% *	11.25% *	140.79
		1.67	0.95	—	—	—	—
			∞	8.89% *	9.07% *	9.32% *	157.68
	0.5	1	0.95	14.17% *	15.33% *	16.94% *	628.23
			∞	17.32% *	19.47% *	21.59% *	264.38
		1.33	0.95	10.48% *	10.86% *	11.15% *	246.07
			∞	10.57% *	12.11% *	18.18% *	207.03
		1.67	0.95	—	—	—	—
			∞	10.52% *	10.65% *	10.91% *	132.26

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95	13.57% *	17.39% *	19.61% *	14.1
			∞	11.58% *	16.52% *	18.25% *	18.96
		1.33	0.95	0.74%	3.91%	7.1%	11.01
			∞	3.55%	5.18%	7.67%	6.55
		1.67	0.95	0.57%	2.03%	5.38%	5.1
			∞	0.0%	1.54%	2.65%	6.32
	0.35	1	0.95	3.63%	6.16%	9.22%	12.45
			∞	0.0%	2.14%	6.05%	15.35
		1.33	0.95	0.0%	5.12%	9.4%	10.43
			∞	1.74%	4.01%	6.34%	10.49
		1.67	0.95	0.06%	2.25%	5.01%	4.98
			∞	0.0%	2.09%	5.4%	4.68
	0.5	1	0.95	0.06%	2.34%	3.68%	9.17
			∞	3.35%	4.72%	10.17%	8.96
		1.33	0.95	0.0%	0.89%	2.62%	6.38
			∞	0.0%	1.98%	3.36%	5.48
		1.67	0.95	0.0%	1.86%	3.75%	5.11
			∞	0.0%	1.27%	2.55%	4.19

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.2	1	0.95	—	—	—	—
			∞	3.88%	6.46%	7.65%	17.08
		1.33	0.95	—	—	—	—
			∞	1.67%	3.89%	7.01%	6.65
		1.67	0.95	3.37%	4.6%	6.47%	9.5
			∞	3.37%	3.99%	5.06%	8.96
	0.35	1	0.95	0.05%	2.81%	9.05%	11.14
			∞	0.05%	4.51%	9.48%	9.17
		1.33	0.95	2.44%	3.94%	6.01%	7.74
			∞	1.79%	3.39%	6.5%	5.74
		1.67	0.95	0.76%	2.32%	3.37%	5.2
			∞	0.98%	2.56%	4.4%	5.6
	0.5	1	0.95	1.41%	3.22%	6.7%	13.06
			∞	0.38%	4.81%	8.54%	9.02
		1.33	0.95	0.98%	3.7%	10.21%	6.16
			∞	0.0%	2.28%	4.29%	7.41
		1.67	0.95	0.87%	2.61%	4.18%	4.69
			∞	0.33%	0.71%	1.19%	4.57
ft70	0.2	1	0.95	5.55% *	6.42% *	7.0% *	19.99
			∞	5.5% *	6.2% *	6.73% *	22.0
		1.33	0.95	3.25%	3.98%	4.87%	25.06
			∞	2.56%	3.15%	3.89%	17.93
		1.67	0.95	2.73%	3.9%	4.83%	24.65
			∞	3.25%	3.84%	4.74%	16.94
	0.35	1	0.95	2.77%	3.28%	4.12%	22.47
			∞	2.22%	2.85%	3.87%	24.31
		1.33	0.95	2.15%	3.77%	5.01%	18.69
			∞	2.12%	2.43%	2.97%	18.84
		1.67	0.95	—	—	—	—
			∞	2.29%	2.75%	4.16%	21.34
	0.5	1	0.95	—	—	—	—
			∞	2.31%	3.4%	4.45%	13.1
		1.33	0.95	—	—	—	—
			∞	2.4%	3.09%	3.33%	16.64
		1.67	0.95	—	—	—	—
			∞	1.9%	2.74%	4.1%	17.77
kro124p	0.2	1	0.95	—	—	—	—
			∞	6.83% *	9.21% *	13.35% *	54.72
		1.33	0.95	4.1%	5.11%	5.95%	31.54
			∞	3.2%	4.66%	6.91%	27.79
		1.67	0.95	6.01%	7.94%	11.12%	31.27

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kro124p	0.2	1.67	∞	3.87%	6.26%	10.89%	32.83
	0.35	1	0.95	—	—	—	—
			∞	6.47% *	8.48% *	10.69% *	66.12
		1.33	0.95	3.88%	5.47%	6.68%	50.36
			∞	1.73%	3.98%	5.28%	36.04
		1.67	0.95	5.67%	7.63%	10.1%	34.69
			∞	3.61%	8.31%	11.82%	26.13
	0.5	1	0.95	—	—	—	—
			∞	3.74% *	8.18% *	11.49% *	65.2
		1.33	0.95	5.09%	7.36%	9.66%	40.26
			∞	7.06%	8.62%	10.43%	42.8
		1.67	0.95	5.13%	5.89%	7.02%	42.81
			∞	3.94%	7.61%	11.65%	28.53
ftv170	0.2	1	0.95	—	—	—	—
			∞	8.13% *	14.71% *	17.18% *	147.31
		1.33	0.95	3.63%	8.93%	12.22%	61.53
			∞	6.09%	8.69%	11.06%	72.25
		1.67	0.95	7.22%	8.55%	11.78%	54.25
			∞	9.35%	10.17%	11.97%	52.86
	0.35	1	0.95	—	—	—	—
			∞	4.07% *	7.12% *	8.32% *	105.43
		1.33	0.95	—	—	—	—
			∞	5.74%	9.52%	12.38%	56.95
		1.67	0.95	6.1%	8.42%	12.45%	38.32
			∞	5.41%	6.29%	8.68%	65.77
	0.5	1	0.95	—	—	—	—
			∞	4.99% *	6.72% *	8.99% *	71.2
		1.33	0.95	—	—	—	—
			∞	4.84%	7.91%	12.47%	48.68
		1.67	0.95	5.84%	8.47%	13.58%	54.77
			∞	5.88%	6.85%	7.84%	48.72

AFI as the constructive heuristic

Table B.14: Computational results, for $\omega = 1/2$ and $MaxIt = 2500$, using the solution obtained through the AFI heuristic as the initial solution.

Symmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	8.41
		1.33	0.95	4.4% *	4.49% *	4.76% *	8.02
			∞	5.38% *	5.47% *	5.74% *	5.11
		1.67	0.95	0.0%	1.04%	1.99%	9.48
			∞	0.88%	1.77%	1.99%	7.05
	0.35	1	0.95	0.0%	0.33%	1.17%	6.44
			∞	0.0%	0.18%	0.23%	6.84
		1.33	0.95	0.0%	0.47%	2.36%	7.8
			∞	0.0%	0.0%	0.0%	6.26
		1.67	0.95	0.0%	0.47%	2.36%	5.45
			∞	0.0%	0.47%	2.36%	5.57
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	5.01
		1.33	0.95	0.0%	1.07%	1.34%	7.3
			∞	0.29%	0.29%	0.29%	4.13
		1.67	0.95	0.0%	0.47%	1.33%	6.16
			∞	0.0%	0.53%	1.33%	5.03
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.81% *	2.23% *	14.8
		1.33	0.95	1.03% *	3.16% *	3.86% *	19.66
			∞	1.04% *	2.14% *	3.84% *	12.92
		1.67	0.95	0.39% *	0.46% *	0.74% *	14.52
			∞	0.33% *	0.37% *	0.52% *	14.38
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.28% *	4.89% *	15.44
		1.33	0.95	0.0%	0.02%	0.08%	9.75
			∞	0.0%	0.14%	0.21%	11.79
		1.67	0.95	0.0%	0.03%	0.07%	12.23
			∞	0.0%	0.09%	0.26%	10.31
	0.5	1	0.95	—	—	—	—
			∞	0.0%	2.7%	4.07%	15.09
		1.33	0.95	0.0%	1.53%	2.74%	11.5
			∞	0.0%	1.65%	2.74%	12.02

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.5	1.67	0.95	—	—	—	—
			∞	0.0%	0.09%	0.26%	10.28
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.25% *	9.46% *	10.98% *	67.07
		1.33	0.95	0.0%	0.0%	0.0%	20.35
			∞	0.0%	0.0%	0.0%	15.47
		1.67	0.95	0.0%	0.0%	0.0%	17.54
			∞	0.0%	0.0%	0.0%	16.21
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.33% *	6.33% *	53.12
		1.33	0.95	2.79% *	4.23% *	4.85% *	26.07
			∞	1.56% *	1.57% *	1.6% *	57.97
		1.67	0.95	0.0%	0.0%	0.0%	16.0
			∞	0.0%	0.02%	0.04%	20.44
	0.5	1	0.95	—	—	—	—
			∞	2.62% *	3.21% *	4.75% *	53.2
		1.33	0.95	1.76% *	1.92% *	2.57% *	76.34
			∞	1.78% *	2.0% *	2.9% *	33.82
		1.67	0.95	0.0%	0.0%	0.0%	18.86
			∞	0.0%	0.0%	0.0%	21.25
pr124	0.2	1	0.95	6.84% *	6.93% *	7.32% *	46.74
			∞	5.04% *	5.95% *	6.54% *	32.52
		1.33	0.95	9.44% *	9.44% *	9.44% *	20.89
			∞	9.23% *	9.27% *	9.3% *	28.99
		1.67	0.95	10.93% *	10.93% *	10.93% *	20.13
			∞	11.0% *	11.0% *	11.0% *	18.03
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	20.94
		1.33	0.95	—	—	—	—
			∞	0.0%	0.42%	0.88%	20.18
		1.67	0.95	6.69% *	7.27% *	8.14% *	31.5
			∞	5.92% *	6.61% *	7.44% *	22.25
	0.5	1	0.95	—	—	—	—
			∞	4.32% *	4.7% *	5.27% *	36.48
		1.33	0.95	1.98% *	2.15% *	2.87% *	30.89
			∞	4.33% *	4.58% *	4.95% *	27.31
		1.67	0.95	0.0%	0.29%	0.78%	18.54
			∞	0.0%	0.36%	0.6%	27.35
pr152	0.2	1	0.95	—	—	—	—
			∞	13.95% *	14.54% *	15.37% *	107.5
		1.33	0.95	13.43% *	14.31% *	15.11% *	86.9

