ON UNIDIRECTIONAL CYCLIC LAYOUTS, HAMILTONIAN CIRCUITS, CAPACITATED VEHICLE ROUTES AND MINIMAL SPANNING TREES

by

Temel Öncan

B.S., Industrial Engineering, İstanbul Technical University, 1996

M.S., Industrial Engineering, Boğaziçi University, 1998

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering
Boğaziçi University

2004

ON UNIDIRECTIONAL CYCLIC LAYOUTS, HAMILTONIAN CIRCUITS, CAPACITATED VEHICLE ROUTES AND MINIMAL SPANNING TREES

APPROVED BY:

Prof. İ. Kuban Altınel (Thesis Supervisor) .....

Assist. Prof. Necati Aras .....

Prof. Gülay Barbarosoğlu .....

Assoc. Prof. A. Tamer Ünal .....

Prof. Tülin Yazgaç .....

DATE OF APPROVAL: 9.9.2004

Patience et longueur du temps font
plus que force ni que rage.
La Fontaine

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude and sincere appreciation to my thesis advisor and mentor, Professor İ. Kuban Altınel for his invaluable guidance, patience and encouragement. Absolutely, I would not have been able to accomplish this work without him. All of his critics, suggestions, comments and corrections, which have finely shaped my character, gave me the motivation to excel in what I am doing despite the inevitable difficulties. His broad knowledge, research enthusiasm and deep insights have been a very important resource and inspiration to my accomplishment. I am very proud of being his student and, I thank him for all his unforgettable contributions to my life and for his generously bestowed helps. I hope to continue learning from his spirit on both the personal and scientific levels throughout my life.

I have been very fortunate to have been given the opportunity to have a thesis evaluation committee with Professor Gülay Barbarosoğlu and Professor Tülin Yazgaç. I can not thank them enough for their encouraging and constructive suggestions since September 2000. Their feedback and helpful comments have been invaluable. A tremendous debt of gratitude is owned to other members of my dissertation committee particularly to Assistant Professor Necati Aras for his interest and patience, and Associate Professor A. Tamer Ünal for his thoroughness and generosity.

I dedicate this work to my mother Nimet and my father Turgut. They have been a source of wisdom and consolation. Without their invaluable love, sacrifices and patient support, this dissertation would never have been possible. Most importantly I would not have planned to start it. I am definitely indebted to them for what I am today. I owe an incalculable dept of gratitude to my sister Emel and my brother-in-law Tayfun Sarı for their unequivocal affection. My special gratitude extends to my nephew Osman Sarı for accepting me as a friend and making me smile all the time. I am eternally grateful for him. Special thanks also to Aysun Dedeyetimoğlu for her kind helps and supports which were very valuable.

I owe special thanks to Orhan Feyzioğlu, my office mate, for his academic cooperation and devoted friendship which has been and will always be invaluable to me. I would like to thank a lot my dear friend Ş. İlker Birbil, for his helps in both professional and personal perspectives. I wish also to thank my friends Alain Ketterlin, Vladimir Radevski, Burak Aksoy and Burak Parlak, for their encouragements over the past few years.

This research has been partly supported by Boğaziçi Research Fund Grant No: 04HA301D.

ABSTRACT

ON UNIDIRECTIONAL CYCLIC LAYOUTS, HAMILTONIAN CIRCUITS, CAPACITATED VEHICLE ROUTES AND MINIMAL SPANNING TREES

This thesis consists of four major parts. The first part is on the Unidirectional Cyclic Layout Problem (UCLP). First, new efficient heuristics for the UCLP are proposed based on the ideas originally proposed for two well-known combinatorial optimization problems: The Asymmetric Travelling Salesman Problem (ATSP) and the Linear Ordering Problem (LOP). Then, we particularly consider the balanced case of the UCLP's satisfying the additional conservation of flow assumption: the material flow is conserved at every workstation and we develop a Branch and Bound algorithm for the Balanced UCLP (BUCLP). In the second part, we propose new extended ATSP formulations $O(n^3)$ constraints and we analyze the strengths of their linear programming (LP) relaxation both analytically and experimentally. It is shown that the LP relaxation of one of the new formulations can have optimal objective value larger than the LP relaxation of the ATSP's multi-commodity flow formulations. In addition, we also propose new extended ATSP formulations with $O(n^2)$ constraints and compare their strengths with the ones of known ATSP formulations with $O(n^2)$ constraints. Finally, in the third and fourth parts, a new enhancement of the Clarke and Wright's savings heuristic for the Capacitated Vehicle Routing Problem and new enhancements of the Esau and Williams' savings heuristic for the Capacitated Minimal Spanning Tree Problem are respectively proposed.

ÖZET

TEK YÖNLÜ DAİRESEL YERLEŞİMLER, HAMİLTON ÇEVRİMLER, SINIRLI ARAÇ ROTALARI VE EN KÜÇÜK KAPSARAĞAÇLAR ÜZERİNE

Bu tez dört ana kısımdan oluşmaktadır. Birinci kısım, tek yönlü dairesel yerleşim problemi (TYDYP) ile ilgilidir. TYDYP için, aslında Asimetrik Gezgin Satıcı Problemi (AGSP) ve Doğrusal Sıralama Problemi (DSP) için önerilen sezgisel algoritmalardan esinlenerek yeni etkili çözüm yöntemleri önerilmiştir. Daha sonra, TYDYP' nin özel bir durumu olan ve malzeme akışının korunduğu varsayımının yapıldığı, Dengeli TYDYP (DTYDYP) ele alınmış ve bu problem için bir dal-sınır algoritması önerilmiştir. İkinci kısımda, $O(n^3)$ kısıtlı yeni AGSP formülasyonları önerilmiş ve doğrusal programlama (DP) gevşetilmelerinin gücü hem analitik hem de deneysel olarak incelenmiştir. Önerilen yeni formülasyonlardan bir tanesinin DP gevşetilmesiyle elde edilen eniyi değerin bilinen diğer çok mallı AGSP formülasyonlarının DP gevşetilmesiyle elde edilecek değerden daha büyük olduğu gösterilmiştir. Bunlara ilave olarak, $O(n^2)$ kısıtlı yeni AGSP formülasyonları önerilmiş ve DP gevşetilmelerinin gücü bilinen diğer $O(n^2)$ kısıtlı AGSP formülasyonlarının DP gevşetilmeleriyle karşılaştırılmıştır. Son olarak, üçüncü ve dördüncü kısımlarda sırasıyla, Sığa Sınırlı Araç Rotalama Problemi için geliştirilmiş olan Clarke ve Wright kazanım sezgiseli ve Sığa Sınırlı En Küçük Kapsarağaç Problemi için geliştirilmiş olan Esau ve Williams kazanım sezgiseli için yeni iyileştirmeler önerilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

A	Arc set
C	A directed circuit
D	Vehicle capacity
E	Edge set
S	A subset of vertices
V	Vertex set
z	Objective function value
α	Tree shape parameter for the Esau and Williams' expression
β	Second parameter for the Esau and Williams' expression
γ	Demand parameter for the Esau and Williams' expression
δ	Demand parameter proposed by Jothi and Raghavachari
λ	Route shape parameter for the Clarke and Wright's expression
μ	Second parameter for the Clarke and Wright's expression
ν	Demand parameter for the Clarke and Wright's expression
ACVRP	Asymmetric Capacitated Vehicle Routing Problem
AD	Adjacent Dominance rule
AGV	Automated Guided Vehicle
ART	Adaptive Reasoning Technique
AS	Ant System
ATSP	Asymmetric Travelling Salesman Problem
BEUCLP	Balanced Equidistant Unidirectional Cyclic Layout Problem
BILP	Binary Integer Linear Programming
BPP	Bin Packing Problem
BUCLP	Balanced Unidirectional Cyclic Layout Problem
CLAUS	Claus' formulation
CMSTP	Capacitated Minimal Spanning Tree Problem
CNC	Computerized Numerical Control

CP	Circuit Packing
CVRP	Capacitated Vehicle Routing Problem
CW	Clarke and Wright's savings heuristic
CW1	First enhancement of Clarke and Wright's heuristic
CW2	Second enhancement of Clarke and Wright's heuristic
CW3	New enhancement of Clarke and Wright's heuristic
DCVRP	Distance Constrained Capacitated Vehicle Routing Problem
DFJ	Dantzig, Fulkerson and Johnson's formulation
DL	Desrochers and Laporte's formulation
EUCLP	Equidistant Unidirectional Cyclic Layout Problem
EW	Esau and Williams' savings heuristic
EW1	First new enhancement of Esau and Williams' heuristic
EW2	Second new enhancement of Esau and Williams' heuristic
EW3	Third new enhancement of Esau and Williams' heuristic
EX	Exchange heuristic
FB	Esau and Williams' implementation with First Best rule
FCG	Finke, Claus and Gunn's formulation
FGG2	Fox, Gavish and Graves formulation with two constraint sets
FGG3	Fox, Gavish and Graves formulation with three constraint sets
FGG4	Fox, Gavish and Graves formulation with four constraint sets
FMS	Flexible Manufacturing System
FOGA	First Order Greedy Algorithms
GA	Genetic Algorithm
GD	General Dominance rule
GG	Gavish and Graves' formulation
GP1	The first formulation of Gouveia and Pires
GP2	The second formulation of Gouveia and Pires
GP3	The third formulation of Gouveia and Pires
GP4	The fourth formulation of Gouveia and Pires
GP5	The fifth formulation of Gouveia and Pires
GP6	The sixth formulation of Gouveia and Pires

GP7	The seventh formulation of Gouveia and Pires
GP8	The eighth formulation of Gouveia and Pires
GRASP	Greedy Randomized Adaptive Search Procedure
IP	Integer Programming
KK1	Kouvelis and Kim's first construction heuristic
KK2	Kouvelis and Kim's second construction heuristic
KK3	Kouvelis and Kim's third construction heuristic
LANGEVIN	Langevin's formulation
LB	Esau and Williams' implementation with Last Best rule
LHC	Lee, Huang and Chiang's heuristic
LIP	Loop Interconnection Problem
LOP	Linear Ordering Problem
LOULOU	Loulou's formulation
LP	Linear Programming
LUL	Load/ Unload
MCF	Multi-Commodity Flow
MTZ	Miller, Tucker and Zemlin's formulation
MR	Move–Reverse
NEW1	The first new ATSP formulation
NEW2	The second new ATSP formulation
NEW3	The third new ATSP formulation
NEW4	The fourth new ATSP formulation
NEW5	The fifth new ATSP formulation
NEW6	The sixth new ATSP formulation
NEW7	The seventh new ATSP formulation
NN	Neural Networks
NP	Non–deterministically Polynomial
OI	Output–Input
OR	Or-opt heuristic
PQ	Pickard and Queyranne's formulation
Q	Subtree capacity limitation

QAP	Quadratic Assignment Problem
RE	Relocation heuristic
RLT	Reformulation–linearization technique
SA	Simulated Annealing
SCF	Single Commodity Flow
SD	Sherali and Driscoll's formulation
SEW	Sharaiha's Esau and Williams' implementation
SMR	Sort–Move–Reverse
SOGA	Second Order Greedy Algorithms
TCF	Two-Commodity Flow
TDTSP	Time Dependent Travelling Salesman Problem
TS	Tabu Search
TWCVRP	Capacitated Vehicle Routing Problem with Time Windows
UCL	Unidirectional Cyclic Layout
UCLP	Unidirectional Cyclic Layout Problem
WONG	Wong's formulation

1. INTRODUCTION

Today, many businesses in logistics, telecommunication, and manufacturing have a growing need for the optimal use of their resources. Consequently, the utilization of Operations Research and Mathematical Programming techniques have increased in the last decades parallel to the developments of computer systems and information systems. Moreover, the implementation of the modelling and algorithmic tools have also significantly progressed. The proposed models take into account all the characteristics of real-life applications and, the corresponding algorithms produce sufficiently good solutions within acceptable computing times.

In this dissertation, we consider four well-known combinatorial optimization problems from the manufacturing, logistics and telecommunication. These are the Unidirectional Cyclic Layout Problem (UCLP), the Asymmetric Travelling Salesman Problem (ATSP), the Capacitated Vehicle Routing Problem (CVRP) and the Capacitated Minimal Spanning Tree Problem (CMSTP). First, we develop new solution procedures for the UCLP. Second, we propose new formulations for the ATSP. Then, we improve one of the well-known algorithms for the CVRP. Lastly, we propose new enhancements for one of the widely used algorithms for the CMSTP.

1.1. Unidirectional Cyclic Layout Problem

The UCLP is related to the assignment of n workstations to n predetermined locations in a manufacturing cell which consists of a circular material handling. It is shown to be NP-hard by Kouvelis and Kim [1]. Circular material handling systems connect all workstations by passing through each workstation exactly once. The system is assumed to move the materials unidirectionally (e.g. clockwise or counter-clockwise) around the circuit. Typical examples of this type of material handling systems are loop conveyors, tow lines, overhead monorail systems and wire paths of unidirectional automated guided vehicles. One of the workstations serves as the load/unload area; this is where parts enter and leave the manufacturing cell. Each part is routed through

workstations following the sequence specified in its process plan. When a part is processed at one of the workstations, material handling system moves it unidirectionally to the next station pointed in the process plan. If the workstation is occupied, the part is awaited in a buffer until it becomes available. The objective is to determine the assignment of workstations to candidate locations which minimizes total distance travelled by the parts in the manufacturing cell within a unit time. One version of the UCLP is known as the Balanced Unidirectional Cyclic Layout Problem (BUCLP). The balanced case, is particularly relevant to automated manufacturing where no manual interruption is allowed to remove/insert parts from/to the workstations. In other words, the material flow is conserved at each workstation: Total inflow is equal to total outflow at a workstation.

1.2. Asymmetric Travelling Salesman Problem

Any algorithm devised to solve the ATSP tries to answer the following question: Given a set of n cities and possibly asymmetric intercity distances, what is the shortest tour that visits each city exactly once. Even the euclidean travelling salesman problem is known to be NP-Complete [2]. The ATSP is one of the oldest combinatorial problems. According to Punnen [3] the first work published on the ATSP goes back to 1932 by [4] and the name "Travelling Salesman Problem" has been coined by Robinson [5]. The books by Lawler *et al.* [6], Reinelt [7] and Gutin and Punnen [8] summarize various developments on the subject.

1.3. Capacitated Vehicle Routing Problem

The CVRP is the problem concerning the distribution of goods between depots and customers. The classical CVRP can be defined as designing a set of routes with minimum total routing cost for a fleet of m identical vehicles with capacity D which have to serve n customers with nonnegative demand d_i from a depot subject to the following constraints:

- Each customer is served exactly once by exactly one vehicle

- Each route starts and ends at the depot
- The total demand on each route does not exceed vehicle capacity D.

In many instances, additional requirements and assumptions are added to this basic definition in order to model various aspects of real - life distribution problems. For a complete survey see the recent monograph edited by Toth and Vigo [9]. As the ATSP, the CVRP is also a difficult combinatorial optimization problem. It is known to be NP_hard [10].

1.4. Capacitated Minimal Spanning Tree Problem

To define the CMSTP, consider $G = \{V, E\}$ where $V = \{0, 1, ..., n\}$ is the vertex set with 0 as the *central vertex* (root), and $E = \{\{i, j\} : i, j \in V, i \neq j\}$ is the edge set. Each noncentral vertex has a known nonnegative demand d_i. A cost c_{ij}, which is the cost of connecting vertices (terminals) i and j, is associated with edge $\{i, j\}$. The CMSTP is to find a minimal – cost tree which spans over all vertices so that the sum of demands in each subtree obtained by deleting the edges connecting them to the central vertex does not exceed the capacity limitation Q. The subtree containing vertex i is called the *subtree of* i and the set V_i denotes its vertices. The edge connecting the subtree of vertex i is the *gate of* i and has length g_i. When the demands of all terminals are equal to unity, the problem reduces to finding a minimal – cost rooted spanning tree in which each subtree contains at most Q vertices. This unit demand case, which is also known as the homogeneous demand case, is referred as the CMSTP in the literature. In this thesis we are concerned in both the homogeneous and heterogeneous cases and propose new heuristics for the CMSTP in general. The uncapacitated version is the well – known *minimum spanning tree* (MST) problem. The survey paper by Gavish is an excellent resource to learn about the telecommunication network design problems which can be formulated as a CMSTP [11]. It has been shown that even for the unit demand case it is NP-complete when $2 < Q < n/2$ [12].

1.5. Motivations and Contributions

Chapter 1 outlines the UCLP. This problem has several applications in the layout of Flexible Manufacturing Systems (FMS). The determination of workstation locations around a closed loop conveyor system, in the allocation of cutting tools on the sites around a turret, in the positioning of stations around a unidirectional single loop Automated Guided Vehicle (AGV) path are only a few of real-life examples.

Chapter 2 is where we propose new heuristics for the UCLP. We present the relationships between the UCLP and the ATSP and, the Linear Ordering Problem (LOP). By exploring these relations, we adopt for the UCLP several algorithms originally devised for the ATSP and LOP. Then, we improve some of these ideas resulting in new very accurate and fast local improvement heuristics. A comparison of different algorithms is also provided.

In Chapter 3, we propose a branch and bound algorithm for the BUCLP. We use the efficient heuristic developed in Chapter 2, for upper bounding. The LP relaxation of an existing BUCLP formulation is used for lower bounding. A new dominance is also proposed to decrease the size of the enumeration tree.

Chapter 4 reviews entirely the existing ATSP formulations and their comparisons. We present the current state of the classification of the known ATSP formulations. The comparison of the ATSP is an important issue which attracted the attention of several researchers. Ever since the seminal paper by Dantzig, Fulkerson and Johnson [13] there has been many attempts for a perfect formulation of the ATSP. Dantzig, Fulkerson and Johnson have not only solved the ATSP but have also shown that the concept of cutting planes is important for integer linear programs and the complexity of the facet structure of a combinatorial optimization problem can be dealt by using linear programming efficiently. In addition they used, perhaps for the first time, the concept of branch and bound, which shows how the quality of the bound obtained through linear programming relaxations is important.

In general, in integer programming a *good* formulation is of crucial importance to solve the model. Then, the question becomes how different ATSP formulations should be compared. As it is already indicated, ATSP algorithms require a lower bound on the value of the objective function, and the efficiency of the algorithm depend on the sharpness of the bound. In almost all of the cases the lower bound is obtained by solving a linear programming relaxation. Therefore, it is possible to say that given different formulations the formulation whose linear programming relaxation has the largest optimum objective value is the best of them.

A perfect formulation means an ideal description of the ATSP polytope where the linear programming relaxation of it leads us to the optimum. Although such formulations could have been obtained for unrealistically small problems the effort has resulted in different formulations with varying quality as stated by Langevin, Soumis and Desrosiers [14] and Padberg and Sung [15].

Chapter 5 analyzes new extended ATSP formulations with $O(n^3)$ subtour elimination constraints. They are based on an integer programming model developed for the BUCLP by Kıran *et al.* [16]. The strength of new ATSP formulations are analyzed and compared with major compact ATSP formulations presented in Chapter 4. We show that the linear programming relaxation of one of the new formulations can have optimal objective value larger than the one of the linear programming relaxation of ATSP multi-commodity flow formulations. One more outcome of this chapter is another new ATSP formulation with $n(n-1)(n-2)+(n)(n-1)$ subtour elimination constraints only. Different than many other formulations, this formulation interestingly does not require the $2n$ assignment constraints.