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr152	0.2	1.33	∞	13.19% *	13.58% *	14.13% *	82.15
		1.67	0.95	12.48% *	12.97% *	13.1% *	34.82
			∞	12.36% *	12.99% *	14.24% *	44.15
	0.35	1	0.95	—	—	—	—
			∞	22.65% *	24.07% *	25.08% *	74.12
		1.33	0.95	18.38% *	18.82% *	19.12% *	50.19
			∞	19.94% *	20.32% *	20.6% *	42.69
		1.67	0.95	—	—	—	—
			∞	14.1% *	14.64% *	14.82% *	39.66
	0.5	1	0.95	—	—	—	—
			∞	8.65% *	9.14% *	9.33% *	47.87
		1.33	0.95	9.04% *	9.21% *	9.33% *	37.15
			∞	8.43% *	8.89% *	9.32% *	52.69
		1.67	0.95	—	—	—	—
			∞	22.91% *	22.93% *	23.01% *	47.79
rat195	0.2	1	0.95	—	—	—	—
			∞	5.43% *	7.04% *	8.12% *	312.71
		1.33	0.95	—	—	—	—
			∞	4.07% *	4.54% *	4.83% *	157.73
		1.67	0.95	2.29% *	4.2% *	6.01% *	134.16
			∞	3.85% *	4.42% *	5.06% *	150.29
	0.35	1	0.95	—	—	—	—
			∞	3.01% *	3.75% *	4.26% *	202.63
		1.33	0.95	4.09% *	4.44% *	5.12% *	135.64
			∞	3.61% *	4.06% *	4.5% *	155.09
		1.67	0.95	—	—	—	—
			∞	3.0% *	4.03% *	5.38% *	146.68
	0.5	1	0.95	—	—	—	—
			∞	2.86% *	4.0% *	4.93% *	141.34
		1.33	0.95	0.88% *	2.21% *	4.01% *	266.81
			∞	2.4% *	3.92% *	6.66% *	165.56
		1.67	0.95	2.41% *	3.48% *	4.94% *	134.59
			∞	2.14%	3.19%	4.63%	160.67
pr226	0.2	1	0.95	—	—	—	—
			∞	23.19% *	25.08% *	27.76% *	310.7
		1.33	0.95	9.19% *	10.44% *	12.12% *	222.35
			∞	9.07% *	10.47% *	11.83% *	182.47
		1.67	0.95	6.72% *	6.78% *	6.83% *	162.55
			∞	6.85% *	6.94% *	7.1% *	134.94
	0.35	1	0.95	—	—	—	—
			∞	14.51% *	14.61% *	14.89% *	252.52

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
pr226	0.35	1.33	0.95	11.34% *	11.4% *	11.47% *	166.1
			∞	10.48% *	10.63% *	11.14% *	174.28
		1.67	0.95	9.12% *	9.18% *	9.24% *	201.06
			∞	8.76% *	8.85% *	8.9% *	150.8
	0.5	1	0.95				
			∞	19.16% *	19.43% *	19.71% *	383.71
		1.33	0.95	11.21% *	11.58% *	11.91% *	186.87
			∞	10.55% *	10.6% *	10.68% *	131.32
		1.67	0.95	11.04% *	11.23% *	11.83% *	153.58
			∞	10.55% *	10.6% *	10.66% *	95.24

Asymmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1	0.95				
			∞	11.58% *	14.44% *	21.69% *	15.23
		1.33	0.95	0.74%	4.27%	7.46%	8.59
			∞	3.27%	5.77%	7.6%	7.18
		1.67	0.95	0.0%	2.44%	4.13%	6.23
			∞	0.0%	2.46%	9.5%	5.4
	0.35	1	0.95				
			∞	0.85%	3.56%	5.43%	13.31
		1.33	0.95	2.84%	3.7%	4.37%	9.38
			∞	0.85%	2.92%	4.3%	13.94
		1.67	0.95	0.0%	1.12%	3.16%	5.27
			∞	0.0%	1.26%	2.59%	4.83
	0.5	1	0.95	2.61%	3.62%	4.23%	5.34
			∞	2.61%	4.43%	7.6%	5.69
		1.33	0.95	1.71%	2.74%	3.48%	5.86
			∞	0.06%	1.84%	3.98%	6.33
		1.67	0.95	1.56%	2.76%	4.4%	4.97
			∞	0.06%	1.13%	1.96%	5.42
ftv64	0.2	1	0.95	3.88%	6.73%	9.72%	13.97
			∞	2.81%	7.17%	10.14%	10.9
		1.33	0.95	2.53%	4.47%	5.44%	6.53
			∞	0.97%	3.28%	6.36%	8.36
		1.67	0.95				
			∞	2.28%	2.66%	3.48%	5.97
	0.35	1	0.95	0.05%	2.87%	6.39%	12.65
			∞	0.05%	3.5%	6.39%	11.0
		1.33	0.95	0.22%	2.78%	5.8%	7.49
			∞	1.57%	1.97%	2.82%	7.98

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.35	1.67	0.95	2.28%	4.05%	5.33%	3.86
			∞	1.96%	2.81%	4.57%	6.09
	0.5	1	0.95				
			∞	0.43%	1.75%	3.08%	9.13
		1.33	0.95	0.98%	1.27%	1.95%	5.17
			∞	0.98%	2.6%	4.67%	5.57
		1.67	0.95				
			∞	0.98%	2.13%	3.53%	4.47
ft70	0.2	1	0.95	4.48% *	5.98% *	7.11% *	28.18
			∞	4.14% *	5.46% *	6.42% *	35.49
		1.33	0.95	3.45%	4.19%	5.08%	22.02
			∞	2.14%	2.46%	2.72%	25.48
		1.67	0.95	2.08%	3.41%	4.28%	20.37
			∞	1.7%	2.69%	4.12%	17.67
	0.35	1	0.95	2.19%	2.8%	3.74%	21.67
			∞	2.13%	3.11%	4.74%	18.34
		1.33	0.95	1.65%	2.81%	3.93%	17.91
			∞	2.09%	2.7%	3.54%	25.09
		1.67	0.95	3.05%	3.41%	3.97%	17.15
			∞	2.28%	3.38%	4.89%	16.78
	0.5	1	0.95	1.96%	3.09%	4.59%	28.2
			∞	2.11%	2.86%	3.42%	18.01
		1.33	0.95	2.43%	3.4%	5.37%	16.24
			∞	1.32%	2.83%	3.9%	15.01
		1.67	0.95	2.02%	3.15%	4.11%	16.39
			∞	2.46%	2.99%	3.69%	18.15
kro124p	0.2	1	0.95	4.58% *	6.36% *	7.29% *	45.32
			∞	4.81% *	6.04% *	6.99% *	57.84
		1.33	0.95	0.77%	1.96%	5.59%	24.16
			∞	0.15%	2.1%	4.92%	27.81
		1.67	0.95	0.62%	1.14%	1.72%	31.86
			∞	0.81%	1.38%	2.39%	24.19
	0.35	1	0.95				
			∞	8.37% *	9.84% *	11.63% *	76.96
		1.33	0.95	1.01%	2.08%	2.97%	33.67
			∞	0.53%	1.65%	2.13%	30.12
		1.67	0.95	0.43%	1.49%	3.08%	25.48
			∞	3.09%	3.74%	4.13%	19.69
	0.5	1	0.95	2.49% *	4.04% *	5.29% *	41.89
			∞	2.73% *	4.9% *	6.45% *	43.52
		1.33	0.95	3.12%	4.33%	6.13%	32.61

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.5	1.33	∞	3.38%	4.6%	6.28%	39.21
		1.67	0.95	1.13%	2.79%	3.83%	29.3
			∞	0.8%	2.21%	3.78%	26.49
ftv170	0.2	1	0.95	——	——	——	——
			∞	5.0% *	7.24% *	9.41% *	134.46
		1.33	0.95	1.2%	3.81%	5.44%	72.52
			∞	3.23%	4.79%	6.42%	54.44
		1.67	0.95	3.26%	6.58%	9.83%	52.21
			∞	1.23%	3.05%	4.64%	73.86
	0.35	1	0.95	6.31% *	11.63% *	13.53% *	91.66
			∞	7.81% *	11.41% *	16.88% *	101.41
		1.33	0.95	2.87%	4.11%	5.74%	51.59
			∞	2.4%	6.37%	9.55%	80.32
		1.67	0.95	4.36%	6.03%	6.93%	56.42
			∞	3.16%	3.91%	4.97%	44.95
	0.5	1	0.95	7.54% *	9.81% *	14.04% *	140.17
			∞	8.41% *	9.6% *	10.96% *	126.29
		1.33	0.95	4.84%	6.64%	9.36%	82.85
			∞	4.27%	10.16%	12.94%	80.0
		1.67	0.95	2.4%	3.95%	7.7%	47.63
			∞	3.16%	4.44%	5.7%	48.42

ARI as the constructive heuristic

Table B.15: Computational results, for $\omega = 1/2$ and $MaxIt = 2500$, using the solution obtained through the ARI heuristic as the initial solution.

Symmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.2	1	0.95	——	——	——	——
			∞	4.49% *	4.65% *	5.27% *	17.03
		1.33	0.95	4.4% *	4.41% *	4.45% *	12.73
			∞	5.38% *	5.38% *	5.38% *	10.72
		1.67	0.95	0.0%	0.52%	0.88%	18.08
			∞	0.88%	2.02%	3.26%	11.91
	0.35	1	0.95	0.0%	0.5%	1.29%	11.4
			∞	0.0%	0.08%	0.42%	15.49
		1.33	0.95	0.0%	1.91%	2.93%	11.51
			∞	0.0%	1.12%	2.93%	12.04

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.35	1.67	0.95	0.0%	0.73%	3.63%	10.05
			∞	0.0%	1.06%	2.91%	10.32
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.53%	2.64%	10.15
		1.33	0.95	0.83%	1.03%	1.34%	19.76
			∞	0.0%	0.23%	0.29%	8.49
		1.67	0.95	0.0%	0.47%	1.33%	8.09
			∞	0.0%	0.1%	0.52%	10.64
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.75% *	2.16% *	36.52
		1.33	0.95	3.86% *	3.86% *	3.86% *	23.5
			∞	1.11% *	1.43% *	1.86% *	23.24
		1.67	0.95	0.5% *	0.63% *	0.74% *	28.86
			∞	0.33% *	0.52% *	0.82% *	27.25
	0.35	1	0.95	4.22% *	4.22% *	4.22% *	32.67
			∞	4.13% *	4.36% *	5.27% *	28.49
		1.33	0.95	0.0%	0.08%	0.21%	21.19
			∞	0.0%	0.03%	0.08%	20.39
		1.67	0.95	0.0%	0.47%	0.95%	29.48
			∞	0.0%	0.2%	0.95%	22.6
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.02%	0.07%	34.97
		1.33	0.95	0.0%	1.16%	2.74%	25.15
			∞	0.0%	0.55%	2.74%	29.56
		1.67	0.95	0.55%	0.57%	0.59%	20.75
			∞	0.0%	0.68%	0.95%	18.25
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.25% *	8.76% *	9.52% *	184.62
		1.33	0.95	0.0%	0.0%	0.0%	27.06
			∞	0.0%	0.01%	0.04%	34.19
		1.67	0.95	0.0%	0.01%	0.04%	31.41
			∞	0.0%	0.02%	0.09%	32.87
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.47% *	6.93% *	125.12
		1.33	0.95	2.44% *	3.59% *	4.92% *	81.35
			∞	1.56% *	2.55% *	5.69% *	85.45
		1.67	0.95	0.0%	0.02%	0.09%	33.47
			∞	0.0%	0.02%	0.09%	35.63
	0.5	1	0.95	—	—	—	—
			∞	2.41% *	3.2% *	4.38% *	104.57
		1.33	0.95	1.76% *	3.4% *	4.3% *	72.75

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kroA100	0.5	1.33	∞	1.78% *	1.79% *	1.81% *	63.69
		1.67	0.95	0.0%	0.0%	0.0%	38.74
			∞	0.0%	0.0%	0.0%	35.25
pr124	0.2	1	0.95	—	—	—	—
			∞	6.06% *	6.52% *	7.19% *	97.4
		1.33	0.95	9.44% *	9.44% *	9.44% *	42.23
			∞	9.3% *	9.61% *	10.86% *	45.28
		1.67	0.95	10.93% *	11.4% *	12.82% *	37.46
			∞	11.0% *	11.1% *	11.53% *	37.11
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	31.37
		1.33	0.95	0.6%	1.32%	2.04%	39.92
			∞	0.0%	0.24%	0.6%	50.29
		1.67	0.95	6.69% *	6.99% *	8.22% *	70.67
			∞	5.92% *	5.92% *	5.92% *	47.08
	0.5	1	0.95	—	—	—	—
			∞	4.32% *	4.9% *	5.27% *	83.85
		1.33	0.95	1.98% *	2.95% *	3.44% *	49.85
			∞	4.33% *	4.34% *	4.4% *	42.9
		1.67	0.95	0.0%	0.3%	0.6%	44.65
			∞	0.0%	0.59%	1.51%	40.12
pr152	0.2	1	0.95	13.99% *	13.99% *	13.99% *	133.75
			∞	14.02% *	14.17% *	14.33% *	152.64
		1.33	0.95	14.16% *	14.43% *	14.94% *	210.35
			∞	13.19% *	13.72% *	15.17% *	161.84
		1.67	0.95	13.1% *	13.61% *	14.59% *	99.38
			∞	12.36% *	12.68% *	12.98% *	101.39
	0.35	1	0.95	24.02% *	24.21% *	24.39% *	96.48
			∞	22.65% *	23.71% *	24.48% *	150.04
		1.33	0.95	18.38% *	18.71% *	19.7% *	86.32
			∞	19.85% *	20.55% *	21.19% *	62.26
		1.67	0.95	14.07% *	14.07% *	14.07% *	81.3
			∞	13.89% *	14.25% *	14.94% *	86.62
	0.5	1	0.95	—	—	—	—
			∞	8.65% *	12.35% *	19.11% *	113.56
		1.33	0.95	8.44% *	9.08% *	9.64% *	93.77
			∞	8.43% *	9.01% *	9.63% *	73.27
		1.67	0.95	—	—	—	—
			∞	22.24% *	22.82% *	23.36% *	77.01
rat195	0.2	1	0.95	—	—	—	—
			∞	5.61% *	6.72% *	8.21% *	579.07

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
rat195	0.2	1.33	0.95	—	—	—	—
			∞	2.6% *	3.8% *	5.51% *	350.28
		1.67	0.95	2.38% *	3.26% *	4.57% *	327.07
			∞	3.94% *	4.37% *	5.37% *	280.9
	0.35	1	0.95	—	—	—	—
			∞	1.77% *	3.7% *	6.69% *	400.48
		1.33	0.95	2.98% *	3.43% *	4.18% *	315.57
			∞	2.27% *	3.76% *	5.93% *	318.29
		1.67	0.95	—	—	—	—
			∞	1.88% *	3.52% *	4.17% *	296.32
	0.5	1	0.95	—	—	—	—
			∞	2.46% *	3.49% *	5.32% *	334.12
		1.33	0.95	3.04% *	3.61% *	4.19% *	299.24
			∞	2.09% *	3.12% *	5.28% *	268.23
		1.67	0.95	2.23% *	3.89% *	4.81% *	426.2
			∞	2.36%	3.31%	3.83%	269.86
pr226	0.2	1	0.95	21.47% *	23.4% *	25.89% *	1018.22
			∞	22.35% *	25.16% *	27.28% *	715.39
		1.33	0.95	9.28% *	10.72% *	14.93% *	426.14
			∞	9.25% *	9.48% *	9.6% *	714.83
		1.67	0.95	6.73% *	7.01% *	7.39% *	239.59
			∞	6.89% *	6.98% *	7.3% *	239.21
	0.35	1	0.95	14.09% *	19.5% *	26.77% *	639.84
			∞	14.51% *	15.26% *	17.97% *	523.0
		1.33	0.95	11.33% *	11.46% *	11.53% *	242.18
			∞	10.48% *	10.59% *	10.74% *	301.08
		1.67	0.95	9.18% *	9.54% *	9.82% *	185.3
			∞	8.8% *	9.68% *	12.4% *	232.86
	0.5	1	0.95	—	—	—	—
			∞	19.01% *	19.75% *	20.45% *	510.61
		1.33	0.95	11.4% *	14.37% *	19.61% *	328.52
			∞	10.52% *	10.95% *	11.62% *	212.98
		1.67	0.95	11.05% *	11.31% *	11.6% *	230.0
			∞	10.57% *	11.0% *	11.31% *	207.82

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	11.52% *	14.01% *	19.2% *	35.44
			∞	12.62% *	14.97% *	19.56% *	39.62
		1.33	0.95	3.28%	5.92%	10.1%	17.4
			∞	1.72%	4.43%	7.82%	21.4

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft53	0.2	1.67	0.95	2.09%	3.09%	4.43%	13.32
			∞	0.2%	2.3%	6.22%	12.57
	0.35	1	0.95	0.85%	6.17%	11.49%	15.16
			∞	1.88%	2.97%	5.05%	20.22
		1.33	0.95	0.72%	3.36%	6.48%	19.86
			∞	2.6%	4.79%	6.49%	14.32
		1.67	0.95	0.06%	2.62%	9.85%	10.71
			∞	0.0%	1.95%	3.3%	12.38
	0.5	1	0.95	0.0%	1.13%	3.32%	17.83
			∞	2.61%	6.0%	8.96%	11.41
		1.33	0.95	0.0%	2.23%	4.3%	8.57
			∞	0.0%	1.94%	5.95%	8.77
		1.67	0.95	1.03%	5.38%	10.07%	9.45
			∞	1.06%	2.29%	2.74%	10.69
ftv64	0.2	1	0.95	1.75%	6.05%	11.31%	22.32
			∞	4.62%	6.55%	7.97%	22.92
		1.33	0.95	0.16%	3.09%	5.77%	12.28
			∞	1.62%	5.04%	10.13%	12.23
		1.67	0.95	2.99%	5.38%	6.91%	12.75
			∞	2.28%	4.98%	6.91%	11.22
	0.35	1	0.95	2.98%	4.5%	9.97%	24.15
			∞	0.05%	3.1%	5.25%	25.52
		1.33	0.95	1.84%	4.25%	8.78%	16.05
			∞	0.65%	3.48%	8.29%	11.35
		1.67	0.95	1.96%	3.03%	4.79%	11.94
			∞	2.66%	3.37%	4.46%	10.3
	0.5	1	0.95	1.03%	1.03%	1.03%	29.89
			∞	0.54%	3.45%	5.84%	14.59
		1.33	0.95	0.98%	3.15%	7.06%	10.61
			∞	0.81%	2.54%	6.51%	14.43
		1.67	0.95	0.6%	1.78%	3.53%	14.3
			∞	0.87%	2.39%	3.47%	9.57
ft70	0.2	1	0.95	5.19% *	6.33% *	7.07% *	39.15
			∞	3.58% *	5.24% *	6.73% *	40.24
		1.33	0.95	2.86%	3.62%	4.4%	44.93
			∞	2.25%	3.02%	4.02%	38.7
		1.67	0.95	2.19%	3.72%	4.51%	35.59
			∞	1.76%	3.03%	4.1%	30.09
	0.35	1	0.95	1.26%	2.52%	3.45%	36.07
			∞	2.36%	3.16%	4.63%	45.98
		1.33	0.95	1.79%	2.58%	3.16%	35.88

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft70	0.35	1.33	∞	2.49%	3.46%	5.23%	31.39
		1.67	0.95	2.46%	2.99%	3.76%	25.1
			∞	2.69%	3.26%	3.78%	32.48
	0.5	1	0.95	1.43%	1.43%	1.43%	150.83
			∞	2.49%	3.26%	3.75%	32.39
		1.33	0.95	2.8%	3.37%	3.71%	24.13
			∞	1.55%	3.17%	4.78%	23.58
		1.67	0.95	2.46%	3.55%	4.24%	30.66
			∞	2.09%	3.46%	4.3%	25.75
kro124p	0.2	1	0.95	7.42% *	8.04% *	8.74% *	91.73
			∞	6.81% *	8.76% *	10.54% *	100.51
		1.33	0.95	0.48%	3.86%	6.37%	43.09
			∞	0.76%	1.86%	5.04%	47.02
		1.67	0.95	0.53%	3.48%	5.18%	50.46
			∞	2.2%	4.35%	5.66%	52.33
	0.35	1	0.95	—	—	—	—
			∞	6.59% *	10.16% *	15.13% *	129.33
		1.33	0.95	0.34%	1.03%	1.97%	68.39
			∞	0.71%	1.83%	3.0%	74.11
		1.67	0.95	0.7%	1.74%	5.03%	48.99
			∞	0.47%	1.9%	6.45%	63.53
	0.5	1	0.95	3.39% *	4.36% *	5.11% *	60.84
			∞	0.85% *	5.6% *	9.8% *	99.93
		1.33	0.95	0.52%	2.35%	3.74%	98.6
			∞	2.16%	3.39%	4.4%	84.63
		1.67	0.95	0.51%	2.22%	4.29%	73.58
			∞	0.91%	2.95%	5.02%	36.46
ftv170	0.2	1	0.95	—	—	—	—
			∞	6.49% *	14.6% *	22.54% *	192.87
		1.33	0.95	2.25%	4.89%	11.06%	124.11
			∞	3.95%	6.15%	9.28%	93.23
		1.67	0.95	1.45%	6.38%	10.48%	119.05
			∞	3.81%	7.45%	9.68%	79.12
	0.35	1	0.95	8.02% *	12.13% *	16.05% *	217.51
			∞	5.19% *	9.71% *	15.25% *	228.71
		1.33	0.95	7.04%	9.16%	10.74%	114.31
			∞	6.46%	7.28%	8.53%	128.3
		1.67	0.95	1.67%	4.23%	8.64%	78.42
			∞	5.59%	8.13%	9.87%	100.99
	0.5	1	0.95	5.77% *	9.28% *	13.71% *	213.35
			∞	6.37% *	10.39% *	16.16% *	143.61

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ftv170	0.5	1.33	0.95	7.05%	8.68%	11.06%	162.58
			∞	4.7%	6.58%	8.35%	114.47
		1.67	0.95	4.46%	6.73%	7.59%	131.88
			∞	0.73%	3.48%	6.21%	124.18

B.2.3 Results for a maximum of 5000 iterations

ANN as the constructive heuristic

Table B.16: Computational results, for $\omega = 1/2$ and $MaxIt = 5000$, using the solution obtained through the ANN heuristic as the initial solution.

Symmetric instances							
Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	15.11
		1.33	0.95	4.4% *	4.54% *	4.76% *	10.57
			∞	5.24% *	5.29% *	5.38% *	11.21
		1.67	0.95	—	—	—	—
			∞	1.99%	1.99%	1.99%	10.38
	0.35	1	0.95	0.23%	0.41%	0.78%	12.74
			∞	0.0%	0.18%	0.33%	10.11
		1.33	0.95	0.0%	0.0%	0.0%	11.12
			∞	0.0%	0.0%	0.0%	10.18
		1.67	0.95	0.0%	0.0%	0.0%	7.11
			∞	0.0%	0.0%	0.0%	8.82
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	7.68
		1.33	0.95	—	—	—	—
			∞	1.11%	1.11%	1.11%	6.19
		1.67	0.95	0.0%	0.61%	1.33%	10.9
			∞	0.0%	0.44%	1.33%	9.0
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.45% *	1.45% *	22.08
		1.33	0.95	1.54% *	1.54% *	1.54% *	22.17
			∞	1.55% *	1.55% *	1.55% *	21.53
		1.67	0.95	0.39% *	0.53% *	0.61% *	22.66
			∞	0.44% *	0.49% *	0.52% *	21.32
	0.35	1	0.95	—	—	—	—

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.35	1	∞	4.13% *	4.13% *	4.13% *	20.73
		1.33	0.95	0.0%	0.03%	0.08%	12.94
			∞	0.0%	0.03%	0.08%	14.97
		1.67	0.95	0.0%	0.05%	0.07%	27.04
			∞	0.82%	0.82%	0.82%	13.72
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.03%	0.08%	23.73
		1.33	0.95	0.0%	0.0%	0.0%	20.22
			∞	0.0%	0.01%	0.01%	21.12
		1.67	0.95	0.55%	0.55%	0.55%	14.7
			∞	0.82%	0.82%	0.82%	12.81
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.25% *	8.62% *	8.85% *	132.7
		1.33	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	35.31
		1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	28.75
	0.35	1	0.95	—	—	—	—
			∞	6.33% *	6.33% *	6.33% *	90.15
		1.33	0.95	—	—	—	—
			∞	1.56% *	1.57% *	1.6% *	69.68
		1.67	0.95	0.0%	0.0%	0.0%	26.24
			∞	0.0%	0.0%	0.0%	32.79
	0.5	1	0.95	—	—	—	—
			∞	2.62% *	2.7% *	2.81% *	96.81
		1.33	0.95	1.76% *	1.76% *	1.76% *	113.09
			∞	1.78% *	1.78% *	1.79% *	80.65
		1.67	0.95	0.0%	0.0%	0.0%	30.09
			∞	0.0%	0.01%	0.04%	31.98
pr124	0.2	1	0.95	—	—	—	—
			∞	6.06% *	6.06% *	6.06% *	63.32
		1.33	0.95	9.44% *	10.14% *	11.0% *	46.5
			∞	9.3% *	9.66% *	9.83% *	37.22
		1.67	0.95	10.93% *	10.93% *	10.93% *	35.68
			∞	11.0% *	11.0% *	11.0% *	35.64
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	34.96
		1.33	0.95	—	—	—	—
			∞	0.0%	0.49%	0.88%	45.7
		1.67	0.95	—	—	—	—
			∞	5.92% *	6.34% *	6.56% *	37.95