In Chapter 6 and Chapter 7, we respectively present a new enhancement of Clarke and Wright's (CW) savings heuristic for the CVRP [17] and new enhancements of the Esau and Williams's (EW) savings heuristic for the CMSTP [18]. Both of the CW and EW heuristics start with an infeasible solution, and based on a saving criterion they quickly reach a feasible solution. CW and EW savings heuristics obtain a final solution starting from a simple solution. For both of the CVRP and CMSTP, several

heuristic solution methods have been proposed. They can be classified as classical heuristics and metaheuristics. Recent developments have shown that metaheuristics outperform classical heuristics. However, they require long computation times and there are difficulties in their parameter calibration and coding phases. This explains the popularity of the savings heuristics for CVRP and CMSTP in practice. Based on the extensive computational experiments we observe that they improve their accuracy considerably without harming their simplicity.

In Chapter 8 we summarize this dissertation, and discuss potential research problems. We also present further research topics.

To summarize, the contribution of this dissertation is three fold. First, we consider the BUCLP and develop new efficient and accurate heuristics for its solution. Second, we explore the relationship of this cyclic layout problem with the ATSP and, propose new ATSP formulations. Then, we compare these new ATSP formulations with the existing ones. Third, we enhance the CW and EW savings heuristics for the CVRP and CMSTP, respectively.

2. THE DESIGN OF OPTIMAL UNIDIRECTIONAL CYCLIC LAYOUTS

2.1. Introduction

Consider a manufacturing cell which consists of a circular material handling system and n workstations assigned to n candidate locations. Circular material handling systems connect all workstations by passing through each workstation exactly once. The system is assumed to move the parts unidirectionally clockwise, or counterclockwise. Typical examples of this type material handling systems are loop conveyors (Figure 2.1), robot arms rotating unidirectionally (Figure 2.2), and unidirectional single loop automated guided vehicles (Figure 2.3). As it can be seen in the figures one of the workstations serves as the load/unload (LUL) area. A common operational policy for circular material handling systems is to require all parts enter and leave the manufacturing cell at the LUL area. Each part is routed through workstations following the sequence specified in its process plan. When a part is processed at one of the workstations, material handling system moves it unidirectionally to the next workstation pointed in the process plan. If the workstation is occupied, the part is awaited in a buffer until it becomes available. The objective is to determine the assignment of workstations to candidate locations which minimizes total transportation cost of the parts in the manufacturing cell within a unit time. This is a layout problem where a layout is an assignment of workstations to locations. According to Afentakis [19], Unidirectional Cyclic Layouts (UCLs) are mostly preferred because of their relative low initial investment costs, high material handling flexibility and their ability of being easily accommodated to future introduction of new parts and process changes.

As underlined by Kouvelis *et al.* [20] the optimal design of its physical layout is crucial for the performance of an FMS. The work by Afentakis [19] is the very first attempt for the design of an UCL in this respect. He gives a binary integer linear programming formulation to determine locations of the workstations which minimize

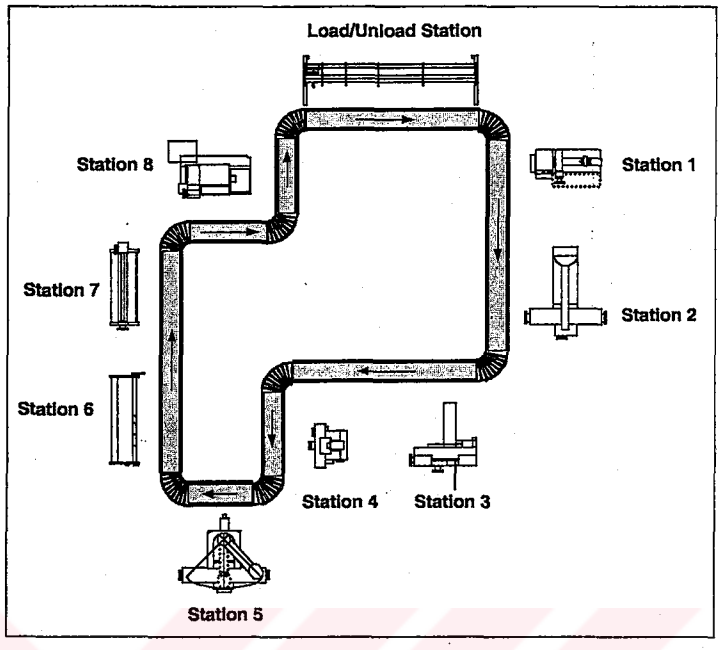

Figure 2.1. Closed loop conveyor system with 9 stations

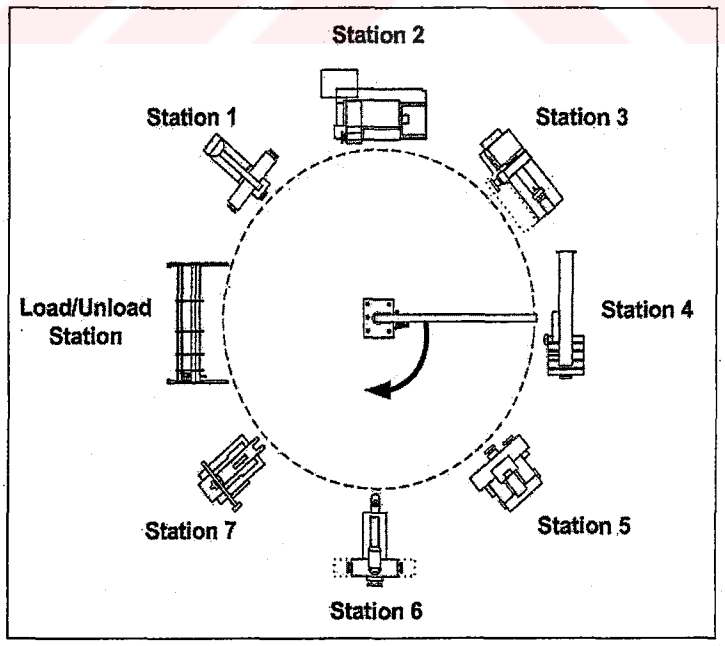

Figure 2.2. Unidirectional robot arm serving 8 stations

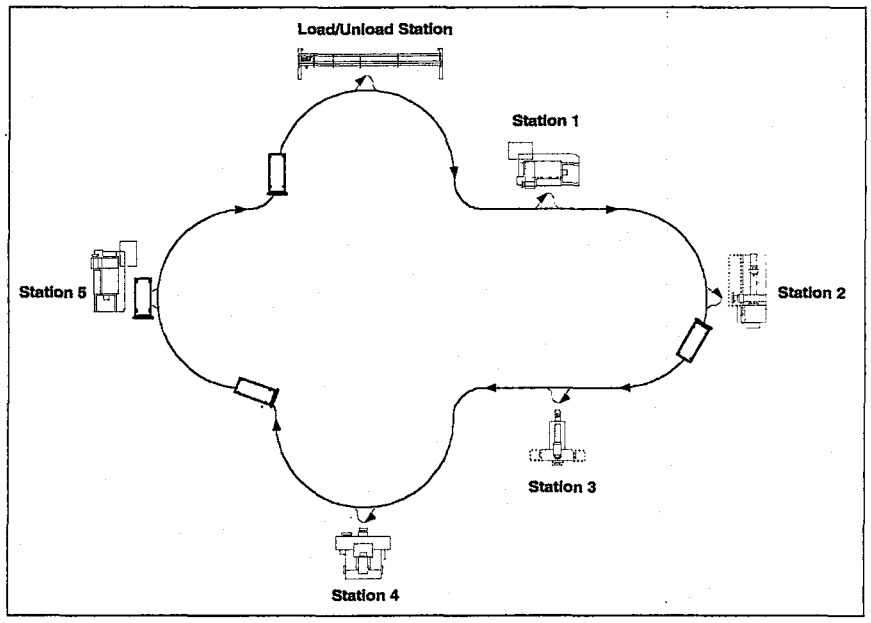

Figure 2.3. AGV system with 6 stations and 4 vehicles

material transport in a UCL. We refer this problem as the UCLP in the sequel. Kouvelis and Kim [1] have shown that the UCLP is NP-hard. A detailed discussion on the complexity of the UCLP is also provided by Tansel and Bilen [21]. UCLP has special forms. In one of them the material flow is conserved at each workstation: Total inflow is equal to total outflow. This version of the UCLP is known as the BUCLP. The BUCLP is particularly relevant in automated manufacturing where no manual interruption is allowed to remove/insert parts from/to the workstations. Another special form of the UCLP is the *Equidistant Unidirectional Cyclic Layout Problem* (EUCLP), where candidate locations served by a unidirectional cyclic material handling system are equally distant. Both problems have been addressed previously by Bozer and Rim [22], and Kıran *et al.* [16] respectively.

Both heuristics and exact methods are proposed for the solution of the UCLP. The works by Kıran and Karabatı [23], Kouvelis and Kim [1], Tansel and Bilen [21], Cheng and Gen [24] and Lee *et al.* [25] are only a few of them. In this work we consider UCLs with single LUL area operational policy. We also assume that the part flow is conserved at every workstation when necessary. We first show new features of the existing formulations. Then we discuss and propose new construction and improvement heuristics for determining optimal layouts. New heuristics are efficient and accurate

according to our experimental results. The rest of this chapter is organized as follows. In the next section we discuss the formulations of the UCLP as a Quadratic Assignment Problem (QAP) and BUCLP as a binary integer linear programming (BILP) problem. We also explain a relaxation of the EUCLP by Bozer and Rim [22] and suggest a slightly more efficient equivalent modification for it. Lower bounds obtained by solving the modified Bozer - Rim relaxation are used in assessing the quality of the heuristics. The third and fourth sections are entirely devoted to heuristics: We summarize some of the existing heuristics, which are used in benchmarking, and propose faster and more accurate new ones. Section five includes computational results and conclusions on the BUCLP.

2.2. Unidirectional Cyclic Layout Problem

Let $N = \{1, \ldots, n\}$ be the set of workstations to be located at the candidate locations connected by a unidirectional circular material handling system where one of the workstations represents the LUL area, and F be the $n \times n$ part flow matrix whose $(i,j)^{th}$ entry $f_{ij} \geq 0$ denotes the average number of jobs to be moved from workstation i to workstation j over a given length of time. Clearly $f_{ii} = 0$. A discussion of how matrix F is determined from process plans can be found in Tansel and Bilen [21]. Unidirectional cyclic material handling systems together with the n candidate locations specified around it can be modelled by a circuit with n vertices. Notice that n is the number of candidate locations (also workstations) including the LUL area. The vertices are numbered 1 through n in increasing order in clockwise direction, which is assumed without loss of generality as the direction of material flow. The system is illustrated in Figure 2.4.

The weight w_t is the length of arc $(t, t+1)$ and determines the distance between locations t and $t+1$. Notice that location $n+1$ denotes location 1 because of circularity. As a consequence, d_{ij}, the transportation distance from location i to location j, becomes

$$d_{ij} = \begin{cases} \sum_{t=i}^{j-1} w_t & \text{if } i < j \\ \lambda - \sum_{t=i}^{j-1} w_t & \text{if } i > j \end{cases} . \tag{2.1}$$

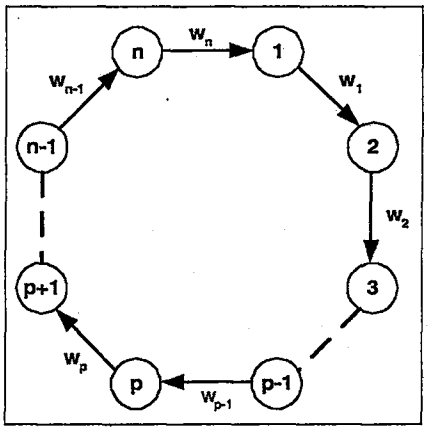

Figure 2.4. A unidirectional cyclic material handling system

Here the constant λ is the length of the circuit. Namely, $\sum_{t=1}^{n} w_t = \lambda$ and distance d_{ij} satisfies the following metric properties:

1. $d_{ij} = 0$ $\quad i, j = 1, \ldots, n; i = j$
2. $d_{ik} + d_{kj} \geq d_{ij}$ $\quad i, j, k = 1, \ldots n; i \neq j \neq k$
3. $d_{ij} + d_{ji} = \lambda$ $\quad i, j = 1, \ldots, n; i \neq j$
4. $d_{ij} \neq d_{ji}$ unless $d_{ij} = \lambda/2$ $\quad i, j = 1, \ldots, n; i \neq j$

A layout of n workstations, which has already been defined as the assignment of workstations $i = 1, \ldots, n$ to candidate locations $\{1, \ldots, n\}$ with one workstation per location, can be denoted as a permutation vector $\pi = (\pi_1, \ldots, \pi_n)$ where π_i is the number of workstation assigned to location i. Then the question is to determine a layout which minimizes certain appropriate cost function of material transport. Two types of objective function have been used in the literature:

1. Minimization of the total part transport distance per unit time;
2. Minimization of the total number of parts that pass through the LUL area per unit time.

As shown by Bozer and Rim [22] and Kıran *et al.* [16] under the first objective the UCLP becomes the QAP after letting Π denote the set of all possible layouts, namely

all possible permutations of indices $\{1,\ldots,n\}$

$$\min_{\pi\in\Pi} c_1(\pi) = \min_{\pi\in\Pi} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} f_{\pi_i\pi_j} d_{ij}. \tag{2.2}$$

To formulate the UCLP with the second objective function Afentakis [19], and Kouvelis and Kim [1] have defined the indicator function

$$I(i,j) = \begin{cases} 1 & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}, \tag{2.3}$$

which is used to count the number of parts passing through the LUL area. If the location i is greater than location j, the parts going from workstation π_i, located at location i, to workstation π_j, located at location j, pass through the LUL area; and this results in the following QAP formulation of the UCLP:

$$\min_{\pi\in\Pi} c_2(\pi) = \min_{\pi\in\Pi} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} f_{\pi_i\pi_j} I(i,j). \tag{2.4}$$

In fact,

$$c_1(\pi) = \lambda c_2(\pi) \tag{2.5}$$

both formulations become equivalent for *UCLs with single LUL area* and *balanced part flow*. This result has been shown by Kouvelis and Kim [1], and Tansel and Bilen [26]. Then, any one of the workstations can be chosen as the LUL area. As a remark, although it is not particularly stated in any of these works, balanced part flow assumption is necessary in the derivation of relation (2.5). In order to see this necessity let us consider a UCL of three workstations with unbalanced workstation-to-workstation

part flow matrix

$$F_1 = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 0 & 2 \\ 2 & 4 & 0 \end{pmatrix},$$

and assume that circuit length $\lambda = 1$ and location 1 is the LUL area. For the layout $\pi = (1, 3, 2)$, namely for workstation 1 is located at location 1, workstation 2 is located at location 3, and workstation 3 is located at location 2,

$$\begin{aligned} c_1(\pi) &= 3d_{12} + d_{13} + 2d_{21} + 4d_{23} + d_{31} + 2d_{32} \\ &= 2(d_{12} + d_{21}) + (d_{13} + d_{31}) + 2(d_{23} + d_{32}) + 2d_{23} + d_{12} \\ &= 5 + d_{12} + 2d_{23} > 5 = \lambda c_2(\pi). \end{aligned}$$

However, for

$$F_2 = \begin{pmatrix} 0 & 2 & 3 \\ 1 & 0 & 6 \\ 4 & 5 & 0 \end{pmatrix},$$

which is balanced, we obtain

$$c_1(\pi) = 11 = \lambda c_2(\pi).$$

Relation (2.5) between $c_1(\pi)$ and $c_2(\pi)$ is important because $c_2(\pi)$ has the following property which makes new heuristics computationally efficient for the BUCLP. Consider the initial layout

$$\overline{\pi} = (\pi_1, ..., \pi_{i-1}, \pi_i, \pi_{i+1}, ..., \pi_{j-1}, \pi_j, \pi_{j+1}, ..., \pi_n), \tag{2.6}$$

and *move forward* workstation π_i located at location i to location j (with $i < j$). This

results in the new layout

$$\overline{\pi}^f = (\pi_1, ..., \pi_{i-1}, \pi_{i+1}, ..., \pi_{j-1}, \pi_j, \pi_i, \pi_{j+1}, ..., \pi_n), \tag{2.7}$$

where workstation π_i is now at location j. For the same initial layout (2.6), when workstation π_j located at location j is *moved backward* to location i (with $i < j$) the new layout

$$\overline{\pi}^b = (\pi_1, ..., \pi_{i-1}, \pi_j, \pi_i, \pi_{i+1}, ..., \pi_{j-1}, \pi_{j+1}, ..., \pi_n), \tag{2.8}$$

is obtained. Workstation π_j is now at location i. Then, the changes in the cost function $c_2(\pi)$ due to the forward move of workstation π_i to location j and the backward move of workstation π_j to location i are respectively

$$\Delta^f_{c_2} = c_2(\overline{\pi}) - c_2(\overline{\pi}^f) = \sum_{k=i+1}^{j} (f_{\pi_k\pi_i} - f_{\pi_i\pi_k}), \tag{2.9}$$

and

$$\Delta^b_{c_2} = c_2(\overline{\pi}) - c_2(\overline{\pi}^b) = \sum_{k=i}^{j-1} (f_{\pi_j\pi_k} - f_{\pi_k\pi_j}). \tag{2.10}$$

This is a direct consequence of the definition of the cost function $c_2(\pi)$. We can

determine the costs of layouts $\overline{\pi}$ and $\overline{\pi}^f$ as,

$$\begin{aligned} c_2(\overline{\pi}) = & \sum_{k=j+1}^{n}\sum_{l=1}^{i-1} f_{\pi_k\pi_l} + \sum_{k=j+1}^{n} f_{\pi_k\pi_i} + \sum_{k=j+1}^{n}\sum_{l=i+1}^{j-1} f_{\pi_k\pi_l} + \sum_{k=j+1}^{n} f_{\pi_k\pi_j} \\ & + \sum_{l=1}^{i-1} f_{\pi_j\pi_l} + f_{j,i} + \sum_{l=i+1}^{j-1} f_{\pi_j\pi_l} \\ & + \sum_{k=i+1}^{j-1}\sum_{l=1}^{i-1} f_{\pi_k\pi_l} + \sum_{k=i+1}^{j-1} f_{\pi_k\pi_i} \\ & + \sum_{l=1}^{i-1} f_{\pi_i\pi_l} \\ & + \sum_{k=2}^{i-1}\sum_{l=1}^{k-1} f_{\pi_k\pi_l} + \sum_{k=i+2}^{j-1}\sum_{l=i+1}^{k-1} f_{\pi_k\pi_l} + \sum_{k=j+2}^{n}\sum_{l=j+1}^{k-1} f_{\pi_k\pi_l} \ , \end{aligned}$$

and

$$\begin{aligned} c_2(\overline{\pi}^f) = & \sum_{k=j+1}^{n}\sum_{l=1}^{i-1} f_{\pi_k\pi_l} + \sum_{k=j+1}^{n}\sum_{l=i+1}^{j-1} f_{\pi_k\pi_l} + \sum_{k=j+1}^{n} f_{\pi_k\pi_j} + \sum_{k=j+1}^{n} f_{\pi_k\pi_i} \\ & + \sum_{l=1}^{i-1} f_{\pi_i\pi_l} + \sum_{l=i+1}^{j-1} f_{\pi_i\pi_l} + f_{\pi_i\pi_j} \\ & + \sum_{l=1}^{i-1} f_{\pi_j\pi_l} + \sum_{l=i+1}^{j-1} f_{\pi_j\pi_l} \\ & + \sum_{k=i+1}^{j-1}\sum_{l=1}^{i-1} f_{\pi_k\pi_l} \\ & + \sum_{k=2}^{i-1}\sum_{l=1}^{k-1} f_{\pi_k\pi_l} + \sum_{k=i+2}^{j-1}\sum_{l=i+1}^{k-1} f_{\pi_k\pi_l} + \sum_{k=j+2}^{n}\sum_{l=j+1}^{k-1} f_{\pi_k\pi_l}. \end{aligned}$$

Then $\Delta_{c_2}^f$ can be obtained by subtracting $c_2(\overline{\pi}^f)$ from $c_2(\overline{\pi})$, resulting in many cancellations and finally in the simple expression (2.9). $\Delta_{c_2}^b$ can be obtained similarly. Notice that (2.9) and (2.10) can be used for the BUCLP with $c_1(\pi)$ or $c_2(\pi)$ since both objectives are equivalent. However, this is not possible for the unbalanced UCLP.