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr124	0.5	1	0.95	______	______	______	______
			∞	4.51% *	4.94% *	5.27% *	103.68
		1.33	0.95	______	______	______	______
			∞	4.33% *	4.54% *	4.95% *	48.56
		1.67	0.95	______	______	______	______
			∞	0.0%	0.0%	0.0%	30.11
pr152	0.2	1	0.95	13.9% *	14.16% *	14.6% *	154.29
			∞	14.35% *	14.43% *	14.48% *	224.51
		1.33	0.95	13.19% *	13.43% *	13.61% *	182.26
			∞	13.21% *	13.64% *	14.02% *	156.43
		1.67	0.95	13.07% *	13.07% *	13.08% *	94.24
			∞	12.95% *	13.06% *	13.3% *	78.77
	0.35	1	0.95	22.79% *	23.01% *	23.31% *	221.02
			∞	23.18% *	24.6% *	27.37% *	157.68
		1.33	0.95	19.7% *	19.7% *	19.7% *	63.09
			∞	19.85% *	20.07% *	20.19% *	124.9
		1.67	0.95	13.98% *	14.29% *	14.93% *	86.39
			∞	14.1% *	14.32% *	14.77% *	65.5
	0.5	1	0.95	______	______	______	______
			∞	8.65% *	9.25% *	9.56% *	90.22
		1.33	0.95	9.35% *	9.69% *	10.19% *	85.09
			∞	8.43% *	9.1% *	9.54% *	82.08
		1.67	0.95	______	______	______	______
			∞	22.24% *	23.07% *	23.61% *	91.52
rat195	0.2	1	0.95	______	______	______	______
			∞	4.39% *	5.53% *	6.19% *	544.14
		1.33	0.95	______	______	______	______
			∞	2.78% *	3.09% *	3.31% *	247.94
		1.67	0.95	3.09% *	3.56% *	4.26% *	243.03
			∞	2.82% *	2.9% *	2.96% *	219.16
	0.35	1	0.95	______	______	______	______
			∞	1.29% *	2.41% *	3.06% *	296.49
		1.33	0.95	2.27% *	2.31% *	2.4% *	261.0
			∞	2.45% *	2.84% *	3.08% *	217.0
		1.67	0.95	______	______	______	______
			∞	2.2% *	2.56% *	3.09% *	240.35
	0.5	1	0.95	______	______	______	______
			∞	2.07% *	2.73% *	3.17% *	302.52
		1.33	0.95	1.28% *	1.63% *	1.85% *	422.47
			∞	2.22% *	2.32% *	2.49% *	316.6

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
rat195	0.5	1.67	0.95	1.92% *	2.49% *	2.81% *	230.01
			∞	2.0%	2.36%	3.02%	243.94
pr226	0.2	1	0.95	22.56% *	23.44% *	25.1% *	628.43
			∞	22.01% *	22.83% *	23.83% *	749.82
		1.33	0.95	—	—	—	—
			∞	11.23% *	11.51% *	12.04% *	250.9
		1.67	0.95	6.72% *	6.92% *	7.28% *	304.72
			∞	6.87% *	7.06% *	7.41% *	258.09
	0.35	1	0.95	17.93% *	17.94% *	17.96% *	361.73
			∞	17.67% *	17.88% *	18.25% *	265.2
		1.33	0.95	—	—	—	—
			∞	10.55% *	10.72% *	10.97% *	314.51
		1.67	0.95	—	—	—	—
			∞	8.77% *	8.79% *	8.81% *	247.12
	0.5	1	0.95	14.21% *	16.27% *	20.17% *	1002.22
			∞	16.56% *	18.98% *	21.41% *	1047.23
		1.33	0.95	11.03% *	12.66% *	15.9% *	296.81
			∞	10.51% *	13.3% *	18.12% *	348.91
		1.67	0.95	—	—	—	—
			∞	10.52% *	10.54% *	10.58% *	411.64

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	11.64% *	12.16% *	13.15% *	33.85
			∞	14.85% *	15.97% *	17.81% *	24.1
		1.33	0.95	0.74%	2.07%	4.45%	13.93
			∞	2.55%	4.45%	6.21%	13.82
		1.67	0.95	0.14%	0.98%	2.65%	9.21
			∞	0.14%	3.23%	4.81%	6.75
	0.35	1	0.95	0.0%	5.43%	10.16%	11.01
			∞	1.55%	2.8%	4.66%	16.46
		1.33	0.95	2.22%	6.14%	11.15%	11.54
			∞	0.85%	2.7%	4.7%	18.52
		1.67	0.95	0.0%	2.18%	5.6%	9.82
			∞	0.0%	0.91%	1.78%	6.57
	0.5	1	0.95	0.0%	0.04%	0.06%	10.66
			∞	3.36%	4.09%	5.56%	10.04
		1.33	0.95	0.06%	1.11%	1.71%	5.53
			∞	0.06%	0.89%	1.56%	8.62
		1.67	0.95	0.06%	1.61%	2.39%	8.94
			∞	1.03%	1.21%	1.56%	11.39

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.2	1	0.95	—	—	—	—
			∞	2.6%	3.65%	5.74%	18.93
		1.33	0.95	—	—	—	—
			∞	1.94%	2.79%	3.4%	14.57
		1.67	0.95	3.37%	3.5%	3.7%	15.69
			∞	0.6%	2.12%	3.37%	13.49
	0.35	1	0.95	0.05%	5.81%	9.05%	16.61
			∞	0.05%	2.24%	4.82%	19.35
		1.33	0.95	0.22%	2.49%	4.39%	11.31
			∞	0.22%	1.81%	2.87%	12.95
		1.67	0.95	1.47%	2.19%	3.26%	8.92
			∞	0.87%	2.61%	5.0%	8.32
	0.5	1	0.95	1.03%	2.02%	2.54%	22.96
			∞	0.38%	2.72%	4.54%	10.78
		1.33	0.95	2.01%	3.46%	4.61%	12.79
			∞	2.06%	2.97%	4.78%	6.59
		1.67	0.95	0.33%	0.54%	0.87%	10.21
			∞	0.33%	0.54%	0.98%	8.54
ft70	0.2	1	0.95	4.52% *	5.06% *	5.89% *	41.16
			∞	4.46% *	5.05% *	5.6% *	35.22
		1.33	0.95	3.5%	4.18%	4.74%	31.56
			∞	1.61%	2.96%	4.59%	35.15
		1.67	0.95	1.84%	2.6%	3.24%	67.5
			∞	1.91%	3.08%	3.76%	33.37
	0.35	1	0.95	2.13%	2.32%	2.53%	51.03
			∞	2.82%	3.07%	3.34%	31.0
		1.33	0.95	2.32%	2.85%	3.59%	24.11
			∞	1.8%	2.19%	2.58%	45.34
		1.67	0.95	—	—	—	—
			∞	2.31%	2.89%	3.27%	35.11
	0.5	1	0.95	—	—	—	—
			∞	1.99%	2.64%	3.06%	31.44
		1.33	0.95	—	—	—	—
			∞	2.38%	2.63%	3.07%	37.31
		1.67	0.95	—	—	—	—
			∞	1.22%	2.04%	3.27%	30.78
kro124p	0.2	1	0.95	—	—	—	—
			∞	8.24% *	10.36% *	11.49% *	99.49
		1.33	0.95	5.19%	6.16%	6.7%	57.94
			∞	3.25%	4.56%	6.46%	84.95
		1.67	0.95	4.21%	6.16%	8.66%	57.1

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.2	1.67	∞	2.9%	4.7%	7.05%	68.53
	0.35	1	0.95	—	—	—	—
			∞	5.9% *	6.8% *	8.31% *	173.9
		1.33	0.95	1.84%	3.44%	5.62%	74.42
			∞	3.32%	4.85%	5.98%	45.31
		1.67	0.95	5.29%	5.68%	6.35%	67.28
			∞	4.58%	5.74%	6.39%	67.04
	0.5	1	0.95	—	—	—	—
			∞	7.33% *	8.04% *	9.43% *	127.05
		1.33	0.95	5.1%	6.49%	7.88%	85.02
			∞	7.61%	9.21%	10.47%	64.97
		1.67	0.95	4.88%	5.85%	6.41%	56.27
			∞	5.2%	6.98%	9.19%	61.19
ftv170	0.2	1	0.95	—	—	—	—
			∞	7.91% *	14.59% *	20.13% *	188.1
		1.33	0.95	4.46%	7.34%	10.88%	95.02
			∞	5.08%	6.5%	8.34%	96.64
		1.67	0.95	3.08%	5.6%	7.25%	114.8
			∞	7.54%	8.58%	10.26%	68.69
	0.35	1	0.95	—	—	—	—
			∞	6.9% *	8.38% *	9.95% *	126.99
		1.33	0.95	—	—	—	—
			∞	5.19%	8.2%	10.6%	105.41
		1.67	0.95	4.14%	5.84%	7.26%	75.22
			∞	9.58%	12.08%	14.45%	94.93
	0.5	1	0.95	—	—	—	—
			∞	6.19% *	6.6% *	7.39% *	181.21
		1.33	0.95	—	—	—	—
			∞	5.89%	8.06%	10.77%	78.6
		1.67	0.95	3.48%	6.36%	9.07%	77.97
			∞	4.61%	8.25%	12.09%	77.13

AFI as the constructive heuristic

Table B.17: Computational results, for $\omega = 1/2$ and $MaxIt = 5000$, using the solution obtained through the AFI heuristic as the initial solution.

Symmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.2	1	0.95	—	—	—	—
			∞	4.49% *	4.49% *	4.49% *	11.83
		1.33	0.95	4.4% *	4.41% *	4.45% *	10.59
			∞	5.24% *	5.37% *	5.43% *	14.54
		1.67	0.95	0.39%	1.07%	1.99%	17.18
			∞	1.99%	1.99%	1.99%	8.39
	0.35	1	0.95	0.0%	0.08%	0.23%	9.9
			∞	0.0%	0.0%	0.0%	10.66
		1.33	0.95	0.0%	0.0%	0.0%	10.89
			∞	0.0%	0.79%	2.36%	9.17
		1.67	0.95	0.0%	0.0%	0.0%	10.05
			∞	0.0%	0.0%	0.0%	9.78
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	8.62
		1.33	0.95	0.0%	0.91%	1.39%	14.26
			∞	0.29%	0.29%	0.29%	6.54
		1.67	0.95	0.0%	0.89%	1.33%	9.06
			∞	0.0%	0.36%	0.56%	9.6
pr76	0.2	1	0.95	—	—	—	—
			∞	1.45% *	1.68% *	2.13% *	24.49
		1.33	0.95	1.03% *	1.04% *	1.05% *	27.87
			∞	1.04% *	1.78% *	3.25% *	22.23
		1.67	0.95	0.39% *	0.43% *	0.5% *	18.1
			∞	0.33% *	0.37% *	0.44% *	17.44
	0.35	1	0.95	—	—	—	—
			∞	4.13% *	4.13% *	4.13% *	20.23
		1.33	0.95	0.0%	0.03%	0.08%	13.92
			∞	0.0%	0.07%	0.21%	15.74
		1.67	0.95	0.0%	0.0%	0.0%	19.9
			∞	0.0%	0.0%	0.0%	22.13
	0.5	1	0.95	—	—	—	—
			∞	0.0%	0.0%	0.01%	28.76
		1.33	0.95	0.0%	1.63%	2.74%	28.43
			∞	0.0%	1.83%	2.74%	27.09

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr76	0.5	1.67	0.95	—	—	—	—
			∞	0.0%	0.0%	0.0%	19.11
kroA100	0.2	1	0.95	—	—	—	—
			∞	8.76% *	9.27% *	10.28% *	130.76
		1.33	0.95	0.0%	0.0%	0.0%	36.02
			∞	0.0%	0.0%	0.0%	33.85
		1.67	0.95	0.0%	0.0%	0.0%	30.89
			∞	0.0%	0.0%	0.0%	27.22
	0.35	1	0.95	—	—	—	—
			∞	6.69% *	6.69% *	6.69% *	118.11
		1.33	0.95	2.44% *	3.24% *	4.5% *	79.92
			∞	1.56% *	2.24% *	3.6% *	86.32
		1.67	0.95	0.0%	0.0%	0.0%	29.23
			∞	0.0%	0.0%	0.0%	32.08
	0.5	1	0.95	—	—	—	—
			∞	2.69% *	2.96% *	3.32% *	115.64
		1.33	0.95	1.76% *	1.76% *	1.76% *	105.72
			∞	1.78% *	1.78% *	1.79% *	63.89
		1.67	0.95	0.0%	0.0%	0.0%	31.04
			∞	0.0%	0.0%	0.0%	35.56
pr124	0.2	1	0.95	—	—	—	—
			∞	6.06% *	6.86% *	8.45% *	51.8
		1.33	0.95	9.37% *	9.42% *	9.44% *	51.72
			∞	9.3% *	9.8% *	10.79% *	39.94
		1.67	0.95	10.86% *	10.91% *	10.93% *	54.92
			∞	11.0% *	11.52% *	12.57% *	38.14
	0.35	1	0.95	—	—	—	—
			∞	8.4% *	8.4% *	8.4% *	35.35
		1.33	0.95	—	—	—	—
			∞	0.0%	0.2%	0.6%	52.88
		1.67	0.95	6.69% *	6.69% *	6.69% *	83.61
			∞	5.92% *	6.13% *	6.56% *	47.74
	0.5	1	0.95	—	—	—	—
			∞	4.51% *	4.97% *	5.27% *	75.4
		1.33	0.95	1.98% *	1.98% *	1.98% *	76.16
			∞	4.33% *	4.33% *	4.33% *	62.99
		1.67	0.95	0.0%	0.2%	0.6%	57.93
			∞	0.0%	0.0%	0.0%	49.01
pr152	0.2	1	0.95	—	—	—	—
			∞	13.98% *	14.16% *	14.48% *	186.73
		1.33	0.95	13.21% *	13.94% *	14.89% *	231.68

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr152	0.2	1.33	∞	12.76% *	13.03% *	13.17% *	227.31
		1.67	0.95	13.1% *	13.1% *	13.1% *	76.76
			∞	12.45% *	12.62% *	12.98% *	76.83
	0.35	1	0.95	—	—	—	—
			∞	23.25% *	24.02% *	24.44% *	139.16
		1.33	0.95	18.38% *	18.59% *	19.03% *	116.99
			∞	19.94% *	20.32% *	20.51% *	98.56
		1.67	0.95	—	—	—	—
			∞	14.1% *	14.52% *	14.73% *	99.19
	0.5	1	0.95	—	—	—	—
			∞	8.65% *	9.05% *	9.25% *	112.68
		1.33	0.95	8.44% *	8.78% *	9.24% *	71.44
			∞	9.03% *	9.22% *	9.32% *	82.66
		1.67	0.95	—	—	—	—
			∞	22.24% *	22.49% *	23.01% *	102.05
rat195	0.2	1	0.95	—	—	—	—
			∞	5.65% *	6.41% *	7.35% *	451.02
		1.33	0.95	—	—	—	—
			∞	3.72% *	4.3% *	4.7% *	312.2
		1.67	0.95	3.05% *	4.09% *	5.51% *	219.02
			∞	3.05% *	3.73% *	4.43% *	253.73
	0.35	1	0.95	—	—	—	—
			∞	2.66% *	3.25% *	3.68% *	366.06
		1.33	0.95	2.14% *	2.89% *	4.32% *	428.57
			∞	3.92% *	4.84% *	6.24% *	250.37
		1.67	0.95	—	—	—	—
			∞	3.09% *	4.01% *	4.53% *	343.82
	0.5	1	0.95	—	—	—	—
			∞	2.64% *	3.28% *	4.0% *	397.98
		1.33	0.95	1.32% *	2.06% *	3.48% *	509.68
			∞	3.15% *	3.74% *	4.48% *	259.54
		1.67	0.95	2.32% *	2.57% *	3.07% *	505.87
			∞	2.54%	3.13%	4.0%	236.75
pr226	0.2	1	0.95	—	—	—	—
			∞	22.75% *	23.81% *	24.73% *	621.84
		1.33	0.95	9.31% *	9.74% *	10.19% *	392.6
			∞	9.25% *	9.86% *	10.31% *	677.99
		1.67	0.95	6.72% *	6.75% *	6.77% *	251.28
			∞	6.84% *	6.87% *	6.89% *	301.07
	0.35	1	0.95	—	—	—	—
			∞	14.51% *	14.62% *	14.84% *	462.89

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
pr226	0.35	1.33	0.95	11.33% *	11.38% *	11.42% *	301.72
			∞	10.53% *	10.53% *	10.54% *	242.16
		1.67	0.95	9.09% *	9.13% *	9.19% *	396.78
			∞	8.76% *	8.81% *	8.86% *	277.82
	0.5	1	0.95	—	—	—	—
			∞	19.13% *	19.52% *	19.78% *	391.66
		1.33	0.95	10.56% *	10.87% *	11.36% *	371.93
			∞	10.52% *	10.56% *	10.59% *	344.3
		1.67	0.95	10.97% *	11.04% *	11.11% *	333.43
			∞	10.52% *	10.54% *	10.58% *	359.47

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	—	—	—	—
			∞	10.66% *	12.33% *	14.22% *	26.3
		1.33	0.95	2.51%	5.28%	7.87%	14.84
			∞	3.28%	5.26%	6.98%	13.05
		1.67	0.95	0.0%	2.89%	8.66%	17.3
			∞	0.0%	0.0%	0.0%	7.37
	0.35	1	0.95	—	—	—	—
			∞	0.48%	1.74%	4.25%	21.3
		1.33	0.95	1.89%	2.75%	4.13%	27.96
			∞	2.55%	3.26%	3.67%	13.33
		1.67	0.95	0.06%	1.09%	2.26%	9.05
			∞	1.01%	1.74%	2.84%	6.17
	0.5	1	0.95	2.61%	4.52%	7.6%	6.91
			∞	2.61%	3.22%	3.68%	8.21
		1.33	0.95	1.06%	2.55%	3.98%	8.03
			∞	1.06%	1.79%	2.74%	7.29
		1.67	0.95	0.0%	1.0%	1.97%	9.71
			∞	0.06%	0.7%	1.03%	9.22
ftv64	0.2	1	0.95	1.38%	3.1%	6.32%	30.38
			∞	2.76%	5.77%	8.18%	34.03
		1.33	0.95	0.97%	3.72%	6.42%	8.24
			∞	2.53%	3.58%	5.12%	13.25
		1.67	0.95	—	—	—	—
			∞	2.28%	2.28%	2.28%	10.81
	0.35	1	0.95	0.0%	2.82%	5.09%	15.75
			∞	2.6%	4.04%	6.39%	14.29
		1.33	0.95	1.84%	4.53%	8.94%	8.35
			∞	0.05%	1.53%	2.71%	15.21