In their early work Bozer and Rim [22], later on Kıran *et al.* [16], have shown that the cost $c_1(\pi)$ of a layout π is independent of where the workstations are located

around the unidirectional cyclic material handling system when the part flow is balanced. In other words the predetermined locations can be shifted without modifying the value of the objective function as long as the layout, namely the sequence of the workstations around the unidirectional cyclic material handling system, remains the same. In short, when the part flow is balanced, even the locations are nonequidistant, the problem becomes equivalent to the determination of a cyclic permutation of the workstations around the unidirectional circular material handling system, which minimizes total material transport. In fact it has been shown that the BUCLP and EUCLP are equivalent (Bozer and Rim [22], Kıran *et al.* [16]).

2.2.1. Afentakis' Formulation

In his seminal paper, Afentakis [19] proposed a BILP formulation for the BUCLP which he called as the Loop Interconnection Problem (LIP). In his formulation, a traffic graph $D = (V, A)$ is defined consisting of vertices V denoting the workstations and directed arcs $(i, j) \in A$ reflecting the part flow from workstation i to workstation j, with the costs f_{ij} associated for each arc $(i, j) \in A$. He assumed that parts enter and exit the system through a LUL area and complete an integer number of tours around the loop before leaving the system, which makes the material flow balanced. LIP is given below:

$$\text{LIP:} \qquad \min \sum_{(i,j)\in A}^{n} f_{ij}(1 - y_{ij}) \tag{2.11}$$

$$\text{s.t. } y_{ij} + y_{ji} = 1 \qquad i, j = 1, \ldots, n; i \neq j \tag{2.12}$$

$$y_{ik} + y_{kj} - 1 \leq y_{ij} \qquad i, j, k = 1, \ldots, n; i \neq j \neq k \tag{2.13}$$

$$y_{ij} = 0, 1 \qquad i, j = 1, \ldots, n; i \neq j \tag{2.14}$$

where

$$y_{ij} = \begin{cases} 1 & \text{if workstation } j \text{ is located after} \\ & \text{workstation } i \text{ in an optimal layout} \\ 0 & \text{otherwise} \end{cases}.$$

As noted by Afentakis [19], the objective function (2.11) minimizes the total number of times the parts cross the LUL area. Constraints (2.12) guarantee either workstation i is located before workstation j, or vice versa. Constraints (2.13) state if workstation i is located before workstation k and workstation k is located before workstation j, then workstation i must be located before workstation j.

2.2.2. Kıran, Ünal and Karabatı's Formulation

Later, Kıran *et al.* [16] have developed an extended formulation for the BUCLP. They define the following auxiliary binary decision variables

$$x_{ij} = \begin{cases} 1 & \text{if workstation } j \text{ immediately follows} \\ & \text{workstation } i \text{ in an optimal layout} \\ 0 & \text{otherwise} \end{cases},$$

and let d_{ij} denote the distance between workstations i and j. The distances can be scaled by dividing the constant circuit length λ. This does not affect the solution of the problem. Thus, it is possible to assume that $\lambda = 1$ and $0 \leq d_{ij} \leq 1$ without loss of generality. Then their BILP formulation for the BUCLP is given as

$$\text{BUCLP:} \quad \min \sum_{\substack{i=1 \\ i \neq j}}^{n} \sum_{j=1}^{n} f_{ij} d_{ij} \tag{2.15}$$

$$\text{s.t.} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \tag{2.16}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{2.17}$$

$$d_{ij} + d_{ji} = 1 \qquad i, j = 1, \ldots, n; i \neq j \tag{2.18}$$

$$d_{kj} \geq d_{ki} + d_{ij} + x_{ij} - 1 \qquad i, j, k = 1, \ldots, n; i \neq j \neq k \tag{2.19}$$

$$x_{ij} = 0, 1 \qquad i, j = 1, \ldots, n; i \neq j \tag{2.20}$$

$$0 \leq d_{ij} \leq 1 \qquad i, j = 1, \ldots, n; i \neq j. \tag{2.21}$$

In this formulation constraints (2.16) and (2.17) are exactly the assignment constraints and ensure that each workstation has exactly one predecessor and one successor. Constraints (2.18) and (2.19) define the properties of the distance matrix. Also (2.19) guarantees that $d_{ij} = d_{ik} + d_{kj}$ if workstation j immediately follows workstation k in the flow direction in an optimal layout as shown by Kıran *et al.* [16]. They have shown that this is a valid BUCLP formulation, and conjectured that it is in fact *ideal* and its linear programming relaxation gives always an integer optimal solution based on their test with 3600 random instances generated for $n = 5, 6, 7$ workstations. However, Tansel and Bilen [27] have demonstrated fractional solutions are possible for larger problems, later on. An interesting consequence of this formulation is the interpretation of the distance variables d_{ij}. The flow is balanced first of all. It is also assumed that parts enter and leave the system by the LUL area. As a result the objective function of the BUCLP becomes equivalent to $c_2(\pi)$ of (2.4), and counts the number of times parts pass through the LUL area, since the length λ of the UCL is 1. Therefore, d_{ij} must behave as the indicator function (2.3) and be equal to 1 if workstation i is after workstation j in a layout. This also explains why these variables have always 0 or 1 values at the optimality.

Any feasible solution of the BUCLP consists of a feasible UCL and distances between workstations. Any UCL is a cyclic permutation of workstations which determine a circular sequence of them. As a consequence we can represent a feasible UCL as a Hamiltonian circuit of the Afentakis' traffic graph. Now we would like to discuss some properties of the BUCLP formulation. From now on we will assume that the traffic graph is a complete digraph. This does cause any loss of generality since f_{ij} can be made arbitrarily large for any non existing arc $(i, j) \in A$. Let us consider arbitrarily three workstations i, j, k and the distance constraints (2.19) involved. These are

$$d_{ik} \geq d_{ij} + d_{jk} + x_{jk} - 1 \tag{2.22}$$

and

$$d_{ki} \geq d_{kj} + d_{ji} + x_{ji} - 1. \tag{2.23}$$

If $x_{jk} = 1$ then $x_{ji} = 0$ because there can be only one exit from every workstation as a consequence of the assignment constraint $\sum_{r=1}^{n} x_{jr} = 1$. Also $d_{ij} = 1 - d_{ji}$, $d_{ki} = 1 - d_{ik}$ from the circularity constraints (2.19). Then for $x_{jk} = 1$ (2.22) and (2.23) become respectively

$$d_{ik} \geq d_{ij} + d_{jk} \tag{2.24}$$

and

$$1 - d_{ik} \geq 1 - d_{jk} + 1 - d_{ij} - 1. \tag{2.25}$$

The last inequality can be reorganized to obtain

$$d_{ik} \leq d_{ij} + d_{jk}$$

which implies that

$$d_{ik} = d_{ij} + d_{jk}.$$

when considered together with (2.24). As a consequence of this discussion the following lemma was originally proven by Kıran *et al.* [16].

Lemma 2.1 [16] For any three workstations i,j and k, if j immediately precedes k on a circuit, then $d_{ik} = d_{ij} + d_{jk}$.

Both, the proofs of the next lemma and the following theorem use Lemma 2.1. Notice that Theorem 2.1 strengthens Theorem 3 given in [16].

Lemma 2.2 Consider any two workstations i_p, i_q on a given circuit $D_C = (V_C, C)$ of the traffic graph $D = (V, A)$ and the directed path $\{(i_p, i_{p+1}), (i_{p+1}, i_{p+2}), \ldots, (i_{q-3}, i_{q-2}), (i_{q-2}, i_{q-1}), (i_{q-1}, i_q)\}$ connecting them on the circuit D_C. Then $d_{i_p i_q} = \sum_{r=p}^{q-1} d_{i_r i_{r+1}}$.

Proof. As a consequence of Lemma 2.1 the equalities

$$\begin{aligned} d_{i_p i_q} &= d_{i_p i_{q-1}} + d_{i_{q-1} i_q} \\ d_{i_p i_{q-1}} &= d_{i_p i_{q-2}} + d_{i_{q-2} i_{q-1}} \\ d_{i_p i_{q-2}} &= d_{i_p i_{q-3}} + d_{i_{q-3} i_{q-2}} \\ &\vdots \\ d_{i_p i_{p+2}} &= d_{i_p i_{p+1}} + d_{i_{p+1} i_{p+2}} \end{aligned}$$

can be written since $x_{i_r i_{r+1}} = 1$ for $r = p, p+1, \ldots, q-1$, which reduces to $d_{i_p i_q} = \sum_{r=p}^{q-1} d_{i_r i_{r+1}}$ because of the cancellations. ■

Theorem 2.1 $\sum_{(i,j)\in C} d_{ij} = 1$ for any circuit $D_C = (V_C, C)$ of the traffic graph $D = (V, A)$.

Proof. The assertion is trivially true for $|C| = 2$ since $d_{ij} + d_{ji} = 1$. Let C be the arc set of any circuit of size m and let $C = \{(i_1, i_2), (i_2, i_3), \ldots, (i_{m-1}, i_m), (i_m, i_{m+1})\}$ with $i_{m+1} = i_1$, be the circuit arcs. For the vertices on the circuit we can write

$$d_{i_t i_{p+1}} = d_{i_t i_p} + d_{i_p i_{p+1}} \quad t = 1, \ldots, m, \; p = 1, \ldots, m; \quad t \neq p, p+1. \tag{2.26}$$

as a consequence of Lemma 2.1. The addition of these inequalities side by side results in

$$\sum_{p=1}^{m} \sum_{\substack{t=1 \\ t\neq p,p+1}}^{m} \left(d_{i_t i_p} + d_{i_p i_{p+1}}\right) = \sum_{p=1}^{m} \sum_{\substack{t=1 \\ t\neq p,p+1}}^{m} d_{i_t i_{p+1}}. \tag{2.27}$$

For any triplet (i_t, i_p, i_{p+1}) of vertices, where $t \neq p$ and $t \neq p+1$, there are two possible cases as it can be seen in Figure 2.5. If $t \neq p+2$, then it is possible to write equalities

$$d_{i_t i_p} + d_{i_p i_{p+1}} = d_{i_t i_{p+1}}$$

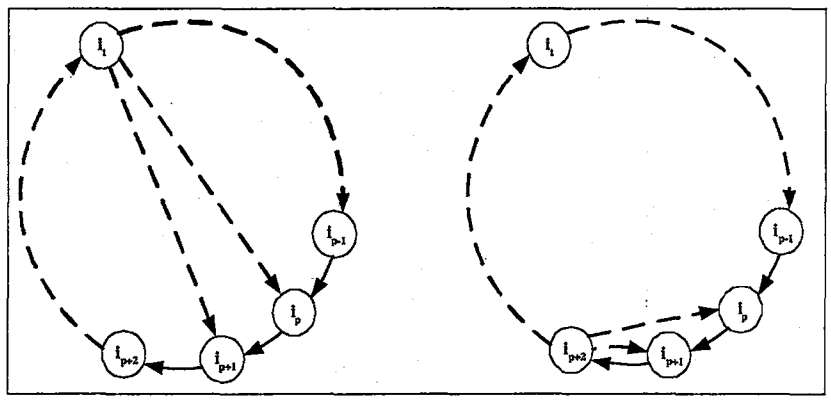

Figure 2.5. Possible cases for triplet (i_t, i_p, i_{p+1})

and

$$d_{i_t i_{p+1}} + d_{i_{p+1} i_{p+2}} = d_{i_t i_{p+2}}$$

respectively for triplets (i_t, i_p, i_{p+1}) and (i_t, i_{p+1}, i_{p+2}). When these two equalities are added side by side to derive (2.27) the term $d_{i_t i_{p+1}}$ is cancelled.

If $t = p + 2$, then it is possible to write

$$d_{i_t i_p} + d_{i_p i_{p+1}} = d_{i_t i_{p+1}}$$

where the term $d_{i_t i_p}$ is cancelled in (2.27) because of the equality

$$d_{i_t i_{p-1}} + d_{i_{p-1} i_p} = d_{i_t i_p}.$$

As a result of both cases (2.27) reduces to

$$(m-2) \sum_{p=1}^{m} d_{i_p i_{p+1}} + \sum_{p=1}^{m} d_{i_p i_{p+1}} = \sum_{p=1}^{m} d_{i_{p+1} i_p} \tag{2.28}$$

whose right hand side can be re-expressed as

$$\sum_{p=1}^{m} \left(1 - d_{i_p i_{p+1}}\right). \tag{2.29}$$

After replacing the right hand side of (2.28) with (2.29) and arranging the terms we obtain

$$m \sum_{p=1}^{m} d_{i_p i_{p+1}} = m$$

which implies that

$$\sum_{p=1}^{m} d_{i_p i_{p+1}} = 1$$

and the proof is complete. ■

There are two direct consequences of Theorem 2.1. The first one, is given as Corollary 2.1, Constraints 2.18 and 2.19 do not allow any subtour of the traffic graph in a feasible solution of BUCLP. Corollary 2.2 points to an interesting feature of the same formulation. Constraints (2.18) and (2.19) eliminate any subtour with one exception: Hamiltonian circuits of the traffic graph do not cause any feasibility and they are not eliminated.

Corollary 2.1 Constraints (2.18) and (2.19) eliminate any subtour.

Proof. Let $D_C = (V_C, C)$ be any non Hamiltonian circuit on the traffic digraph, which is assumed to be the complete digraph with vertex set $V = \{1, \ldots, n\}$. Then for any vertex k which is not on the circuit and for all arc (i, j) on the circuit

$$\sum_{j \in V_C} d_{kj} \geq \sum_{i \in V_C} d_{ki} + \sum_{(i,j) \in C} d_{ij} + \sum_{(i,j) \in C} x_{ij} - |C| \tag{2.30}$$

can be written. Since $\sum_{j \in V_C} d_{kj} = \sum_{i \in V_C} d_{ki}$ (2.30) becomes

$$\sum_{(i,j) \in C} x_{ij} \leq |C| - \sum_{(i,j) \in C} d_{ij}$$

which completes the proof since $\sum_{(i,j) \in C} d_{ij} = 1$ as a consequence of Theorem 2.1. ■

Corollary 2.2 Constraints (2.18) and (2.19) are not violated for any Hamiltonian circuit of the traffic graph.

Proof. Let C be the arc set of size $m \geq 3$ of any circuit $D_C = (V_C, C)$, and let $C = \{(i_1, i_2), (i_2, i_3), \ldots, (i_{m-1}, i_m), (i_m, i_{m+1})\}$ with $i_{m+1} = i_1$, be the circuit arcs. Consider vertices i_1, i_k and i_l, and the inequalities

$$\begin{aligned}
d_{i_k i_2} &\geq d_{i_k i_1} + d_{i_1 i_2} + x_{i_1 i_2} - 1 \\
d_{i_1 i_3} &\geq d_{i_1 i_2} + d_{i_2 i_3} + x_{i_2 i_3} - 1 \\
d_{i_1 i_4} &\geq d_{i_1 i_3} + d_{i_3 i_4} + x_{i_3 i_4} - 1 \\
d_{i_1 i_5} &\geq d_{i_1 i_4} + d_{i_4 i_5} + x_{i_4 i_5} - 1 \\
&\vdots \\
d_{i_1 i_m} &\geq d_{i_1 i_{m-1}} + d_{i_{m-1} i_m} + x_{i_{m-1} i_m} - 1 \\
d_{i_l i_1} &\geq d_{i_l i_m} + d_{i_m i_1} + x_{i_m i_1} - 1
\end{aligned}$$

with $k \neq 1, 2, l$ and $l \neq 1, m, k$. Same argument can be repeated for any other three vertices with the same properties. When these inequalities are summed up side by side

$$(d_{i_k i_2} + d_{i_1 i_m} + d_{i_l i_1}) \geq (d_{i_k i_1} + d_{i_1 i_2} + d_{i_l i_m}) + \sum_{(i,j) \in C} d_{ij} + \sum_{(i,j) \in C} x_{ij} - |C|$$

is obtained because of cancellations. In addition, the replacements $d_{i_k i_2} = 1 - d_{i_2 i_k}$, $d_{i_1 i_m} = 1 - d_{i_m i_1}$ and $d_{i_l i_1} = 1 - d_{i_1 i_l}$, and appropriate manipulations give

$$3 \geq (d_{i_1 i_2} + d_{i_2 i_k} + d_{i_k i_1}) + (d_{i_1 i_l} + d_{i_l i_m} + d_{i_m i_1}) + \sum_{(i,j) \in C} d_{ij} + \sum_{(i,j) \in C} x_{ij} - |C| \,.$$

Then, $d_{i_1 i_2} + d_{i_2 i_k} + d_{i_k i_1} = \sum_{(i,j) \in C} d_{ij}$ and $d_{i_1 i_l} + d_{i_l i_m} + d_{i_m i_1} = \sum_{(i,j) \in C} d_{ij}$ follow from Lemma 2.2. Besides $\sum_{(i,j) \in C} d_{ij} = 1$ by Theorem 2.1. Hence, the last inequality reduces to

$$3 \geq 1 + 1 + 1 + m - m,$$

which does not cause any violation of the subtour elimination constraints.

For the case $k = l \neq 1, 2, m$ same arguments result in

$$2 \geq (d_{i_1 i_2} + d_{i_2 i_k} + d_{i_k i_m} + d_{i_m i_1}) + \sum_{(i,j) \in C} d_{ij} + \sum_{(i,j) \in C} x_{ij} - |C| .$$

This clearly reduces to

$$2 \geq 1 + 1 + m - m,$$

since $d_{i_1 i_2} + d_{i_2 i_k} + d_{i_k i_m} + d_{i_m i_1} = 1$ as a consequence of Lemma 2.2 and Theorem 2.1. Therefore, we can consider a Hamiltonian circuit instead of $D_C = (V_C, C)$ in particular, which completes the proof. ■

Theorem 2.1 and its two corollaries will be used in Chapter 5 in the context of the new ATSP formulations to show their validity and strength.

The previously shown two lemmas and theorem are the basic properties of the BUCLP formulation. We will not only refer them in the following discussion but also in Section 5.1.1 in the context of the new ATSP formulations.