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv64	0.35	1.67	0.95	1.96%	3.17%	4.46%	7.94
			∞	1.96%	2.94%	3.53%	11.62
	0.5	1	0.95	———	———	———	———
			∞	1.19%	2.27%	3.62%	11.64
		1.33	0.95	0.81%	1.66%	3.37%	8.23
			∞	1.25%	1.59%	2.06%	13.09
		1.67	0.95	———	———	———	———
			∞	0.98%	1.18%	1.47%	8.74
ft70	0.2	1	0.95	4.29% *	5.09% *	6.15% *	38.99
			∞	4.22% *	5.22% *	6.27% *	53.44
		1.33	0.95	2.49%	2.92%	3.57%	39.71
			∞	1.91%	3.1%	4.28%	30.38
		1.67	0.95	2.43%	3.45%	5.02%	48.01
			∞	1.61%	2.0%	2.62%	32.1
	0.35	1	0.95	2.12%	2.47%	2.67%	27.94
			∞	2.21%	2.47%	2.8%	30.4
		1.33	0.95	1.68%	2.08%	2.36%	35.52
			∞	1.54%	2.46%	2.92%	33.76
		1.67	0.95	1.76%	2.18%	2.88%	39.23
			∞	1.9%	2.91%	4.06%	28.8
	0.5	1	0.95	1.4%	1.93%	2.31%	56.79
			∞	1.97%	2.9%	4.27%	35.09
		1.33	0.95	2.15%	3.2%	4.27%	45.53
			∞	1.97%	2.29%	2.54%	24.43
		1.67	0.95	1.11%	1.95%	2.86%	39.22
			∞	2.27%	2.92%	3.88%	24.18
kro124p	0.2	1	0.95	5.05% *	5.46% *	6.23% *	103.69
			∞	3.81% *	4.89% *	5.71% *	145.88
		1.33	0.95	0.15%	2.38%	4.19%	53.75
			∞	0.33%	1.77%	4.2%	52.33
		1.67	0.95	0.73%	0.95%	1.29%	32.35
			∞	0.3%	0.44%	0.67%	48.93
	0.35	1	0.95	———	———	———	———
			∞	9.11% *	10.66% *	12.63% *	89.61
		1.33	0.95	0.83%	2.15%	2.97%	39.56
			∞	1.22%	1.54%	1.81%	46.39
		1.67	0.95	0.53%	1.24%	2.31%	55.25
			∞	0.26%	1.83%	3.94%	43.22
	0.5	1	0.95	3.62% *	4.56% *	6.27% *	61.44
			∞	4.25% *	4.69% *	4.92% *	70.91
		1.33	0.95	2.59%	3.7%	5.12%	67.48

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
kro124p	0.5	1.33	∞	2.21%	2.73%	3.23%	67.74
		1.67	0.95	2.53%	3.07%	3.46%	50.21
			∞	2.55%	2.81%	3.27%	36.08
ftv170	0.2	1	0.95	———	———	———	———
			∞	4.81% *	8.36% *	10.87% *	220.22
		1.33	0.95	1.74%	3.26%	4.35%	106.3
			∞	2.94%	5.05%	6.35%	78.19
		1.67	0.95	3.3%	4.64%	5.37%	78.07
			∞	2.5%	4.25%	5.47%	143.57
	0.35	1	0.95	5.54% *	8.25% *	11.45% *	147.67
			∞	5.99% *	9.95% *	13.29% *	158.38
		1.33	0.95	4.28%	5.83%	6.9%	126.06
			∞	2.98%	3.67%	4.83%	105.53
		1.67	0.95	3.01%	3.88%	5.59%	85.24
			∞	3.05%	5.58%	7.66%	90.22
	0.5	1	0.95	4.64% *	7.31% *	9.14% *	185.31
			∞	5.68% *	11.44% *	18.27% *	138.55
		1.33	0.95	4.99%	6.63%	9.87%	91.87
			∞	8.28%	10.17%	12.44%	93.29
		1.67	0.95	2.11%	2.53%	3.3%	105.83
			∞	1.2%	2.52%	3.56%	99.97

ARI as the constructive heuristic

Table B.18: Computational results, for $\omega = 1/2$ and $MaxIt = 5000$, using the solution obtained through the ARI heuristic as the initial solution.

Symmetric instances

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.2	1	0.95	———	———	———	———
			∞	4.49% *	4.49% *	4.49% *	32.19
		1.33	0.95	4.4% *	4.41% *	4.45% *	16.76
			∞	5.24% *	5.35% *	5.43% *	29.99
		1.67	0.95	0.0%	0.13%	0.39%	34.82
			∞	1.99%	1.99%	1.99%	15.87
	0.35	1	0.95	0.0%	0.11%	0.23%	22.94
			∞	0.0%	0.08%	0.23%	23.96
		1.33	0.95	0.0%	0.98%	2.93%	30.09
			∞	2.36%	2.55%	2.93%	16.88

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
berlin52	0.35	1.67	0.95	0.0%	0.0%	0.0%	17.29
			∞	0.0%	0.0%	0.0%	20.68
	0.5	1	0.95	———	———	———	———
			∞	0.0%	0.36%	1.08%	17.43
		1.33	0.95	0.0%	0.65%	1.34%	33.73
			∞	0.29%	0.29%	0.29%	11.9
		1.67	0.95	0.0%	0.61%	1.33%	17.45
			∞	0.0%	0.44%	1.33%	18.19
pr76	0.2	1	0.95	———	———	———	———
			∞	1.45% *	1.79% *	1.95% *	49.17
		1.33	0.95	1.05% *	1.3% *	1.54% *	37.51
			∞	1.04% *	1.18% *	1.48% *	59.03
		1.67	0.95	0.58% *	0.61% *	0.67% *	34.02
			∞	0.61% *	0.61% *	0.61% *	42.13
	0.35	1	0.95	———	———	———	———
			∞	4.13% *	4.13% *	4.13% *	44.11
		1.33	0.95	0.0%	0.07%	0.21%	25.87
			∞	0.0%	0.0%	0.01%	24.51
		1.67	0.95	0.0%	0.63%	0.95%	31.9
			∞	0.0%	0.27%	0.82%	29.59
	0.5	1	0.95	0.0%	0.0%	0.0%	54.68
			∞	0.0%	0.0%	0.0%	37.34
		1.33	0.95	0.01%	0.72%	2.13%	39.38
			∞	0.0%	0.0%	0.0%	30.04
		1.67	0.95	0.0%	0.3%	0.59%	52.83
			∞	0.0%	0.31%	0.86%	35.88
kroA100	0.2	1	0.95	———	———	———	———
			∞	8.25% *	8.89% *	9.32% *	272.76
		1.33	0.95	0.0%	0.0%	0.0%	57.31
			∞	0.0%	0.0%	0.0%	63.46
		1.67	0.95	0.0%	0.03%	0.09%	58.07
			∞	0.0%	0.03%	0.09%	57.38
	0.35	1	0.95	———	———	———	———
			∞	6.33% *	6.33% *	6.33% *	167.46
		1.33	0.95	2.44% *	2.44% *	2.44% *	169.48
			∞	1.56% *	2.7% *	3.6% *	124.93
		1.67	0.95	0.0%	0.03%	0.09%	68.36
			∞	0.0%	0.03%	0.09%	60.99
	0.5	1	0.95	———	———	———	———
			∞	2.41% *	2.5% *	2.58% *	228.7
		1.33	0.95	1.76% *	1.76% *	1.76% *	189.73

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
kroA100	0.5	1.33	∞	1.78% *	1.78% *	1.78% *	169.45
		1.67	0.95	0.0%	0.0%	0.0%	60.53
			∞	0.0%	0.0%	0.0%	61.74
pr124	0.2	1	0.95	—	—	—	—
			∞	5.04% *	5.88% *	6.54% *	107.24
		1.33	0.95	9.44% *	9.44% *	9.44% *	82.96
			∞	9.3% *	9.3% *	9.3% *	83.1
		1.67	0.95	10.93% *	11.11% *	11.47% *	71.15
			∞	11.0% *	11.5% *	12.5% *	71.2
	0.35	1	0.95	8.73% *	8.73% *	8.73% *	68.25
			∞	8.4% *	8.4% *	8.4% *	66.95
		1.33	0.95	0.6%	0.6%	0.6%	67.68
			∞	0.0%	0.02%	0.07%	85.78
		1.67	0.95	6.69% *	6.87% *	7.25% *	123.21
			∞	5.92% *	6.13% *	6.56% *	96.66
	0.5	1	0.95	—	—	—	—
			∞	4.32% *	4.64% *	5.27% *	173.49
		1.33	0.95	1.98% *	2.95% *	3.44% *	87.76
			∞	4.33% *	4.83% *	5.82% *	75.47
		1.67	0.95	0.0%	0.0%	0.0%	72.81
			∞	0.0%	0.48%	1.44%	78.28
pr152	0.2	1	0.95	13.43% *	13.43% *	13.43% *	652.38
			∞	14.33% *	14.86% *	15.69% *	296.29
		1.33	0.95	13.13% *	13.13% *	13.13% *	274.61
			∞	13.15% *	13.3% *	13.54% *	279.43
		1.67	0.95	12.57% *	12.82% *	13.07% *	229.96
			∞	12.36% *	13.01% *	14.0% *	215.15
	0.35	1	0.95	23.31% *	23.47% *	23.64% *	220.23
			∞	23.4% *	24.48% *	25.39% *	269.62
		1.33	0.95	18.38% *	19.04% *	19.36% *	156.53
			∞	19.85% *	20.1% *	20.6% *	108.77
		1.67	0.95	14.72% *	14.72% *	14.72% *	121.77
			∞	14.1% *	14.42% *	14.73% *	145.13
	0.5	1	0.95	—	—	—	—
			∞	8.65% *	11.06% *	14.9% *	234.11
		1.33	0.95	8.44% *	9.14% *	9.64% *	116.41
			∞	8.43% *	8.73% *	9.03% *	129.7
		1.67	0.95	—	—	—	—
			∞	22.24% *	22.83% *	23.36% *	146.76
rat195	0.2	1	0.95	—	—	—	—
			∞	5.78% *	7.03% *	8.79% *	966.38

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
rat195	0.2	1.33	0.95	—	—	—	—
			∞	3.31% *	4.22% *	5.24% *	582.64
		1.67	0.95	2.96% *	3.05% *	3.18% *	537.51
			∞	2.24% *	3.9% *	5.02% *	617.38
	0.35	1	0.95	—	—	—	—
			∞	1.91% *	2.17% *	2.53% *	779.2
		1.33	0.95	2.23% *	2.92% *	3.78% *	520.69
			∞	3.08% *	3.46% *	4.24% *	761.13
		1.67	0.95	3.82% *	3.82% *	3.82% *	738.46
			∞	2.74% *	4.04% *	4.84% *	631.08
	0.5	1	0.95	—	—	—	—
			∞	1.1% *	1.41% *	1.58% *	722.28
		1.33	0.95	1.94% *	2.22% *	2.51% *	844.75
			∞	2.0% *	2.63% *	3.77% *	458.0
		1.67	0.95	2.58% *	3.22% *	3.88% *	504.93
			∞	1.87%	2.45%	3.16%	523.7
pr226	0.2	1	0.95	20.93% *	22.68% *	25.1% *	2439.54
			∞	25.76% *	26.93% *	27.82% *	1234.05
		1.33	0.95	9.17% *	9.6% *	9.81% *	662.38
			∞	9.14% *	9.35% *	9.68% *	736.6
		1.67	0.95	6.71% *	6.84% *	7.03% *	375.8
			∞	6.84% *	6.94% *	7.0% *	623.46
	0.35	1	0.95	14.21% *	15.96% *	17.72% *	951.22
			∞	14.56% *	16.77% *	21.07% *	960.84
		1.33	0.95	11.33% *	11.67% *	12.29% *	504.8
			∞	10.47% *	10.48% *	10.48% *	524.06
		1.67	0.95	9.12% *	9.12% *	9.12% *	553.3
			∞	8.77% *	9.83% *	11.83% *	547.87
	0.5	1	0.95	—	—	—	—
			∞	19.1% *	19.45% *	20.1% *	1069.6
		1.33	0.95	10.46% *	10.84% *	11.15% *	969.31
			∞	10.58% *	11.88% *	14.49% *	486.41
		1.67	0.95	11.03% *	11.3% *	11.7% *	662.47
			∞	10.53% *	11.13% *	11.74% *	647.0

Asymmetric instances

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1	0.95	13.27% *	16.39% *	20.23% *	57.9
			∞	11.44% *	13.19% *	14.55% *	77.84
		1.33	0.95	0.0%	1.6%	4.8%	37.15
			∞	4.64%	6.38%	8.41%	22.49

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ft53	0.2	1.67	0.95	0.14%	0.75%	1.41%	27.02
			∞	0.14%	0.77%	1.47%	20.76
	0.35	1	0.95	0.85%	2.05%	3.25%	39.65
			∞	2.6%	3.22%	3.7%	26.65
		1.33	0.95	1.72%	4.67%	7.66%	46.15
			∞	0.85%	1.01%	1.27%	54.17
		1.67	0.95	0.94%	2.03%	4.2%	19.19
			∞	0.0%	2.63%	4.97%	19.03
	0.5	1	0.95	0.06%	3.89%	7.72%	16.18
			∞	0.0%	3.18%	6.26%	19.04
		1.33	0.95	0.06%	1.11%	1.71%	15.07
			∞	0.0%	0.82%	2.4%	17.76
		1.67	0.95	1.03%	3.61%	6.52%	9.65
			∞	0.0%	1.28%	3.84%	14.71
ftv64	0.2	1	0.95	4.62%	5.93%	6.8%	25.59
			∞	6.37%	7.42%	8.71%	46.52
		1.33	0.95	2.75%	3.38%	3.88%	28.92
			∞	1.19%	2.93%	5.44%	23.8
		1.67	0.95	0.6%	2.41%	3.75%	27.47
			∞	2.23%	2.27%	2.28%	22.32
	0.35	1	0.95	3.14%	3.86%	4.28%	26.24
			∞	0.98%	2.24%	2.98%	24.17
		1.33	0.95	0.22%	2.56%	4.6%	29.83
			∞	1.84%	2.89%	3.9%	23.47
		1.67	0.95	3.26%	4.64%	5.87%	14.52
			∞	1.96%	3.63%	4.89%	15.46
	0.5	1	0.95	0.43%	3.08%	4.86%	44.11
			∞	0.27%	1.59%	3.95%	37.29
		1.33	0.95	0.0%	1.56%	3.85%	23.12
			∞	0.81%	0.89%	0.98%	19.83
		1.67	0.95	0.87%	2.86%	3.85%	14.46
			∞	0.81%	0.98%	1.14%	16.17
ft70	0.2	1	0.95	5.17% *	5.55% *	5.88% *	58.17
			∞	5.05% *	5.22% *	5.33% *	88.94
		1.33	0.95	3.0%	3.76%	4.22%	82.37
			∞	2.1%	3.01%	3.52%	57.18
		1.67	0.95	2.65%	3.64%	5.03%	103.87
			∞	2.16%	2.2%	2.26%	65.32
	0.35	1	0.95	1.52%	1.9%	2.3%	90.67
			∞	1.73%	2.63%	3.9%	81.71
		1.33	0.95	1.75%	2.7%	3.48%	37.61

Matrix name	α	β	γ	Minimum Gap	Average Gap	Maximum Gap	Average CPU time (seconds)
ft70	0.35	1.33	∞	1.83%	2.07%	2.32%	46.57
		1.67	0.95	2.1%	2.77%	3.35%	63.73
			∞	1.88%	2.71%	3.27%	54.39
	0.5	1	0.95	2.52%	2.61%	2.7%	88.48
			∞	2.58%	2.93%	3.22%	53.89
		1.33	0.95	2.4%	2.79%	3.18%	53.34
			∞	1.9%	2.28%	2.68%	55.25
		1.67	0.95	1.84%	2.29%	3.08%	64.35
			∞	2.11%	3.08%	4.07%	48.46
kro124p	0.2	1	0.95	6.81% *	7.95% *	9.47% *	122.26
			∞	5.05% *	6.08% *	6.78% *	187.51
		1.33	0.95	1.78%	5.12%	7.6%	69.63
			∞	4.28%	4.83%	5.27%	82.35
		1.67	0.95	0.8%	1.77%	3.05%	84.82
			∞	0.63%	1.65%	3.65%	105.35
	0.35	1	0.95	—	—	—	—
			∞	5.86% *	8.45% *	11.15% *	271.6
		1.33	0.95	1.49%	1.84%	2.4%	120.65
			∞	0.36%	1.11%	2.13%	97.34
		1.67	0.95	0.35%	0.51%	0.61%	116.81
			∞	0.15%	0.99%	1.57%	60.89
	0.5	1	0.95	3.15% *	4.21% *	4.77% *	216.52
			∞	1.73% *	3.18% *	5.8% *	172.8
		1.33	0.95	1.15%	1.8%	2.68%	112.52
			∞	0.46%	1.6%	2.83%	130.99
		1.67	0.95	0.35%	3.6%	6.85%	124.9
			∞	0.63%	1.56%	3.27%	90.43
ftv170	0.2	1	0.95	10.82% *	13.88% *	16.93% *	449.9
			∞	9.63% *	14.15% *	17.36% *	323.29
		1.33	0.95	5.33%	6.85%	8.67%	200.74
			∞	3.7%	6.22%	10.62%	134.36
		1.67	0.95	5.33%	5.81%	6.35%	175.81
			∞	2.54%	4.09%	5.69%	194.31
	0.35	1	0.95	9.7% *	10.1% *	10.32% *	326.3
			∞	8.02% *	10.14% *	11.4% *	485.49
		1.33	0.95	5.34%	5.8%	6.24%	186.95
			∞	6.86%	9.32%	11.65%	166.46
		1.67	0.95	6.06%	7.94%	9.11%	126.08
			∞	5.63%	7.36%	10.45%	162.96
	0.5	1	0.95	5.44% *	9.65% *	13.86% *	320.53
			∞	8.99% *	9.73% *	10.27% *	263.53

Matrix name	α	β	γ	**Minimum Gap**	**Average Gap**	**Maximum Gap**	**Average CPU time (seconds)**
ftv170	0.5	1.33	0.95	6.36%	7.81%	8.57%	198.44
			∞	3.72%	5.4%	7.19%	285.81
		1.67	0.95	5.92%	7.51%	9.4%	249.9
			∞	2.83%	5.35%	9.0%	242.04