Now we would like to present a new property of the BUCLP formulation (2.15)–(2.21). Consider any feasible solution (x, d). Then x must be the characteristic vector of an Hamiltonian circuit of the traffic graph as a consequence of Corollary 2.1 and Corollary 2.2. Notice that the vertex x represent a feasible unidirectional cyclic layout (UCL) in the mean time. Moreover, the vector d satisfies (2,18) and $d_{ik} = d_{ij} + d_{jk}$ for any three workstations $i \neq j \neq$ as a consequence of Lemma 2.2. Since $d_{ik} = 1 - d_{ki}$, $d_{ij} + d_{jk} + d_{ki} = 1$ for $i < j < k$ in particular. In other words, auxiliary variable vector

d of any feasible BUCLP solution satisfies constraints

$$d_{ij} + d_{ji} = 1 \qquad i < j \tag{2.31}$$

$$d_{ij} + d_{jk} + d_{ki} = 1 \qquad i < j < k \tag{2.32}$$

The converse is also true. Consider the vector (x, d) where x is the characteristic vector of any Hamiltonian circuit of the traffic graph and assume d satisfies equalities (2.31) and (2.32). It trivially satisfies equalities (2.18) because of the symmetry of the variables d_{ij} and d_{ji}. For any two workstations j and k where k is the immediate successor of j on a feasible layout, $x_{jk} = 1$. Then for any $i \neq j, k$ constraints (2.19) becomes

$$d_{ij} + d_{jk} + d_{ki} \leq 1 \qquad i = 1, \ldots, n; \;\; i \neq j$$

which is clearly satisfied by any d satisfying (2.31). In case k is not the immediate successor of j, $x_{jk} = 0$ and (2.19) becomes

$$d_{ij} + d_{jk} + d_{ki} \leq 2 \qquad i = 1, \ldots, n; \;\; i \neq j$$

which is again satisfied by any d satisfying (2.31). Therefore we have shown the following lemma.

Lemma 2.3 For any given UCL the BUCLP formulation (2.15)–(2.21) reduces to the following LP in d_{ij}:

$$\text{P:} \qquad \min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ij} d_{ij}$$

$$\text{s.t. } (2.31), (2.32), d_{ij} \geq 0 \qquad i < j$$

Now we can write the dual problem of P as

$$\text{D:} \quad \max \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} v_{ijk} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} u_{ij} \tag{2.33}$$

$$\text{s.t. } u_{ij} + \sum_{\substack{k=3 \\ j<k}}^{n} v_{ijk} + \sum_{\substack{k=1 \\ k<i}}^{n-2} v_{kij} \leq f_{ij} \quad i,j = 1,\ldots,n;\ i<j \tag{2.34}$$

$$u_{ji} + \sum_{\substack{k=2 \\ j<k<i}}^{n-1} v_{jki} \leq f_{ij} \quad i,j = 1,\ldots,n;\ j<i \tag{2.35}$$

$$u_{ij},\ v_{ijk} \text{ unrestricted} \quad i,j,k = 1,\ldots,n;\ i<j<k \tag{2.36}$$

Here u_{ij} and v_{ijk} are the dual variables associated respectively with constraints (2.31) and (2.32). Consider the primal – dual solution pair

$$d_{ij} = \begin{cases} 1 & i,j = 1,\ldots,n; j<i \\ 0 & \text{otherwise} \end{cases} \tag{2.37}$$

$$v_{ijk} = \begin{cases} f_{jk} - f_{kj} & i = 1,\ 2 \leq j < k \\ 0 & \text{otherwise} \end{cases} \tag{2.38}$$

$$u_{ij} = \begin{cases} f_{j1} + \sum_{\substack{k=2 \\ k<j}}^{n} (f_{jk} - f_{kj}) & i = 1,\ 2 \leq j \\ f_{ji} & \text{otherwise} \end{cases}$$

They are respectively primal and dual feasible. Notice that

$$f_{1j} + \sum_{\substack{i=2 \\ i>j}} (f_{ij} - f_{ji}) = f_{j1} + \sum_{\substack{i=2 \\ i<j}} (f_{ji} - f_{ij})$$

follows for all $j > 1$, because the part flow is balanced and $\sum_{i=1, i\neq j} f_{ji} = \sum_{i=1, i\neq j} f_{ij}$ for $j = 1,\ldots,n$. If we let z_P and z_D denote primal and dual objective values, then for

(2.37)

$$z_P = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij} d_{ij}$$
$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ji}$$

and for (2.38)

$$z_D = \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} v_{ijk} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} u_{ij}$$
$$= \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} v_{1ij} + \sum_{j=2}^{n} u_{1j} + \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} u_{ij}$$
$$= \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} (f_{ij} - f_{ji}) + \sum_{j=2}^{n} (f_{j1} + \sum_{\substack{i=2 \\ i<j}}^{n} (f_{ji} - f_{ij})) + \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} f_{ji}$$
$$= \sum_{j=2}^{n} f_{j1} + \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} f_{ji}$$
$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ji}.$$

As a result of this discussion we have proven that (2.37) and (2.38) are primal and dual feasible solutions with equal objective function values. We state this result with the following theorem.

Theorem 2.2 (2.37) and (2.38) are primal and dual optimal solutions of P.

In short, for any given feasible UCL we can determine d_{ij} values and evaluate the part flow cost very efficiently, which makes the tailoring of any ATSP heuristic for the BUCLP straightforward.

2.2.3. Bozer and Rim's Formulation

Bozer and Rim [22] have proposed the following linear programming (LP) formulation for the EUCLP and claimed that it always gives integer solutions and thus it is an ideal EUCLP formulation.

$$\text{EUCLP}_{LP}: \quad \min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} f_{ij} d_{ij} \tag{2.39}$$

$$\text{s.t.} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} = \frac{n(n-1)}{2} \qquad i = 1, \ldots, n \tag{2.40}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} d_{ij} = \frac{n(n-1)}{2} \qquad j = 1, \ldots, n \tag{2.41}$$

$$d_{ij} + d_{ji} = n \qquad i, j = 1, \ldots, n; i \neq j \tag{2.42}$$

$$d_{ij} + d_{jk} \leq d_{ik} + n \qquad i, j, k = 1, \ldots, n; i \neq j \neq k \tag{2.43}$$

$$d_{ij} + d_{jk} \geq d_{ik} \qquad i, j, k = 1, \ldots, n; i \neq j \neq k \tag{2.44}$$

$$d_{ij} \geq 0 \qquad i, j = 1, \ldots, n; i \neq j \tag{2.45}$$

They assume that the length of the cyclic material handling system, namely λ, is equal to n, the number of workstations. Therefore, the distance between two consecutive locations is 1 and the variable d_{ij} eventually counts the number of workstations between workstation i and workstation j, including workstation j but not workstation i. Constraints (2.40)–(2.41) define the properties of the distances. First of all, regardless of the workstation sequences, the distances from workstation i to all other workstations add up to the same constant value:

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} d_{ij} = 1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}. \tag{2.46}$$

Constraints (2.42) are the circularity constraints and follow from the fact that the layout is cyclic. Constraints (2.43) and (2.44) are related to the precedence relationship of any three workstations around the unidirectional material handling system: Starting

at workstation i, either workstation j precedes workstation k, or workstation k precedes workstation j. Namely,

$$d_{ij} + d_{jk} = \begin{cases} d_{ik} & \text{if workstation } j \text{ precedes } k \\ d_{ik} + n & \text{if workstation } k \text{ precedes } j \end{cases} \tag{2.47}$$

However, if (2.47) is employed in the formulation as a constraint set, the formulation would have to include binary variables. As a result Bozer and Rim [22] propose to use a relaxed version of (2.47), namely constraints (2.43) and (2.44) instead. Although they give a number of intermediate results to show that the relaxation EUCLP$_{LP}$ has integer optimum solutions and solves the EUCLP to optimality, the proof has not been validated so far. According to Tansel and Bilen [27], their claim seems to be hardly true since both the BUCLP and EUCLP are equivalent and there are instances for which the LP relaxation of the BUCLP has optimal fractional solutions.

Bozer and Rim's relaxation EUCLP$_{LP}$, can be simplified. First, n in the right hand side of the inequality (2.43) can be replaced with $d_{ij} + d_{ji}$, which gives

$$d_{ij} + d_{jk} \leq d_{ik} + d_{ij} + d_{ji}, \quad i, j, k = 1, \ldots, n; i \neq j \neq k. \tag{2.48}$$

Then, d_{ij} can be cancelled from both sides to obtain

$$d_{jk} \leq d_{ji} + d_{ik}, \qquad i, j, k = 1, \ldots, n; i \neq j \neq k, \tag{2.49}$$

which are in fact the triangle inequalities (2.44). Therefore, we can agree that constraints (2.44) are redundant since they are implied by (2.42) and (2.43). On the other hand, we can rewrite constraints (2.40) and (2.41) as

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} = \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ji} = \frac{n(n-1)}{2} \quad i = 1, \ldots, n. \tag{2.50}$$

When we add equalities (2.42) for $j = 1, \ldots, n$, $j \neq i$ we end up with

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} (d_{ij} + d_{ji}) = n(n-1) \quad i = 1, \ldots, n. \tag{2.51}$$

Together with (2.50), (2.51) implies that it suffices to use

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} = \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ji} \qquad i = 1, \ldots, n \tag{2.52}$$

instead of constraints (2.40) and (2.41). In other words, this new relaxation, which is referred as EUCLP'$_{LP}$ in the sequel, consists of the minimization of the objective function (2.39) subject to the constraint sets (2.42), (2.43), (2.45) and (2.52). As a consequence of this discussion the following proposition follows.

Proposition 2.1 EUCLP$_{LP}$ is equivalent to EUCLP'$_{LP}$

Note that EUCLP$_{LP}$ has n^3+n^2+n constraints instead of $2n^3+n^2+2n$. Therefore, EUCLP'$_{LP}$ is used to calculate lower bounds on the optimal objective value of the BUCLP and also UCLP, which provide benchmarks in assessing the quality of the solutions new heuristics produce.

2.3. Construction and Improvement Heuristics

We briefly summarize five existing heuristic procedures used to construct and improve UCLs in the following. Some of them are essentially proposed for solving the UCLP, namely for the unbalanced and unequally spaced unidirectional cyclic layout problem. They can be used for the solution of its balanced version, namely the BUCLP, in particular. We implement and use them as benchmarks in assessing the performance of our new heuristics.

2.3.1. Kouvelis and Kim's Heuristic

Kouvelis and Kim [1] have proposed three construction heuristics: KK1, KK2, and KK3 for the UCLP. They are based on dominance rules, which suggest locating a workstation at the first available location (i.e. the beginning of a UCL) if it has only outflow, and to the last available location (i.e. the end of a UCL) if it has only inflow. We consider only KK3 in assessing the performance of our new heuristics since it has the highest accuracy comparing KK1's and KK2's. KK3 starts with the part flow matrix F and determines workstation pair (i^*, j^*) which gives the largest

$$RC_{ij} = (R_i - C_i) - (R_j - C_j) + f_{ji} - f_{ij} \tag{2.53}$$

value. Here, R_i and C_i are respectively obtained by summing up the entries of the i^{th} row and i^{th} column of F. They denote total outflow and total inflow for workstation i. As for f_{ij} and f_{ji}, they are the number of parts processed in workstation i per unit time that must be routed to workstation j, and vice versa. These two workstations, namely i^* and j^*, are respectively assigned to the first and the last available locations. Then rows i^* and j^*, and columns $i^*$and j^* are deleted from F and a new (i^*, j^*) is determined by using (2.53) on the new part flow matrix F. These steps are repeated until the part flow matrix consists of a single element. KK3 yields a layout where workstations with higher outflow rate are located towards the beginning and workstations with higher inflow rate towards the end of a layout.

Notice that formula (2.53) becomes

$$RC_{ij} = f_{ji} - f_{ij} \tag{2.54}$$

when the problem is balanced since $R_i = C_i$ for all $i = 1, \ldots, n$, and KK3 uses only flow values between workstation pairs.

2.3.2. Move Heuristic

Tansel and Bilen [21] consider the BUCLP in particular, and propose heuristics based on the improvements that moving workstations from one location to another one can do in the objective function. Given the current assignment of n workstations to n available locations, a layout π, move heuristic (MOVE) computes the change in the objective value that results because of moving any workstation π_i located at location i to any of the locations $j \neq i$. This is done for every workstation and the improvements are recorded. Then the move resulting in the largest improvement is realized. The procedure is repeated until no improvement is possible. Moving workstation π_i from location i to location j consists of a sequence of location changes. First workstation π_i is moved to location j. Then, if $j < i$ workstations $\pi_j, \pi_{j+1}, \ldots, \pi_{i-1}$ at locations $j, j+1, \ldots, i-1$, are shifted forward to locations $j+1, j+2, \ldots, i$. If $i < j$, backward shift occurs for workstations $\pi_{i+1}, \pi_{i+2}, \ldots, \pi_j$, from locations $i+1, i+2, \ldots, j$ to locations $i, i+1, \ldots, j-1$.

2.3.3. Pairwise Interchange and 3-Way Interchange Heuristics

One of the oldest improvement heuristics is based on the idea of improving a given layout by means of pairwise interchanges (*swaps*). Famous CRAFT algorithm, which is used to obtain a good feasible solution for the QAP, is a pairwise interchange heuristic (Francis, McGinnis and White, [28]). Since the UCLP is a QAP then it seems natural to exercise pairwise interchange heuristic (SWAP) on it. In short, given an initial feasible assignment, all possible pairwise interchanges are considered and the best one, namely the one with the best improvement in the objective function, is performed. This is repeated until no more improvement is possible. Note that every pairwise interchange can be achieved by two individual move operations and therefore, MOVE can be seen as a special form of SWAP as an heuristic. If we let π be the initial layout we obtain the new layout

$$\overline{\pi} = (\pi_1, ..., \pi_{i-1}, \pi_j, \pi_{i+1}, ..., \pi_{j-1}, \pi_i, \pi_{j+1}, ..., \pi_n) \tag{2.55}$$

when we interchange workstations π_i and π_j located at locations i and j which results in the changes $c_1(\pi)-c_1(\overline{\pi})$ or $c_2(\pi)-c_2(\overline{\pi})$ depending on the type of the objective function used. It is possible to generalize SWAP to k-way interchange heuristic, as discussed by Tansel and Bilen [21]. In this work we only consider 3-way interchange heuristic (3WI): Three workstations exchange their locations in order to yield a better layout. At every step, the improvements of all possible 3-way interchanges are computed and the one with the best improvement is performed.

It may seem possible to obtain even further improvements if MOVE is followed by pairwise interchanges at the first look. However, this is not so since a pairwise interchange is equivalent to two individual move operations as explained above.

2.3.4. Lee, Huang and Chiang's Heuristic

Lee, Huang and Chiang heuristic (LHC) is based on the new conditions for optimal UCLs [25]. They show that, any workstation i with $f_{ij} = f_{ji}$, $j = 1, ..., n$, $j \neq i$, can be arbitrarily located in any position at the optimal solution, and for any pair of adjacent workstations i and j, if $f_{ij} < f_{ji}$ than workstation i must be located before workstation j in an optimal layout. Moreover, they also prove that for any workstation i, if $R_i > C_i$ ($R_i < C_i$) where R_i is the i^{th} row sum and C_i is the i^{th} column sum of the part flow matrix, then workstation i cannot be assigned to the last (first) location immediately before (after) the LUL area. They have devised a branch and bound algorithm using the depth first search strategy and these rules.

In their heuristic, as initialization, workstations i with symmetric part flows (i.e. $f_{ij} = f_{ji}$, $j = 1, ..., n$, $j \neq i$) are detected first. These workstations can arbitrarily be located later. Second, the workstations with $C_i = 0$ ($R_i = 0$) are assigned to available locations immediately after (before) the LUL area. Then, workstation i with the largest $|R_i - C_i|$ value is chosen. If this value is positive (nonpositive) the workstation is located at the first available location after (before) the LUL area. These are the layout construction steps of the LHC.

Finally, an improvement procedure is applied. Workstations with the largest positive difference $R_i - C_i$ $(C_i - R_i)$ are forced to interchange their locations with the workstations located closer to the LUL area against the part flow direction (in the part flow direction) of the material handling system. These interchanges are repeated as long as the objective function keeps decreasing. As the last step, SWAP is applied only for workstations that are not considered in the previous interchanges.

2.4. New Heuristics

As we have already mentioned, the UCLP is a special form of the QAP. Yet, another special form of the QAP is the ATSP. In short, the UCLP and the ATSP are relatives. Both problems deal with the determination of an optimal cyclic permutation; an optimal assignment of workstations to candidate locations on a circle (an optimal UCL) in the case of the UCLP, and an optimal assignment of cities to visit orders (an optimal tour), which can be seen as locations around a circle without loss of generality. In fact, the ATSP and the balanced problem BUCLP are very close. The relation between x_{ij} and d_{ij} variables we have shown in Section 2.2 is another characteristic of the BUCLP that supports this claim. Besides, as it was shown in Chapter 4, it is possible to obtain a valid extended ATSP formulation with $O(n^3)$ subtour elimination constraints by only changing the objective function (2.15) of the formulation (2.15)–(2.21).

Another combinatorial optimization problem related to the UCLP is the well-known LOP (Reinelt, [29]). There is a detailed discussion on the equivalence of both problems in the recent work by Potts and Whitehead [30]. LOP consists of finding a permutation of the columns and rows in order to maximize the sum of the weights in the upper triangle for a given matrix of weights. In economics the LOP has been known as the triangulation problem of the input-output matrices. The economy of a region consists of n sectors and an $n \times n$ input-output matrix is constructed with each $(i,j)^{th}$ entry denoting material flow from sector i to sector j in a given year. Then the triangulation problem becomes permuting the rows and columns of the input-output matrix in order to maximize the sum of the entries above the main diagonal. An

optimal solution clearly gives an ordering, namely a permutation, of the sectors. Since the solution of the UCLP is related to the ordering of the columns and rows of the part flow matrix, approaches to solve the LOP can be adopted for the UCLP. At sum the new heuristics proposed in this section exploit the relations between the UCLP, ATSP and LOP.

2.4.1. Exchange and Relocation Heuristics

The basic exchange heuristic (EX) is originally suggested for the ATSP by Lin [31], which is the well-known *k-opt*. We have used this idea for our problem, namely the UCLP, by exploiting the possibility to represent a unidirectional cyclic layout of workstations as a Hamiltonian circuit. In this heuristic, the location of four workstations is replaced with the ones of four others addressing a decrease in the value of the objective function. For instance, consider an initial layout

$$(p, ..., i, j, ..., k, l, ..., m, n, ..., o),$$

where the locations of workstations i and j, k and l, m and n and, o and p are adjacent. One possible exchange of adjacency is illustrated in Figure 2.6; it results in the new layout

$$(p, ..., i, n, ..., o, l, ..., m, j, ..., k).$$

Notice the differences in the adjacent workstations. As for example workstation i is not adjacent with workstation j anymore; it is adjacent with workstation n. This is one of the $2 \times 4! = 48$ possible exchanges. At each step the best of 48 exchanges is performed and the algorithm is run until no further improvement in the objective function is possible.

In the relocation heuristic (RE), which is originally proposed for the ATSP by Or [32] and known as Or-opt, the interest is in a partial layout consisting of four consecutive workstations. Relocations are restricted only to moving the partial layout

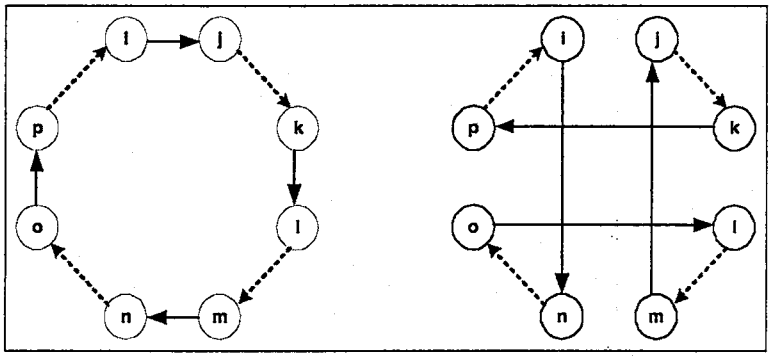

Figure 2.6. An exchange of workstations i, j, k, l, m, n, o, p

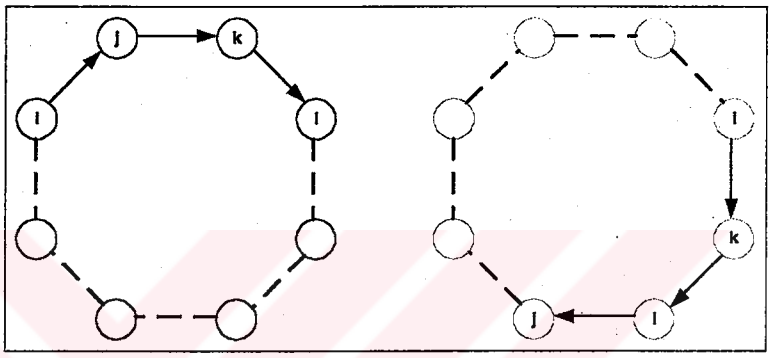

Figure 2.7. A relocation of (i, j, k, l) workstation sequence

consisting of a permutation of four consecutive workstations to $n-4$ possible locations. Out of $4! \times (n-4) = 24 \times (n-4)$ possible layouts the one with the best improvement is selected as the new layout. Two strategies can be used during the relocation of the workstations: Either the first relocation of the four-workstation partial layout which improves the objective function, or the relocation with the best improvement in the objective function is performed. In our implementation, we adopted the best improvement strategy. In Figure 2.7, we illustrate the relocation of a partial layout of four workstations i, j, k, l in a layout of eight workstations. Notice that in the new layout they are located according to the permutation, i, k, l, j. It is possible to see that MOVE is a special form of RE.

2.4.2. Outflow-Inflow Ratio Heuristic

In his early work on the LOP, Becker [33] proposed a construction heuristic based on the ratios of row and column sums of the cost matrix. Each ratio i measures the

attractiveness of sector i by dividing the sum of the entries in the i^{th} row to the sum of the entries in the i^{th} column of the input-output matrix. As we have already mentioned the solution of the UCLP is related to the ordering of the columns and rows of the part flow matrix. Hence, Becker's idea can be adopted to locate the workstations. For this purpose, first the ratios

$$r_i = \frac{R_i}{C_i} \tag{2.56}$$

are calculated for workstations $i = 1, \ldots, n$. Here, R_i and C_i are defined as in KK3 and hence r_i is the outflow-inflow (OI) ratio for workstation i. The workstation with the largest ratio is assigned to the closest available location to the LUL area in the part flow direction. Then, the corresponding column and row are deleted from the part flow matrix and the new ratios for the remaining workstations represented with the remaining part flow matrix are calculated and the workstation with the largest ratio is located right after the previous one. The procedure continues until every workstation is located, namely a layout is determined.

Notice that it is not meaningful to use OI when the problem is balanced since $R_i = C_i$ and thus $r_i = 1$ for all $i = 1, \ldots, n$. Therefore no values will be reported in Table 2.2 and Table 2.4 in Section 2.5.2.

2.4.3. Sort and Reverse Heuristic

As another consequence of the relation between the UCLP and the LOP, we have adopted the efficient sort and reverse heuristic (SR) by Chanas and Kobylanski [34] originally devised for the LOP. In the SR, *sort* and *reverse* operations are successively applied until no further improvement in the objective function value is possible.

A *sort* operation improves a given initial layout. In fact, workstations are assigned to locations from scratch. Consider the layout $\pi = \{\pi_1, \pi_2, \pi_3, ..., \pi_n\}$ of workstations. First, π_1 and π_2, namely workstations at the first and the second locations are considered. The best combination of these two workstations, which has the smallest objective

function value is chosen. That is to say, we initially consider either the partial layout $\{\pi_1, \pi_2\}$ or the partial layout $\{\pi_2, \pi_1\}$. For instance suppose that the partial layout $\{\pi_2, \pi_1\}$ is better and chosen. Next, the third workstation in the initial layout, i.e. π_3, is considered. It is assigned in all possible locations of the partial layout consisting of two workstations. In other words, among the partial layouts $\{\pi_3, \pi_2, \pi_1\}$, $\{\pi_2, \pi_3, \pi_1\}$ and $\{\pi_2, \pi_1, \pi_3\}$, the one with the smallest objective function value is chosen as the new partial layout. The process of reassigning the workstations into partial layouts continues until all of the workstations are located and a complete layout is at hand.

A *reverse* operation simply reverses the order of workstations of a given layout. That is to say, considering the initial layout $\pi = \{\pi_1, \pi_2, \pi_3, ..., \pi_n\}$ of workstations, reverse operation yields the following layout: $\{\pi_n, ..., \pi_3, \pi_2, \pi_1\}$. In particular when applied to the layout obtained by a *sort* operation, usually results in a layout with a larger objective function value. However, when sort is implemented on the reversed layout, a new layout which is much better than the one we obtained with the previous sort operation, can be obtained. In other words, a *reverse* operation somewhat represents an uphill move to escape from a local optimum solution obtained by a *sort* operation. Moreover, it can be easily shown that the *sort* operation following a *reverse* operation does not worsen the layout the *sort* operation preceding the *reverse* operation gives, modifying the arguments by Chanas and Kobylanski [34] proposed for LOP.

2.4.4. Move and Reverse Heuristic

According to the computational results, which can be found in detail in the next section, SR is the fastest improvement method comparing to other improvement heuristics; but unfortunately its solution quality is not as good as MOVE's. Recall that a *reverse* operation in SR results in a jump out of the local minimum latest *sort* operation produces. Similarly the combination of a *reverse* operation with a *move* operation can improve on MOVE's performance by causing a jump out of the local minimum obtained by *move* operations. We propose replacing a *sort* in SR with a run of MOVE to obtain *move and reverse* (MR) heuristic. In MR, MOVE and *reverse* alternate until no further decrease in the objective function is possible.

Now we formally discuss the correctness of MR, namely why the use of MR does never worsen a given feasible layout. Let $MOVE(\pi)$ denote one run of MOVE given the layout π of workstations, and $m(\pi_i; j; \pi)$ represent the individual move operation for workstation π_i, initially at location i, to location j in the layout π. Recall that during one run of MOVE on the given layout of π, namely the execution of $MOVE(\pi)$, we first compute the objective function value of the layouts obtained by performing individual move operations $m(\pi_i; j; \pi)$ of all workstations $\pi_1, ..., \pi_n$ from locations $i = 1, ..., n$ to locations $j = 1, ..., n$. Then, $MOVE(\pi)$ updates the layout π by realizing the individual move operation yielding the best improvement in the objective function value. MOVE runs until no further improvement is possible. These steps are now repeated on the last layout.

Notice that, after performing one *reverse* operation for the layout π, we can obtain the same layout in at least $n - 1$ and at most $n \times (n - 1)/2$ individual *move* operations $m(\pi_i; j; \pi)$. Therefore, this shows that we can not worsen layout π after reversing it, namely executing $REVERSE(\pi)$.

For instance, consider the initial layout $\pi = \{\pi_i, \pi_j\}$ consisting of two workstations. The following cases occur while running MR:

Case 1: Suppose $f_{ij} > f_{ji}$. MR performs first the *reverse* operation which yields

$$\{\pi_j, \pi_i\} \leftarrow REVERSE(\{\pi_i, \pi_j\}).$$

Then, all possible layouts obtained by performing individual move operations $m(\pi_i; 1; \pi)$, $m(\pi_i; 2; \pi)$, $m(\pi_j; 1; \pi)$, and $m(\pi_j; 2; \pi)$ are considered. These operations yield two possible layout options $\{\pi_j, \pi_i\}$ and $\{\pi_i, \pi_j\}$. Among them MOVE performs the one with the smallest objective function value, namely $\pi' = \{\pi_j, \pi_i\}$ is the new layout, since

$f_{ij} > f_{ji}$. These operations can be summarized step by step as follows:

$$MOVE(\{\pi_j, \pi_i\}) \leftarrow MOVE(REVERSE(\{\pi_i, \pi_j\})),$$
$$\{\pi_j, \pi_i\} \leftarrow MOVE(\{\pi_j, \pi_i\}),$$
$$\pi' \leftarrow \{\pi_j, \pi_i\},$$

with $c_1(\pi) > c_1(\pi')$, or $c_2(\pi) > c_2(\pi')$ depending on the type of the cost function used.

Case 2: Suppose, $f_{ij} \leq f_{ji}$. A *reverse* operation produces

$$\{\pi_j, \pi_i\} \leftarrow REVERSE(\{\pi_i, \pi_j\})$$

and individual *move* operations work on layouts $\{\pi_i, \pi_j\}$ and $\{\pi_j, \pi_i\}$. The first layout is performed since it has smaller or equal objective value. This is shown as

$$\{\pi_i, \pi_j\} \leftarrow MOVE(\{\pi_j, \pi_i\}).$$

Finally, $\pi'' = \{\pi_i, \pi_j\}$ is the solution of MR for $f_{ij} \leq f_{ji}$. These steps can be summarized as follows

$$MOVE(\{\pi_j, \pi_i\}) \leftarrow MOVE(REVERSE(\{\pi_i, \pi_j\})),$$
$$\{\pi_i, \pi_j\} \leftarrow MOVE(\{\pi_j, \pi_i\}),$$
$$\pi'' \leftarrow \{\pi_i, \pi_j\}.$$

with $c_1(\pi) = c_1(\pi'')$ or $c_2(\pi) = c_2(\pi'')$ depending on the type of the cost function used.

So far, we have obtained $c_1(\pi) > c_1(\pi')$ for case 1 and $c_1(\pi) = c_1(\pi'')$ for case 2. As a result, we deduce that for two workstations, MR can yield layout π' or layout π'', which are at least as good as the input layout, π. Now we generalize our results further by induction on the number of workstations. Consider a layout π of $n-1$ workstations and assume that MR yields a layout which is at least as good as the input layout π,

namely

$$c_1(\pi) \geq c_1(MOVE(\{REVERSE(\{\pi\})\})).$$

Then attach the end of this layout a new workstation to obtain layout $\pi' = \{\pi, \pi_n\}$ of n workstations. It is clear that

$$c_1(\pi') = c_1(\{\pi, \pi_n\}).$$

On the other hand, by definition of the cost function c_1

$$c_1(\{\pi, \pi_n\}) \geq c_1(MOVE(\{REVERSE(\{\pi\}), \pi_n\})),$$

since the same value, which is actually the cost term associated with workstation π_n, is added to both sides of the previous inequality. Moreover, the move of workstation π_n to location n (this is what it is in fact) for the layout $\pi' = \{\pi, \pi_n\}$ does not affect the value of the objective function c_1, and

$$c_1(MOVE(\{REVERSE(\{\pi\}), \pi_n\}) = c_1(m(\pi_n; n; \{MOVE(\{REVERSE(\{\pi\})\}), \pi_n\})).$$

can be written, from which

$$c_1(m(\pi_n; n; \{MOVE(\{REVERSE(\{\pi\})\}), \pi_n\})) \geq$$
$$c_1(m(\pi_n; j = 1, \ldots, n; \{MOVE(\{REVERSE(\{\pi\})\}), \pi_n\}))$$

follows. Note that, the right hand side of the last inequality is the objective function value of the best layout after performing all possible individual move operations for workstation π_n in the layout $\pi' = \{\pi, \pi_n\}$. Finally, since the layout itself, namely $\{\pi, \pi_n\}$, and its permutations $\{REVERSE(\pi), \pi_n\}$, $REVERSE(\{\pi, \pi_n\})$ and $REVERSE(\{REVERSE(\pi), \pi_n\})$ are considered as four particular cases in a run of MOVE($\{\pi, \pi_n\}$), which is $m(\pi_n; j = 1, \ldots, n; MOVE(\{REVERSE(\{\pi\})\})$, we can

write

$$m(\pi_n; j = 1, \ldots, n; \{MOVE(\{REVERSE(\{\pi\})\}), \pi_n\}) = MOVE(REVERSE(\{\pi, \pi_n\})).$$

Therefore

$$MOVE(REVERSE(\{\pi, \pi_n\})) = MOVE(REVERSE(\pi'))$$

holds, showing the correctness of the new MR heuristic.

2.5. Computational Results

In this section we discuss the accuracy and efficiency of the heuristics introduced in Section 2.3 and Section 2.4. We first explain how we generate random part flow matrices to form our test bed. We then compare our new heuristics described in Section 2.4 with the heuristic procedures of Section 2.3.

2.5.1. Test Bed

In an FMS environment workstations, which are interconnected by a material handling system, can process different part types simultaneously. In our experiments we consider four different FMS environments with respectively 20, 30, 40 and 50 workstations interconnected by a unidirectional cyclic material handling system. We fix the number of different part types to 50.

Let $\mathcal{P}= \{1, \ldots, P\}$ be the set of part types and f_{ij}^p be the total number of parts of type p that flow from station i to station j per period of time. Notice P=50 in our test bed. Then,

$$f_{ij} = \sum_{p \in \mathcal{P}} f_{ij}^p \quad i, j = 1, \ldots, n; \; i \neq j \tag{2.57}$$

determines the part flow per time period from station i to station j. When the flow is balanced f_{ij}^p can be expressed as

$$f_{ij}^p = n_p n_{ij}^p \quad i,j = 1,\ldots,n;\ i \neq j; \qquad p \in \mathcal{P} \tag{2.58}$$

where n_p denotes the number of parts of type p to be processed in the system and n_{ij}^p denotes the number of moves part type p makes from station i to station j per period of time, since no part is lost and no new part is created during the process to cause an unbalance in the flow. Each part type may have a different process plan. The process plan S_p is the sequence in which part type p visits the workstations. Notice that in any process plan a part can visit a workstation more than once, and hence a workstation's number can appear more than one time in S_p, but not consecutively. Hence, n_{ij}^p specifies the number of times workstations i and j appear consecutively (i immediately before j) in the process plan S_p. We have generated randomly 30 process plans, for each one of 50 part types, for our four FMS environments. For this purpose we first created uniform positive integer numbers between 1 and the number of workstations, (i.e. 20, 30, 40 or 50) to determine the number of workstations to be visited by this part type. This actually gives the size of a linear array representing a process plan, which we fill entry by entry by uniform positive integers between 1 and the number of workstations while preventing the same integer to occur more than one time in row. This makes $30\times\ 50\times\ 4 = 600$ process plans at sum; and they are all used in the generation of both balanced and unbalanced instances.

As it can be observed when the part flow is balanced formula (2.58) applies and the value of n_p has a direct effect to the magnitude of f_{ij}. Hence, it is possible to control the range of numbers in the part flow matrix by means of n_p. We generate three sets of n_p with $p = 1,\ldots,50$ respectively from uniform distributions *U(1,10)*, *U(1,50)* and *U(1,100)* as done by Tansel and Bilen [21]. They correspond to low, medium and high variation part flows. We will mark the instances with letters L, M and H to identify the type of variation they have, in the sequel. We first use formula (2.58) then formula (2.57) to obtain balanced part flows between workstation pairs. We generate 10 test problems per variation type.

As for the unbalanced part flow, formula (2.58) does not apply anymore since the number of part type p parts, n_p, does not necessarily remain fixed during the process. As a result we generate n_{ij}^p random integers, and sum them up to obtain f_{ij}^p for every part type p. Then we use formula (2.57) to obtain unbalanced part flows between workstation pairs. We use uniform distributions *U(1,10)*, *U(1,50)* and *U(1,100)* for generating those random integers to create low, medium and high variation part flows.

In summary our test bed consists of the set of balanced and unbalanced instances. In each set, instances are grouped according to the type of variation their flows have. Each group is formed by four 10-problem packages respectively with 20, 30, 40 and 50 workstations. For example the package `UnBal20-M` consists of ten test problems each with 20 workstations and medium variation in part flows. The package `Bal20-M` is its balanced version. In short, there are 80 test instances for each flow type, which makes a test bed of 240 instances.

2.5.2. Discussion of the Results

Our computational results are summarized in Tables 2.1 – 2.4. Table 2.1 and Table 2.2 include average relative per cent deviations of the ten-problem test sets, for unbalanced and balanced instances respectively. Table 2.3 and Table 2.4 list average CPU times. The first columns of the tables consist of the test sets. Each one of the remaining columns is associated with one of the heuristics explained in the previous sections. Columns representing new heuristics are emphasized with bold. The last three rows of Table 2.1 and Table 2.2 are respectively the column averages (**UnBalAver**, **BalAver**), and the averages of the smallest (**UnBalMinAver**, **BalMinAver**) and largest (**UnBalMaxAver**, **BalMaxAver**) relative deviations. They are obtained with the ten instances of the test sets. The last rows of Table 2.3 and Table 2.4 are simply the averages of the average CPU times reported in each column. The columns of Table 2.1 and Table 2.2 are in decreasing order of the column averages. However, the ones in Table 2.3 and Table 2.4 are in increasing order of their averages. We believe this exposes better the performances of the heuristics.

The relative per cent deviations are computed using the formula

$$100 \times \frac{(Upper\ Bound\ -\ Lower\ Bound)}{(Lower\ Bound)}$$

where, *Lower Bound* is the optimal objective value of the EUCLP'$_{LP}$ and *Upper Bound* is the total flow cost obtained by the corresponding heuristic. We have used Cplex ver. 7.0's barrier solver to solve the EUCLP'$_{LP}$. In all of them, only OI and KK3 are construction heuristics and build gradually a feasible layout. The remaining ones are improvement methods and need an initial layout to start. We use KK3 for this purpose since it performs 9.3 per cent better than OI in the average (i.e. the average of both unbalanced and balanced instances). Although LHC can be treated as an improvement heuristic it also includes a construction phase as its first step. Hence, there is no need to run KK3 to construct an initial layout for LHC.

The cost function c_1 is used in all computations. However, we take advantage of the equivalence between c_1 and c_2 for balanced instances and use formulae (2.9) or (2.10) to prepare Table 2.1 and Table 2.2. This increases the efficiency considerably. We run all of the heuristics on both unbalanced and balanced instances although some of them are originally proposed for solving the BUCLP. Two such examples are MOVE and its close relative MR. RE and EX are two others; they exploit the relation between the BUCLP and ATSP.

Based on **UnBalAver** values, MR is the most accurate method for the unbalanced instances and has an average (average of test set averages) relative per cent deviation 2.61 per cent. It has also the highest accuracy for the balanced instances. Its average relative deviation, namely **BalAver** value, is 2.72 per cent. As a consequence, the overall accuracy of MR is the highest; its overall average ((**UnBalAver** + **BalAver**)/2) relative deviation is 2.67 per cent. RE, MOVE and EX follow MR in this order with overall average relative deviations respectively, 3.08, 3.26 and 4.61 per cent. However, we should point out that **UnBalMinAver** of RE is 1.62 per cent, which is slightly lower than 1.63 per cent of MR. In addition, while looking at the ranges of **UnBalMinAver** and **UnBalMaxAver** in Table 2.1, and ranges of **BalMinAver**

and **BalMaxAver** in Table 2.2 it is possible to say that MR is the most robust of all the heuristics: the ranges are respectively 2.20 for unbalanced and 2.46 for balanced instances.

Within this four heuristics MOVE is the fastest. Its overall average CPU is 0.35 seconds. EX's efficiency is very low with a value of 6.73 CPU seconds in the average. Although they have the third and fourth highest average CPU seconds, which are 2.24 and 1.27, RE and MR can also be considered quite efficient. Note that the computational efforts of OI and KK3 are significantly smaller (almost none) than the ones of the remaining methods, since they are construction methods and have $O(n)$ worst case time complexity.

Comparing **UnBalAver** and **BalAver** values it seems impossible to draw a general conclusion about the effect of balancedness on the accuracy. Some of the heuristics perform better on balanced instances, while some do on unbalanced problems. Nevertheless, it is possible to say that usually the heuristics tend to have higher accuracy when the problem is balanced. Similar conclusion holds for their efficiency.

We can say that variation in part flow matrix does not affect the quality of the heuristics. However, computation time increases with increasing variation. As a summary of our experimental results, among all of the considered heuristics, we can say rank one goes to MR. It has a very high performance and requires relatively moderate computational effort is very high.

Table 2.1. Relative per cent deviations from the $EUCLP'_{LP}$ lower bound: Unbalanced instances

Instance	OI	KK3	SWAP	LHC	SR	3WI	EX	MOVE	RE	MR
UnBal20-L	**9.02**	7.98	4.62	4.61	**3.55**	3.21	**2.18**	1.49	**1.24**	**0.65**
UnBal20-M	**7.85**	6.52	4.61	3.83	**2.98**	3.04	**2.07**	1.07	**0.90**	**0.63**
UnBal20-H	**8.57**	8.04	3.92	4.16	**2.78**	2.41	**2.33**	1.24	**1.18**	**0.71**
UnBal30-L	**11.67**	10.52	5.97	6.24	**4.29**	4.71	**4.11**	2.79	**2.55**	**2.06**
UnBal30-M	**12.23**	10.89	7.12	5.80	**4.24**	4.53	**3.94**	2.09	**1.90**	**1.45**
UnBal30-H	**9.80**	9.82	5.81	6.35	**3.61**	4.20	**3.54**	2.15	**1.78**	**1.18**
UnBal40-L	**13.96**	13.49	8.47	6.88	**6.13**	5.89	**4.91**	3.97	**3.72**	**3.48**
UnBal40-M	**13.75**	12.78	7.83	7.69	**6.40**	5.82	**4.96**	3.43	**3.39**	**2.98**
UnBal40-H	**14.34**	12.86	7.59	8.22	**6.28**	6.26	**5.66**	4.11	**3.82**	**3.51**
UnBal50-L	**16.17**	15.77	10.82	9.78	**8.32**	7.14	**6.99**	5.32	**5.16**	**4.91**
UnBal50-M	**15.13**	15.47	8.84	8.96	**7.67**	6.93	**6.81**	5.36	**4.95**	**4.72**
UnBal50-H	**16.06**	15.54	10.41	10.54	**7.82**	8.24	**7.04**	6.08	**5.82**	**5.08**
UnBalAver	**12.38**	11.64	7.17	6.92	**5.34**	5.20	**4.55**	3.26	**3.03**	**2.61**
UnBalMinAver	**9.53**	9.07	5.02	4.25	**3.49**	2.84	**2.73**	1.84	**1.62**	**1.63**
UnBalMaxAver	**15.68**	14.39	9.31	9.57	**7.67**	7.05	**6.63**	4.86	**4.77**	**3.83**

Table 2.2. Relative per cent deviations from the $EUCLP'_{LP}$ lower bound: Balanced instances

Instance	KK3	SWAP	LHC	SR	3WI	EX	MOVE	RE	MR
Bal20-L	5.66	3.89	4.84	**3.00**	2.57	**2.42**	0.90	**0.90**	**0.79**
Bal20-M	5.44	4.12	4.31	**2.25**	2.39	**2.28**	0.99	**0.99**	**0.48**
Bal20-H	5.63	4.15	4.21	**2.81**	2.74	**2.51**	1.62	**1.44**	**0.81**
Bal30-L	7.18	5.12	6.26	**4.87**	4.31	**3.59**	2.13	**1.95**	**1.93**
Bal30-M	8.07	5.70	6.16	**4.45**	4.24	**3.45**	2.38	**2.01**	**1.76**
Bal30-H	8.31	6.35	6.38	**4.97**	4.37	**3.85**	2.52	**2.19**	**1.60**
Bal40-L	10.20	7.62	7.97	**5.80**	5.98	**5.15**	4.39	**4.38**	**3.58**
Bal40-M	10.49	7.80	7.75	**6.31**	5.94	**5.90**	3.65	**3.46**	**2.99**
Bal40-H	10.23	7.46	7.61	**6.36**	5.81	**5.11**	3.39	**3.36**	**3.35**
Bal50-L	11.89	9.34	9.66	**8.18**	7.60	**7.14**	5.64	**5.60**	**4.81**
Bal50-M	12.70	9.48	9.52	**8.02**	7.55	**7.29**	5.78	**5.57**	**5.27**
Bal50-H	12.65	10.24	9.55	**8.26**	7.64	**7.36**	5.66	**5.56**	**5.29**
BalAver	9.04	6.77	7.02	**5.44**	5.10	**4.67**	3.25	**3.12**	**2.72**
BalMinAver	6.88	4.60	4.63	**3.64**	2.93	**1.85**	1.96	**1.90**	**1.59**
BalMaxAver	11.31	8.73	9.46	**7.17**	7.08	**6.39**	4.97	**4.79**	**4.05**

Table 2.3. Average CPU times: Unbalanced instances

Instance	OI	KK3	SR	SWAP	MOVE	LHC	MR	RE	3WI	EX
UnBal20-L	**0.00**	0.00	**0.00**	0.01	0.02	0.03	**0.08**	**0.15**	0.29	**0.41**
UnBal20-M	**0.00**	0.00	**0.00**	0.02	0.03	0.04	**0.11**	**0.19**	0.33	**0.54**
UnBal20-H	**0.00**	0.00	**0.00**	0.01	0.03	0.04	**0.12**	**0.18**	0.38	**0.59**
UnBal30-L	**0.00**	0.00	**0.00**	0.04	0.08	0.11	**0.27**	**0.43**	0.84	**1.14**
UnBal30-M	**0.00**	0.00	**0.00**	0.06	0.08	0.13	**0.34**	**0.44**	0.95	**1.29**
UnBal30-H	**0.00**	0.00	**0.00**	0.05	0.09	0.13	**0.38**	**0.51**	0.91	**1.47**
UnBal40-L	**0.00**	0.00	**0.02**	0.33	0.39	0.51	**1.48**	**2.94**	5.72	**9.59**
UnBal40-M	**0.00**	0.00	**0.02**	0.34	0.44	0.57	**1.61**	**3.36**	5.45	**11.15**
UnBal40-H	**0.00**	0.00	**0.03**	0.37	0.48	0.64	**1.74**	**3.51**	6.08	**12.49**
UnBal50-L	**0.00**	0.00	**0.04**	0.99	1.14	1.58	**3.58**	**6.12**	15.73	**16.81**
UnBal50-M	**0.00**	0.00	**0.05**	1.06	1.21	1.54	**3.87**	**6.57**	16.1	**18.37**
UnBal50-H	**0.00**	0.00	**0.07**	1.11	1.35	1.69	**4.24**	**7.12**	17.19	**20.7**
Average	**0.00**	0.00	**0.02**	0.37	0.45	0.58	**1.49**	**2.63**	5.83	**7.88**

Table 2.4. Average CPU times: Balanced instances

Instance	KK3	SR	SWAP	MOVE	LHC	MR	RE	3WI	EX
Bal20-L	0.00	**0.00**	0.00	0.01	0.02	**0.06**	**0.11**	0.24	**0.28**
Bal20-M	0.00	**0.00**	0.00	0.01	0.02	**0.07**	**0.13**	0.29	**0.37**
Bal20-H	0.00	**0.00**	0.01	0.02	0.03	**0.08**	**0.15**	0.31	**0.44**
Bal30-L	0.00	**0.00**	0.02	0.04	0.07	**0.18**	**0.33**	0.65	**0.87**
Bal30-M	0.00	**0.00**	0.02	0.05	0.09	**0.24**	**0.37**	0.71	**1.08**
Bal30-H	0.00	**0.00**	0.05	0.07	0.09	**0.29**	**0.43**	0.85	**1.26**
Bal40-L	0.00	**0.01**	0.17	0.27	0.41	**1.14**	**1.98**	4.19	**5.76**
Bal40-M	0.00	**0.01**	0.18	0.29	0.39	**1.23**	**2.21**	4.37	**7.15**
Bal40-H	0.00	**0.01**	0.20	0.28	0.44	**1.38**	**2.45**	4.8	**7.84**
Bal50-L	0.00	**0.02**	0.40	0.67	0.88	**2.18**	**4.08**	11.86	**12.73**
Bal50-M	0.00	**0.03**	0.45	0.73	0.95	**2.61**	**4.59**	11.03	**13.42**
Bal50-H	0.00	**0.04**	0.63	0.71	1.05	**3.22**	**5.37**	13.25	**15.71**
Average	0.00	**0.01**	0.18	0.26	0.37	**1.06**	**1.85**	4.38	**5.58**

3. AN EXACT ALGORITHM FOR THE BALANCED UNIDIRECTIONAL CYCLIC LAYOUT PROBLEM

3.1. Introduction

The UCLP is defined in Chapter 1: It concerns with the determination of the locations of workstations around a circular material handling system moving the parts unidirectionally. One special form of the UCLP is when the material flow is conserved at each workstation: Total inflow is equal to total outflow at every workstation. This version is known as the BUCLP.

In this chapter we propose a branch and bound algorithm to obtain the optimum solution of the BUCLP. To the best of our knowledge, there is no other work proposing a branch and bound algorithm for the BUCLP. Only, Lee *et al.* [25], Kouvelis and Kim [1] and Kıran and Karabati [23] have devised branch and bound approaches for the general UCLP.

3.2. The Branch and Bound Algorithm

In general there are two major procedures of any branch and bound algorithm: branching and bounding. The branching procedure partitions the problem into smaller sub-problems. Each subproblem, represented by a node, corresponds to a partial solution. A search strategy is associated with the branching scheme in order to decide which node to branch next. Depth first search and breadth first search are two of them. The bounding procedures are used to calculate the lower and upper bound values for each node. For a minimization type problem, the best upper bound value is kept throughout of the algorithm and this value is updated when a new node yields a better upper bound value.

At the beginning of our branch and bound algorithm, we have an empty sequence of workstations and when the algorithm stops we obtain a complete sequence of work-

stations such that total material transportation cost is minimized. The first (last) position of the sequence corresponds to the first (last) candidate location immediately after (before) the Load/Unload area. At every node of the branching tree, we have two sets of workstations: assigned and unassigned. The set of assigned workstations are kept in a partial sequence, with fixed positions. However, the positions of the set of unassigned workstations are not determined.

When a new node is generated, one workstation from the unassigned set is chosen and included into the assigned set. In other words, every time a new node is generated in the branching tree we assign a workstation from the unassigned set, to the first available location of the partial sequence. The partial sequence of a node is kept of all of its descendant branching nodes. That is to say the partial sequence of a node is always inherited from its ancestors. For every node a new partial sequence is obtained by choosing and locating an unassigned workstation to the first available position in the inherited partial sequence.

Moreover, for each node, we compute the lower and upper bound considering the assigned and unassigned sets of workstations. To compute the lower bound, we first compute the cost of assigned workstations. Then, we consider the costs due to the set of unassigned workstations. The upper bound value is the objective function value of the solution obtained with the heuristic procedure. During the search process, we keep the best upper bound value of all generated nodes.

For any node in the tree, if the lower bound is greater than the best known upper bound we fathom this node. If all nodes are discarded, the optimum solution has been found and the current best upper bound value is the optimum objective value. Otherwise, if we have more than one nodes to explore, we select a node with the lowest lower bound value for further branching. Then, we perform a depth first search until we reach the bottom of the tree.

In summary, at the start of the algorithm, n nodes are generated. For each of them, there is only one assigned workstation and its location is fixed at the first po-

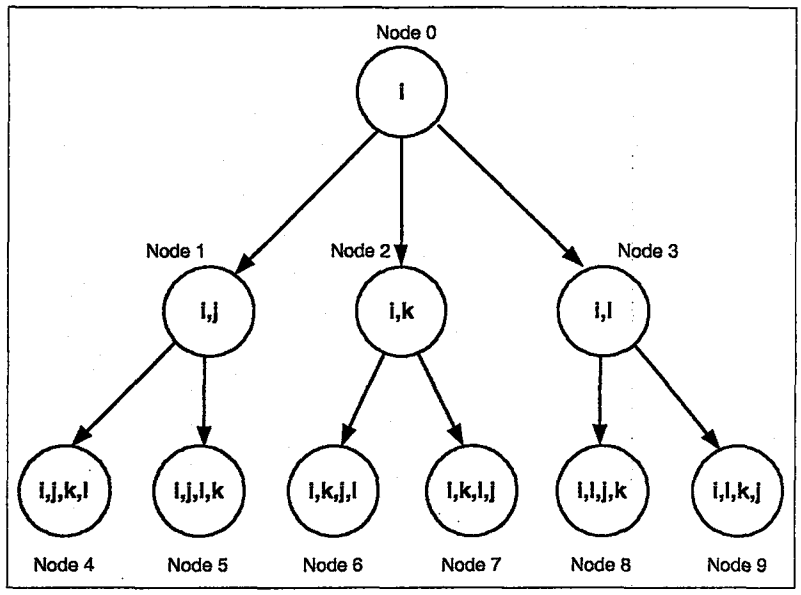

Figure 3.1. The enumeration tree of a 4-workstation example

sition of the corresponding partial sequence. Then, we run branching and bounding procedures consecutively. The lower and upper bound values of new nodes are computed as explained above. The nodes whose lower bound values are greater than the current best upper bound values are discarded for further consideration. Then, the node with the lowest lower bound value is chosen for further branching to generate new nodes. These steps continue until we have only one node with equal lower and upper bound values.

To illustrate these steps, we present in Figure 3.1 a branch and bound subtree for a unidirectional cyclic layout with four workstations. At the first node, namely at Node 0, we assign workstation i into the first location. In other words, we have the partial sequence $\{i\}$ for Node 0. Node 1, Node 2 and Node 3 are in the second level of the tree and they are branches of Node 0. Workstations j, k and l are respectively assigned into the second available locations. Their partial sequence are respectively $\{i,j\}$, $\{i,k\}$ and $\{i,l\}$. Node 4, Node 5, Node 6, Node 7, Node 8 and Node 9 are in the third level of the tree and they represent all possible solutions with workstation i assigned to location 1.

3.2.1. Preprocessing Steps

The preprocessing steps constitute the initialization phase of our branch and bound algorithm. These procedures are run at the very beginning of the algorithm with the purpose to reduce the size of the material flow matrix, and therefore the size of the problem.

Kouvelis and Kim have proposed the preprocessing steps in their seminal paper [1] on the UCLP. In their work, to reduce the size of the problems, the preprocessing phase looks for the workstations which have only inflows and/or outflows. They have shown that in an optimal solution, a workstation i that has only inflow (outflow) is always located at the last (first) candidate location in an optimal sequence. As a consequence, these workstations should be located at the appropriate locations at the beginning.

In addition to this rule, Lee *et al.* [25] have proposed two more rules based on the properties of the material flow matrix which they apply at the beginning of the algorithm. Their first rule is to detect all workstations i with $f_{ij} = f_{ji}$ for all $j = 1, \ldots, n$. They have shown that, the optimal layout is independent of the position of such a workstation. Their second rule is related with the number of workstations with $R_i > C_i$. When the number of workstations with $R_i > C_i$ is greater than the number of workstations with $C_i > R_i$ they propose to construct the layout by assigning the workstations starting from the first position. For the other case, they suggest to locate the workstation starting from the last position. Notice that, this rule is not applicable for the BUCLP since $R_i = C_i$ for all $i = 1, \ldots, n$.

3.2.2. Lower Bounding

To compute the lower bound values we need to consider three flow quantities: Flows between assigned workstations, flows between unassigned-assigned workstation pairs, and flows between unassigned workstations. To calculate a bound in the latter case may be more complicated than the first two. To provide a bound, the LP relaxation of the formulation by Kıran *et al.* [16] has to be solved. The flows between

assigned workstations and the flows between unassigned-assigned workstations pairs can be directly computed since the actual position of assigned workstations is known. In fact, to compute directly these two flow quantities we use the following observation. For a given sequence of workstations, the workstation located in the first position contributes to the objective function value as much as the sum of the inflows from all other workstations. Namely, if workstation i is located in the first position, the contribution to the total cost due to workstation i is $C_i = \sum_{k,k\neq i} f_{ki}$ where f_{ki} is the fixed flow quantity from workstation k to workstation i. Moreover suppose that, workstation j is located in the second position of that partial sequence. Then, the contribution to the total objective function is $(\sum_{k,k\neq j} f_{kj}) - f_{ij} = (C_j) - f_{ij}$. In other words, it is equal to the total amount of flow to workstation j except the inflow quantity from workstation i to workstation j.

To explain better this step, consider a sequence $\{i,k,l,j\}$ of four workstations. Our objective it to calculate the objective function $\sum_i \sum_j f_{ij} d_{ij}$. Recall that $d_{ij} = 1$ if workstation i is located after workstation j and the LUL area is located at some position on the path from workstation i to workstation j. For the given sequence of workstation the objective function value is

$$C_i + (C_k - f_{ik}) + (C_l - f_{il} - f_{kl}) \tag{3.1}$$

where C_i is the sum of column i of the flow matrix, i.e. $C_i = \sum_{j,j\neq i} f_{ji}$.

Now, we can formally present our lower bounding procedure. For any workstation j, $j = 1,\ldots,n$, let $F^j_{\{i\}}$ be the amount of part flow into workstation j, except f_{ij} where $\{i\}$ is the workstation located before workstation j. In other words, $F^j_{\{i\}} = C_j - f_{ij}$ where C_j is the i^{th} column's sum of the part flow matrix. Consider the case where a sequence of two workstations, namely $\{i,k\}$, is located before workstation j. The amount of part flows into workstation j, except f_{ij} and f_{kj}, is $F^j_{\{i,j\}} = C_j - f_{ij} - f_{kj}$. For a partial sequence of workstations $\{i,j,k\}$ we have the lower bound $LB = C_i + F^j_{\{i\}} + F^j_{\{i,k\}} + lb(k)$, where $l(bk)$ is the bound obtained by solving the linear programming formulation by Kıran *et al.* [16], with workstations i, j and k deleted. For instance

consider the branch and bound subtree of Figure 3.1. For Node 0 and Node 1 the lower bounds are respectively, $LB_0 = C_i + lb(i)$ and $LB_1 = C_i + (C_j - f_{ij}) + lb(i,j)$. For Node 4, the lower bound is

$$\begin{aligned} LB_4 &= C_i + (C_j - f_{ij}) + (C_k - f_{ik} - f_{jk}) + lb(i,j,k) \\ &= C_i + (C_j - f_{ij}) + (C_k - f_{ik} - f_{jk}) + (C_l - f_{il} - f_{jl} - f_{kl}) \\ &= C_i + (C_j - f_{ij}) + (C_k - f_{ik} - f_{jk}). \end{aligned}$$

3.2.3. Upper Bounding

To compute the upper bounds, we temporarily complete the partial sequence with the set of unassigned workstations. To assign the unassigned workstations into candidate (available) locations in the rest of the partial sequence, we use the KK3 heuristic. As a result we have a complete sequence of workstations. Then, we perform the Move-Reverse (MR) heuristic to improve the newly inserted set of workstations. Notice that, we perform the KK3 and MR without harming the initial partial sequence.

To better illustrate this, consider Node 1 in Figure 3.1. In Node 1, the set of assigned workstations i and j, are respectively assigned in the first and second available locations in the partial sequence. To compute the upper bound value of Node 1 we need to obtain a complete sequence of all four workstations. In other words, we need to temporarily locate the unassigned workstations k and l to the candidate positions, namely the third and fourth positions. We first run KK3 to construct a sequence then MR to improve the layout for the rest of the partial sequence. Suppose that MR outputs $\{l,k\}$. Then the feasible solution can be represented with the solution $\{i,j,l,k\}$ and the upper bound value is $\mathrm{UB}_1 = C_i + (C_j - f_{ij}) + (C_l - f_{il} - f_{jl})$.

3.2.4. Dominance Rules

Dominance rules help to identify the set of dominant sequences. Given two sequences π and $\overline{\pi}$, π dominates $\overline{\pi}$, means that the objective function value obtained with π is better than the one obtained with $\overline{\pi}$.

In this section we first present existing dominance rules, which are based on the notion of location interchanges. Then we propose a new dominance rule based on the notion of move operations. An interchange operation is the basic action of the famous SWAP heuristic which is widely used for solving the facility layout problems. In Chapter 2, we have also computationally experimented and compared its performance with other existing and new heuristics. On the other hand, a move operation is more elementary than an interchange operation; and every interchange operation can be performed by two move operations. In Chapter 2 we have observed that MOVE heuristic is more efficient than SWAP heuristic. Now, consider the initial sequence

$$\pi = (1, ..., i-1, i, i+1, ..., j-1, j, j+1, ..., n) \tag{3.2}$$

and the sequence obtained by interchanging the locations of workstations i and j

$$\overline{\pi} = (1, ..., i-1, j, i+1, ..., j-1, i, j+1, ..., n). \tag{3.3}$$

Then the change in the cost function is

$$\Delta c_2 = c_2(\overline{\pi}) - c_2(\pi) = \sum_{k=i+1}^{j-1} (f_{kj} - f_{jk}) + (f_{ik} - f_{ki}) + f_{ik} - f_{ji}. \tag{3.4}$$

where c_2 is the objective function of the UCLP. Recall that the derivation of (3.4) is explained in Section 2.2. Clearly when Δc_2 is positive then we have a positive gain from this interchange operation. Kouvelis and Kim [1] uses (3.4) to develop new dominance rules for identifying local optimal solutions. Based on these rules they have devised three construction heuristics: KK1, KK2 and KK3. KK3, is the best of them; it is

presented in the previous chapter. Later, Kıran and Karabatı [23] have used formula (3.4) to propose General Dominance (GD) and Adjacent Dominance (AD) rules which they used within their branch and bound algorithm. The GD rule is applied within the branch and bound algorithm when a new node in the branching tree is generated. Every time a new node is generated, a workstation i from the unassigned set is located at the first available location of the partial sequence. With the GD rule, we first compute all the interchange cost of workstation i with the workstations of the partial sequence, and in case of a positive gain, we do not consider this newly generated node for further consideration. One weakness of the GD is its CPU time requirement. Considering this, Kıran and Karabatı [23] come up with another dominance rule: AD which is a simplified version of the GD rule. They consider only adjacent workstations. In the branch and bound algorithm, the AD rule computes the interchange cost of workstation i and the workstation immediately before workstation i in the partial sequence. In case this value is positive, the new node is fathomed. The AD rule is also used by Lee *et al.* [25] within their branch and bound algorithm.

Both GD and AD are interchange based dominance rules. Now we propose a new move based dominance rule. Our motivation is based on the recent results by Tansel and Bilen [21], and the results of our computational experiments including MOVE and SWAP heuristics presented in Chapter 2. Since MOVE heuristic is more efficient than SWAP heuristic, it is expected that the move based dominance rule is more efficient than GD and AD rules.

In our branch and bound algorithm the move based dominance rule applies when a new workstation i is assigned to the last available position of the partial sequence. We efficiently compute the cost of all possible backward move operations of workstation i through the partial sequence by using formula (2.9). In case a positive gain is detected in the objective function value we do not consider this node for further branching namely, this node is fathomed. For instance consider Node 1. At this node we have assigned workstation j to the available position at the end of the partial sequence $\{i\}$ of Node 0, and we have obtained partial sequence $\{i, j\}$. However, if the cost of partial sequence $\{j, i\}$, which is obtained by moving workstation j into location i, is better

than the partial sequence $\{i, j\}$, then the partial sequence $\{i, j\}$ may be reordered as $\{j, i\}$ in the optimal sequence.

3.3. Computational Results

Our results are summarized in Table 3.1 and Table 3.2. The first, fourth and seventh columns of both tables stand for instances. These instances are randomly generated as explained in the previous chapter. The second, fifth and eighth columns give the number of nodes explored until the optimal solution is reached. The third, sixth and ninth columns are the total CPU times.

As it can be observed, the average number of nodes explored grows as the variation in part flow increases. For example, for instances with low variation having 20 and 30 workstations, the average number of nodes explored are 119.6 and 665.5 respectively. These values become 121.7 and 925.8 for medium variation, and 214.2 and 1656.9 for high variation instances.

To compute lower bounds, we solve the LP relaxations of the BUCLP formulation by Kıran *et al.* [16] with barrier procedure of Cplex ver. 7.0. This part of the computational effort constitutes the bottleneck of our branch and bound algorithm. The development of more efficient lower bounding procedures will solve this problem.

We could not solve all of the instances of our test bed generated in the Chapter 2, it is because the computational effort to find the optimum solution grows exponentially with the number of workstations. For example, for instances with 20 workstations the overall average CPU time is 37.22, while this value becomes 1454 for instances with 30 workstations. Therefore, we have not been able to test our code on instances with more than 30 workstations.

The relation between UCLP and the Linear Ordering Problem is implied by Afentakis [19], and Potts and Whitehead [30]. There are many other combinatorial optimization problems with similar structures. The permutation flow–shop scheduling

problem and single machine scheduling problem, single row layout problem are only a few of them. Several researchers have devised branch and bound algorithms for these problems. One more research topic is the adaptation of techniques developed for the BUCLP to its relatives.

Table 3.1. Computational Results for 20 workstation instances

Instance	Nodes Explored	CPU	Instance	Nodes Explored	CPU	Instance	Nodes Explored	CPU
Bal20-L1	20	5.02	Bal20-M1	20	5.13	Bal20-H1	20	14.85
Bal20-L2	20	5.21	Bal20-M2	567	117.04	Bal20-H2	426	109.8
Bal20-L3	20	5.06	Bal20-M3	20	4.93	Bal20-H3	50	21.0
Bal20-L4	31	7.54	Bal20-M4	36	8.62	Bal20-H4	65	21.7
Bal20-L5	20	5.27	Bal20-M5	20	4.93	Bal20-H5	257	70.7
Bal20-L6	20	4.98	Bal20-M6	29	7.20	Bal20-H6	123	37.1
Bal20-L7	753	146.03	Bal20-M7	20	5.09	Bal20-H7	576	147.7
Bal20-L8	20	5.22	Bal20-M8	427	90.47	Bal20-H8	20	13.6
Bal20-L9	20	5.08	Bal20-M9	37	8.81	Bal20-H9	521	136.7
Bal20-L10	272	60.81	Bal20-M10	41	9.80	Bal20-H10	84	21.5
Average	119.6	25.02	Average	121.7	26.2	Average	214.2	59.5

Table 3.2. Computational Results for 30 workstation instances

Instance	Nodes Explored	CPU		Nodes Explored	CPU		Nodes Explored	CPU
Bal 30-L1	92	112	B30-M1	927	1334.8	B30-H1	618	940.5
Bal 30-L2	116	165.1	B30-M2	2315	3257.1	B30-H2	983	1218.4
Bal 30-L3	1483	2048.3	B30-M3	365	498.7	B30-H3	1128	1371.1
Bal 30-L4	387	598.3	B30-M4	1025	1428.7	B30-H4	7384	8877.3
Bal 30-L5	3084	4538.4	B30-M5	121	180.8	B30-H5	529	675.5
Bal 30-L6	324	543.3	B30-M6	426	604.5	B30-H6	284	407.5
Bal 30-L7	697	953.4	B30-M7	3479	5248.8	B30-H7	653	910.4
Bal 30-L8	56	101.1	B30-M8	196	307.1	B30-H8	1325	1650.3
Bal 30-L9	173	234.5	B30-M9	257	388.8	B30-H9	1683	2022.0
Bal 30-L10	243	324.8	B30-M10	147	219.80	B30-H10	1982	2458.5
Average	665.5	961.9	Average	925.8	1346.9	Average	1656.9	2053.2

4. ASYMMETRIC TRAVELLING SALESMAN PROBLEM FORMULATIONS

4.1. Introduction

Ever since the publication of the seminal work by Dantzig, Fulkerson and Johnson [13], a *perfect formulation* of the ATSP has been attempted many times. A perfect ATSP formulation is a representation of the ATSP polytope which is ideal in the sense the solution of its linear programming (LP) relaxation gives an optimal tour. Although such formulations could have been obtained only for unrealistically small sized problems (problems with eight cities or less [35]), the effort has resulted in different formulations with varying quality.

As it is the case with most combinatorial optimization problems, methods that solve the ATSP to optimality combine polyhedral results with enumeration. Since the efficiency of enumeration depends on the optimal objective value of a sequence of LP problems obtained by modifying the LP relaxation of an ATSP formulation, the strength of the relaxation determines the quality of the formulation. The strength of the LP relaxation is measured by the value of the minimization objective function at the optimality (i.e. LP bound). Thus it is possible to say that given two formulations the one with larger LP bound has a stronger LP relaxation; it is a better ATSP formulation in short.

The strength of LP relaxations, or equivalently the strength of formulations, can also be determined by using polyhedral information they provide. Suppose, two different formulations F_1 and F_2 that are stated in the same space of variables are given. In addition, assume the objective is of minimization type. Let $P(F_1)$ and $P(F_2)$ be the polyhedra associated with them. If $P(F_1)$ is a proper subset of $P(F_2)$ then F_1 is a better formulation than F_2 since the lower bound obtained by solving the LP relaxation (LP bound) of F_1 is larger than or equal to the one obtained by solving the LP relaxation of F_2. In other words the minimization of the cost function over $P(F_1)$ brings us closer

to the integer optimum. As a consequence, the inclusion of polyhedra determines the strength of linear programming relaxations since this provides an ordering of their LP bounds.

Different formulations of a given problem can be frequently stated in terms of different set of variables, as it is the case with the extended formulations of the ATSP. Say F_3 is a formulation with this property. However, it is possible to project the extended polyhedra $P(F_3)$ of F_3 into the subspace of the original variables without losing any integer solution, namely any of the tours, and compare the lower bound obtained over the projected polyhedron $TP(F_3)$. To be precise given the polyhedron

$$P(F_3) = \{(x, y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} : Ax + By \leq b\}$$

where A, B and $b \in \mathbb{R}^m$, the *projection of* $P(F_3)$ *into the subspace of* x *variables*, or *into* $\mathbb{R}^{n_1}$, is

$$TP(F_3) = \{x \in \mathbb{R}^{n_1} : \text{there exists } y \in \mathbb{R}^{n_2} \text{ such that } (x, y) \in P(F_3)\},$$

and if $TP(F_3)$ is a proper subset of $P(F_1)$ then F_3 is a better formulation since the LP bound it gives is larger than the one F_1 does [36].

Several attempts have been made to compare the existing ATSP formulations. The works by Wong [37], Padberg and Sung [15], Langevin *et al.* [14], Gouveia and Voß [38], Orman and Williams [39] and Gouveia and Pires [40, 41] are examples in this direction. In this chapter we consider a new extended ATSP formulation which is originally proposed for the BUCLP [16].

4.2. Classical Formulations and Their Subtour Elimination Constraints

Many of the valid ATSP formulations follow the general framework of an assignment problem with integrality and subtour elimination constraints of the form:

$$\text{ATSP}: \quad \min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{4.1}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n \tag{4.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n \tag{4.3}$$

$$0 \leq x_{ij} \leq 1 \qquad i,j = 1,\ldots,n;\ i \neq j \tag{4.4}$$

$$x_{ij} = 0,1 \qquad i,j = 1,\ldots,n;\ i \neq j \tag{4.5}$$

$$\{(i,j) : x_{ij} = 1,\ i,j = 2,\ldots,n;\ i \neq j\} \text{ do not contain subtours} \tag{4.6}$$

Binary variables x_{ij} are used to indicate whether city i precedes city j in an optimal tour. Coefficients c_{ij} represent the distance from city i to city j. For simplicity, it can be assumed that the formulation is defined on a complete, simple digraph $D = (V, A)$ with $x_{ij} = 0$ or $c_{ij} = +\infty$ for any arc $(i,j) \notin A$. Constraints (4.6) defeat subtours over the set $\{2,\ldots,n\}$. Together with the assignment constraints (4.2) and (4.3) they also eliminate subtours containing vertex 1; and the subtour elimination constraints for $i = 1$ become redundant.

4.2.1. Dantzig, Fulkerson and Johnson's Formulation

In their seminal work, Dantzig, Fulkerson and Johnson (DFJ) formulate the subtour elimination constraints directly in the space of original variables as [13]

$$\sum_{(i,j)\in S, i\neq j} x_{ij} \leq |S| - 1 \quad S \subseteq \{2,\ldots,n\},\ |S| \geq 2 \tag{4.7}$$

These *clique packing*, or simply *clique*, inequalities state that the number of arcs that can be placed in the clique defined by the set of vertices S cannot exceed $|S| - 1$. These inequalities are known to be facet defining [42]. Unfortunately, there are $O(2^n)$ clique inequalities. However, rather than introducing them to the formulation all at

once, they can be added to the formulation gradually throughout a branch–and–bound scheme, which usually results in an optimal solution after only a subset of them are used.

4.2.2. Miller, Tucker and Zemlin's Formulation

The earliest known extended formulation of the ATSP is due to Miller, Tucker and Zemlin (MTZ). MTZ formulation is originally proposed for a vehicle routing problem where each route is restricted to not having more than a certain number of customers [43]. In this formulation; the number of subtour elimination constraints are reduced from $O(2^n)$ to $O(n^2)$ at the expense of additional variables u_i for $i = 2, \ldots, n$. The subtour elimination constraints of the MTZ formulation are equivalently stated by Desrochers and Laporte [44] as

$$u_i - u_j + (n-1)x_{ij} \leq n-2, \qquad i,j = 2,\ldots,n;\ i \neq j \tag{4.8}$$

$$1 \leq u_i \leq n-1 \qquad i = 2,\ldots,n \tag{4.9}$$

Constraints (4.8) imply $u_j \geq u_i + 1$ whenever $x_{ij} = 1$, and the value of each u_i can be treated as the visit order of city i from city 1 on the optimal tour. Given a circuit $D_C = (V_C, C)$, with vertices V_C and arcs C, summing up equation set (4.8) for $(i,j) \in C$ yields the contradiction $2 \leq 1$. Notice that the same contradiction is also obtained when D_C is an Hamiltonian circuit, which means that the formulation is infeasible. However, this case does not occur since the MTZ formulation is concerned only with subtours of size smaller than $n-1$. We should mention that u_i variables are unrestricted in the original paper [43]. Simple bounds (4.9) are introduced later on. This enables u_i to present the order vertex i is visited on a tour, and does not affect the LP bound obtained on P(MTZ).

4.2.3. Circuit Packing Formulation

The subset of clique inequalities known as the *circuit packing* or simply *circuit* inequalities, given by

$$\sum_{(i,j)\in C} x_{ij} \leq |C| - 1 \quad \text{for all circuits } D_C = (V_C, C),\ V_C \subseteq \{2, \ldots, n\} \tag{4.10}$$

are sufficient to eliminate all subtours and the formulation obtained by replacing clique inequalities (4.7) with circuit inequalities (4.10) is a valid ATSP formulation; it is referred as the *circuit packing* (CP) formulation. As it can be noticed the polytope P(DFJ) described by (4.2)–(4.4) and (4.7), is a proper subset of the polytope P(CP) represented by (4.2)–(4.4) and (4.10). This implies that the DFJ formulation has a stronger LP relaxation, and hence a larger LP bound, than CP formulation does.

Let P(MTZ) be the polyhedron described by (4.2)–(4.4), (4.8) and (4.9). In their work on the analytical comparison of different ATSP formulations Padberg and Sung [15] have shown that the polyhedron obtained by projecting P(MTZ) into the subspace of x_{ij} variable is

$$TP(MTZ) = \{x \in \mathbb{R}^{|A|} : (4.2) - (4.4), \text{and} \sum_{(i,j)\in C} x_{ij} \leq |C| - \frac{|C|}{n-1}$$
$$\text{for all circuits } D_C = (V_C, C),\ V_C \subseteq \{2, \ldots, n\}\}. \tag{4.11}$$

TP(MTZ) is also known as the *weak circuit polytope* because (4.11) are a weaker version of the circuit inequalities (4.10). This result shows that P(CP) is a proper subset of TP(MTZ), therefore MTZ formulation is weaker than CP and DFJ formulations. The weakness of MTZ formulation is also demonstrated by Langevin *et al.* in their early work in which they classify nine ATSP formulations with respect to the LP bounds they produce [14]. Moreover, using the Fourier–Motzkin elimination to project several ATSP formulations into a common variable space, Orman and Williams show that P(MTZ) contains the polyhedra of some of seven existing formulations [39].

4.2.4. Desrochers and Laporte's Formulation

The compact polynomial representation of P(MTZ) is particularly useful when the ATSP arises as a subproblem within the context of a larger problem. Two such examples are the *distance constrained capacitated vehicle routing problem* (DCVRP) and the *capacitated vehicle routing problem with time windows* (TWCVRP) [45]. The DFJ formulation can also be extended to the CVRP [46]. However, its generalizations to the DCVRP and TWCVRP are not straightforward. Another advantage of the MTZ formulation is that the subtour elimination constraints (4.8) and (4.9) can be incorporated into other type of problem formulations together with stronger constraints. Motivated by these facts, Desrochers and Laporte lifted subtour elimination constraints (4.8) and (4.9) to obtain the following stronger ones [44]:

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2, \qquad i,j = 2,\ldots,n;\ i \neq j \qquad (4.12)$$

$$1 + (n-3)x_{i1} + \sum_{j=2,j\neq i}^{n} x_{ji} \leq u_i \leq n-1-(n-3)x_{1i} - \sum_{j=2,j\neq i}^{n} x_{ij}$$

$$i = 2,\ldots,n. \qquad (4.13)$$

They have shown that inequalities (4.12) are facet defining. Their proof relies on the early result of Grötschel and Padberg [42] who demonstrated that clique inequalities are facet defining for $|S| = 2$ and $n \geq 6$. They also showed that (4.13) are valid inequalities. Later, Driscoll proved that (4.13) are actually facet defining [47].

4.2.5. Sherali and Driscoll's Formulation

The concern of Sherali and Driscoll's recent paper [48] is also strengthening the relaxations of MTZ formulation for the ATSP. The authors apply only a partial first level version of the *Reformulation-linearization technique* (RLT) [49] to a nonlinear

reformulation of MTZ formulation's subtour elimination constraints to obtain

$$\sum_{j=2,j\neq i}^{n} y_{ij} + (n-1)x_{i1} = u_i \qquad i = 2,\ldots,n \tag{4.14}$$

$$\sum_{i=2,i\neq j}^{n} y_{ij} + 1 = u_j \qquad j = 2,\ldots,n \tag{4.15}$$

$$x_{ij} \leq y_{ij} \leq (n-2)x_{ij} \qquad i,j = 2,\ldots,n; i \neq j \tag{4.16}$$

$$u_j + (n-2)x_{ij} - (n-1)(1-x_{ji}) \leq$$
$$y_{ij} + y_{ji} \leq u_j - (1-x_{ji}) \qquad i,j = 2,\ldots,n;\ i \neq j \tag{4.17}$$

$$1 + (1-x_{1j}) + (n-3)x_{j1} \leq$$
$$u_j \leq (n-1) - (n-3)x_{1j} - (1-x_{j1}) \qquad j = 2,\ldots,n. \tag{4.18}$$

These $O(n^2)$ constraints are shown to be valid subtour elimination constraints and imply (4.12) and (4.13) of the DL formulation [48]. In short the SD formulation is stronger than the DL formulation and has a larger LP bound. Since the new variable $y_{ij} = u_i x_{ij}$, it assumes a nonzero value that represents the order (starting with 0) of the arc (i,j) if it is on a given tour. This value is zero otherwise.

4.2.6. Gavish and Graves' Formulation

A large class of the extended ATSP formulations have been known as the *commodity flow* formulations [14] where the additional variables represent commodity flows through the arcs that satisfy additional flow conservation constraints. These models can be further subdivided into three groups: single commodity flow (SCF), two-commodity flow (TCF) and multi-commodity flow (MCF) formulations.

The earliest SCF formulation is due to Gavish and Graves [50]. The additional continuous nonnegative variables y_{ij} they use describe the flow of a single commodity to vertex 1 from every other vertex. $O(n^2)$ subtour elimination constraints are then

given by

$$\sum_{j=1}^{n} y_{ij} - \sum_{j\neq 1}^{n} y_{ji} = 0 \quad i = 2,\ldots,n \tag{4.19}$$

$$y_{ij} \leq (n-1)x_{ij} \qquad i = 2,\ldots,n;\ j = 1,\ldots,n;\ i \neq j \tag{4.20}$$

$$y_{ij} \geq 0 \qquad i = 1,\ldots,n;\ j = 1,\ldots,n;\ i \neq j \tag{4.21}$$

Constraints (4.20) and (4.21) imply that $0 \leq y_{ij} \leq n-1$ if $x_{ij} = 1$. An interpretation similar to the one given for the additional variables u_i of the MTZ formulation can also be attached to variables y_{ij}; they can be interpreted as the number of arcs which are included in the path between vertex 1 and arc (i,j) in the optimal tour [40].

Let P(GG) be the polyhedron described by (4.2)–(4.4), (4.19)–(4.21). As a consequence of a result for CVRP [51] it can be shown that the polyhedron obtained by projecting P(GG) into the subspace of x_{ij} variables is

$$\begin{aligned} TP(GG) =& \{x \in \mathbb{R}^{|A|} : (4.2) - (4.4), \\ & \text{and} \sum_{(i,j)\in S, i\neq j} x_{ij} \leq |S| - \frac{|S|}{n-1} \quad S \subseteq \{2,\ldots,n\},\ |S| \geq 2\}, \end{aligned} \tag{4.22}$$

which is known as the *weak clique polytope* [40]. Therefore P(DFJ) is a proper subset of TP(GG) and the GG formulation is weaker than the DFJ formulation. However, GG formulation is stronger than MTZ formulation since TP(GG) is a proper subset of TP(MTZ). This follows from the fact that constraint set (4.22) includes also constraints (4.11) as a subset.

It is also possible to construct a relationship between the SD and GG formulations, and the DL and GG formulations. Recall that both have $O(n^2)$ subtour elimination constraints. Consider constraints (4.14)–(4.16) of the SD formulation and observe that the right hand sides of (4.14) and (4.15) are in fact equal. As a result we can write

$$\sum_{j=2,j\neq i}^{n} y_{ij} + (n-1)x_{i1} = \sum_{j=2,j\neq i}^{n} y_{ji} + 1 \quad i = 2,\ldots,n. \tag{4.23}$$

Then by adding this set of inequalities for $S \subseteq V = \{1, \ldots, n\}$ we obtain

$$\sum_{\substack{i \in S \setminus \{1\} \\ j \in S^c \setminus \{1\}}} y_{ij} + (n-1) \sum_{i \in S \setminus \{1\}} x_{i1} = \sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} y_{ij} + |S| \tag{4.24}$$

where $S^c = V \setminus S$. On the other hand the aggregation of constraints (4.16) over S gives

$$(n-2) \sum_{\substack{i \in S \setminus \{1\} \\ j \in S^c \setminus \{1\}}} x_{ij} \geq \sum_{\substack{i \in S \setminus \{1\} \\ j \in S^c \setminus \{1\}}} y_{ij} \tag{4.25}$$

and

$$\sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} y_{ij} \geq \sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} x_{ij}. \tag{4.26}$$

Hence equality (4.24) and inequalities (4.25) and (4.26) imply

$$(n-2) \sum_{\substack{i \in S \setminus \{1\} \\ j \in S^c \setminus \{1\}}} x_{ij} + (n-1) \sum_{i \in S \setminus \{1\}} x_{i1} \geq \sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} x_{ij} + |S|, \tag{4.27}$$

which results in

$$(n-2) Degree(S \setminus \{1\}) + (n-1) \sum_{i \in S \setminus \{1\}} x_{i1} \geq (n-1) \sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} x_{ij} + |S| \tag{4.28}$$

after adding $(n-2) \sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} x_{ij}$ to both sides of (4.27). Here $Degree(S \setminus \{1\}) = \sum_{\substack{i \in S \setminus \{1\} \\ j \in S^c \setminus \{1\}}} x_{ij} + \sum_{\substack{i \in S^c \setminus \{1\} \\ j \in S \setminus \{1\}}} x_{ij}$, namely the number of arcs having exactly one endpoint (head or tail but not both) in the subset $S \setminus \{1\}$. Inequality (4.28) can be equivalently rewritten as

$$(n-2) Degree(S \setminus \{1\}) + (n-1) \sum_{\substack{i \in S \setminus \{1\} \\ j \in S^c}} x_{ij} \geq (n-1) Degree(S \setminus \{1\}) + |S| \tag{4.29}$$

after adding $(n-1)\sum_{\substack{i\in S\setminus\{1\}\\ j\in S^c\setminus\{1\}}} x_{ij}$ to its right and left hand sides from which

$$(n-1)\sum_{\substack{i\in S\setminus\{1\}\\ j\in S^c}} x_{ij} \geq Degree(S\setminus\{1\}) + |S| \tag{4.30}$$

follows. Since

$$\sum_{i,j\in S} x_{ij} + \sum_{\substack{i\in S\\ j\in S^c}} x_{ij} = |S| \tag{4.31}$$

for all subsets S of the set of vertices V, we can write

$$\sum_{i,j\in S\setminus\{1\}} x_{ij} + \sum_{\substack{i\in S\setminus\{1\}\\ j\in S^c}} x_{ij} \leq |S| \qquad S\subseteq V, \tag{4.32}$$

or equivalently

$$\sum_{i,j\in S\setminus\{1\}} x_{ij} \leq |S| - \sum_{\substack{i\in S\setminus\{1\}\\ j\in S^c}} x_{ij} \qquad S\subseteq V. \tag{4.33}$$

Now, by using (4.33) we can rewrite (4.30) as

$$(n-1)\left(|S| - \sum_{i,j\in S\setminus\{1\}} x_{ij}\right) \geq Degree(S\setminus\{1\}) + |S| \tag{4.34}$$

which implies weak clique inequalities

$$|S| - \frac{|S|}{n-1} \geq \sum_{i,j\in S} x_{ij} \qquad S\subseteq\{2,\ldots,n\},\ 2\leq|S|\leq n-1 \tag{4.35}$$

since $Degree(S\setminus\{1\})$ is nonnegative. A direct consequence of this discussion is that the projection of a polyhedron which includes P(SD) is included in TP(GG) and consequently TP(SD) is in TP(GG); the SD formulation is at least as strong as GG

formulation. In addition it is always possible to find an ATSP instance for which the LP bound of the SD formulation, is strictly larger than the one produced by the GG formulation on the same instance. One such example is `att48`: the LP bound the GG formulation gives is 8786.94 compared to 10070.3 for the SD formulation. Therefore TP(SD) is a proper subset of TP(GG) and the SD formulation is stronger than the GG formulation.

As for the relations between the DL and GG formulations, we can say they are incomparable. In order to see this it is enough to consider ATSP instances `ft53` and `kroA124p`. The LP bounds obtained by solving the LP relaxations of the DL and GG formulations are respectively 6045.04 and 6011.33 for `ft53`. However they become 34248 and 34976.7 for `kroA124p`.

4.2.7. Wong's Formulation

Wong is the first to formulate the ATSP as a MCF model [37]. He uses additional nonnegative variables to describe the flow of $2(n-1)$ commodities between vertex 1 and other vertices, namely commodities $Y^k = y_{ijk}$, $k = 2, ..., n$ and commodities $Z^k = z_{ijk}$, $k = 2, ..., n$. His $O(n^3)$ subtour elimination constraints consist of the followings:

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} y_{ijk} - \sum_{\substack{j=1 \\ i \neq j}}^{n} y_{jik} = 0 \qquad i, k = 2, \ldots, n;\ i \neq k \tag{4.36}$$

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} z_{ijk} - \sum_{\substack{j=1 \\ i \neq j}}^{n} z_{jik} = 0 \qquad i, k = 2, \ldots, n;\ i \neq k \tag{4.37}$$

$$\sum_{\substack{i=2 \\ i \neq j}}^{n} y_{1ij} = 1,\ \sum_{\substack{i=1 \\ i \neq j}}^{n} y_{ijj} = 1,\ \sum_{\substack{i=1 \\ i \neq j}}^{n} y_{jij} = 0 \qquad j = 2, \ldots, n \tag{4.38}$$

$$\sum_{\substack{i=2 \\ i \neq j}}^{n} z_{1ij} = 1,\ \sum_{\substack{i=1 \\ i \neq j}}^{n} z_{ijj} = 1,\ \sum_{\substack{i=1 \\ i \neq j}}^{n} z_{jij} = 0 \qquad j = 2, \ldots, n \tag{4.39}$$

$$y_{ijk} \leq x_{ij},\ y_{ijk} \geq 0 \qquad i, j = 1, \ldots, n;\ k = 2, \ldots, n;\ i \neq j \tag{4.40}$$

$$z_{ijk} \leq x_{ij},\ z_{ijk} \geq 0 \qquad i, j = 1, \ldots, n;\ k = 2, \ldots, n;\ i \neq j. \tag{4.41}$$

Constraints (4.36) and (4.38) guarantee that one unit of commodity Y^k travels from vertex 1 to vertex k while one unit of commodity Z^k travels from vertex k to vertex 1. Notice that, only the x_{ij}'s are subject to integrality since z_{ijk} and y_{ijk} variables have integer values whenever the x_{ij}'s are, because the flow matrix for commodities Y^k and Z^k is totally unimodular. In fact the variable x_{ij} can be interpreted as defining a capacity on arc (i,j). Wong has shown that the LP bound of his formulation is equal to the LP bound of the DFJ formulation's and thus both of them have the same strength.

One modification of the Wong's MCF model (WONG) was proposed by Langevin [52]. He replaced constraints (4.37), (4.39) and (4.41), by $y_{ijk} + z_{ijk} \leq x_{ij}$ for i,j,k. A similar modification is proposed by Loulou [53] but, he replaced the constraints (4.37), (4.39) and (4.41), by $y_{ijk} + z_{ijk} = x_{ij}$ for i,j,k. Hence, Langevin's formulation (LANGEVIN) is a restriction of the WONG formulation, and Loulou's formulation (LOULOU) is a restriction of both Langevin's formulation and WONG formulation.

4.2.8. Claus' Formulation

Different than the WONG formulation, Claus' more recent formulation (CLAUS) uses $(n-1)$ commodities [54]. It can be obtained from the WONG formulation by eliminating half of the flow variables and related constraints. Claus defines nonnegative flow variables z_{ijk} to describe the amount of commodity k that flows from vertex 1 to vertex k through arc (i,j). The $O(n^3)$ subtour elimination constraints can be given as follows:

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} z_{ijk} - \sum_{\substack{j=1 \\ i \neq j}}^{n} z_{jik} = 0 \qquad i,k = 2,\ldots,n; i \neq k \tag{4.42}$$

$$\sum_{\substack{i=2 \\ i \neq j}}^{n} z_{1ij} = 1, \sum_{\substack{i=1 \\ i \neq j}}^{n} z_{ijj} = 1, \sum_{\substack{i=1 \\ i \neq j}}^{n} z_{jij} = 0 \qquad j = 2,\ldots,n \tag{4.43}$$

$$z_{ijk} \leq x_{ij} \qquad i,j = 1,\ldots,n;\ k = 2,\ldots,n;\ i \neq j \tag{4.44}$$

$$z_{ijk} \geq 0 \qquad i,j = 1,\ldots,n;\ k = 2,\ldots,n;\ i \neq j \tag{4.45}$$

Constraints (4.42) ensure that the flow is conserved at every vertex except vertex 1, which is the home city, and vertex k, which is the sink of commodity k. Constraints (4.43) ensure exactly 1 unit of outflow of commodity j from vertex 1 to any other vertex i, exactly 1 unit of inflow of commodity j to vertex j, and zero flow of commodity j out of vertex j, respectively. Constraints (4.44) and (4.45) imply that the value of flow of commodity k which can be sent from city 1 to city k is nonnegative and does not exceed 1. Conservation of flow constraints for vertex 1 are not considered because they are redundant in the LP relaxation of the formulation. In this manner, flow variables of SCF and MCF formulations become related:

$$\sum_{k=2}^{n} z_{ijk} = y_{ij} \quad i,j = 1,\ldots,n;\ i \neq j. \tag{4.46}$$

There are two results on the strength of the CLAUS formulation. The first one is due to Langevin *et al.* [14]. They demonstrate that LP bounds of the WONG and CLAUS formulations [37, 54] are equal, which means both formulations have equal strengths. The second one is due to Padberg and Sung [15]. They obtain the projection TP(CLAUS) of polyhedron P(CLAUS) described by constraints (4.2)–(4.4), (4.42)–(4.45) into the subspace of x_{ij} variables and show that it is equivalent to P(DFJ). As a consequence, the CLAUS formulation is as strong as the DFJ and WONG formulations, and its LP bound has the same value as theirs.

4.2.9. Gouveia and Pires' Formulations

4.2.9.1. Generalizations Based on Miller, Tucker and Zemlin Formulation. In their first work on extending the MTZ formulation, Gouveia and Pires show that the set of inequalities

$$x_{ij} + v_{ki} - v_{kj} \leq 1 \quad i,j,k = 2,\ldots,n;\ i \neq j \neq k \tag{4.47}$$

$$x_{ij} - v_{ij} \leq 0 \quad i,j = 2,\ldots,n;\ i \neq j \tag{4.48}$$

$$x_{ij} + v_{ji} \leq 1 \quad i,j = 2,\ldots,n;\ i \neq j \tag{4.49}$$

eliminate subtours and together with constraints (4.2)–(4.5) they give a valid ATSP formulation [40]. This first formulation of Gouveia and Pires is referred as GP1 formulation in the sequel. The GP1 formulation has $O(n^3)$ subtour elimination constraints and becomes infeasible when the subtour elimination constraints for vertex 1 are also included, as is also the case with the MTZ formulation and its extensions the DL and SD formulations. Here v_{ij} are the new additional variables. Inequalities (4.47) and (4.48) imply $v_{ij} = 1$ if vertex i is in the path from vertex 1 to vertex j, and inequalities (4.49) state that arc (i, j) is in the optimal tour if and only if vertex j is not on the path from vertex 1 to vertex i. Recall that u_i of the MTZ formulation can be interpreted as the number of intermediate cities in the path from vertex 1 to vertex i of the optimal tour. However, in this version considered by Gouveia and Pires, u_i goes from 0 to $n-2$, rather than from 1 to $n-1$. This is different than the form of the MTZ formulation considered in the derivation of the DL and SD formulations. The addition of such bounds on the additional variables does not change the LP bound of the original MTZ formulation in which u_i is unrestricted. Here u_i and v_{ij} are related by

$$u_i = \sum_{k=2, k\neq i}^{n} v_{ki} \quad i = 2, \ldots, n. \tag{4.50}$$

Notice that it is possible to obtain subtour elimination constraints (4.8) by adding constraints (4.47) for $k = 2, \ldots, n$, $k \neq i$, and $k \neq j$ first, and constraints (4.48) and (4.49) then, for the same pair (i, j), and finally using relation (4.50).

They also show that the projection of the polyhedron P(GP1) into the subspace of original x_{ij} variables, is the polyhedron

$$TP(GP1) = \left\{ x \in \mathbb{R}^{|A|} : (4.2) - (4.4), \text{ and } (4.10) \right\}.$$

Observing that TP(GP1) is exactly the circuit polytope P(CP), which is a proper subset of TP(MTZ), it is possible to say that the GP1 formulation is stronger than the MTZ formulation.

Circuit inequalities (4.10) can be lifted to obtain new families of facet defining inequalities for the ATSP polytope [40, 55]. Since the projection of the polyhedron P(GP1) into the subspace of x_{ij} variables is the circuit polytope, one interesting question is whether the projection into the subspace of x_{ij} variables of any lifting of inequalities (4.47)–(4.49) is a lifting of the circuit inequalities. Notice that the summing up of constraints (4.48) and (4.49) for pairs (i,j) and (j,i) for $j \neq i$ results respectively in the facet defining clique inequalities (4.7) for $S = \{i,j\}$. In other words any lifting of constraint (4.48) and/or (4.49) would have generated an inequality which is stronger than a facet–defining inequality. Therefore (4.48) and/or (4.49) cannot be lifted. On the other hand constraints (4.47) can be lifted in two different ways [40]:

$$x_{ji} + x_{ij} + v_{ki} - v_{kj} \leq 1 \qquad i,j,k = 2,\ldots,n;\ i \neq j \neq k \tag{4.51}$$

and

$$x_{kj} + x_{ik} + x_{ij} + v_{ki} - v_{kj} \leq 1 \quad i,j,k = 2,\ldots,n;\ i \neq j \neq k. \tag{4.52}$$

New formulations, namely the GP2 and GP3 formulations, which are obtained by replacing (4.47) first with (4.51), and then (4.52) respectively, are valid ATSP formulations. For (4.51); $x_{ji} + x_{ij} \leq 1$ is always true and $v_{ki} = 1$ if and only if $v_{kj} = 1$ when $x_{ji} + x_{ij} = 1$. For (4.52); if $x_{ij} = 1$ then $x_{kj} = x_{ik} = 0$ and $v_{ki} = 1$ if and only if $v_{kj} = 1$. If $x_{ij} = 0$, then (4.52) can be obtained by adding (4.48) for the pair (k,j) with (4.49) for the pair (i,k).

If we sum up constraints (4.51) for $k = 2,\ldots,n$ and $i \neq j \neq k$ for a given pair (i,j), adding the resulting aggregation to (4.48) and (4.49) for the same pair (i,j), and using relation (4.50), then inequalities (4.12) can be obtained [40]. It should be noted that (4.12) can not be lifted further since for any other variable x_{kl}, there is a tight solution to the inequality if $x_{kl} = 0$ or $x_{kl} = 1$.

It is also possible to project the LP relaxations of the GP2 and GP3 formulations,

namely the polyhedrons

$$P(GP2) = \{x \in \mathbb{R}^{|A|} : (4.2) - (4.4),\ (4.48),\ (4.49),\ (4.51)\}$$

and

$$P(GP3) = \{x \in \mathbb{R}^{|A|} : (4.2) - (4.4),\ (4.48),\ (4.49),\ (4.52)\}$$

into the subspace of original x_{ij} variables. Gouveia and Pires [40] have shown that these sets are

$$TP(GP2) = \{x \in \mathbb{R}^{|A|} : (4.2) - (4.4),\ \sum_{(i,j)\in C} x_{ij} + \sum_{\substack{(i,j)\in C \\ i\neq k, j\neq k}} x_{ji} \leq |C| - 1$$

$$k \in V_C,\ \text{for all circuits } D_C = (V_C, C),\ V_C \subseteq \{2, \ldots, n\},\ |C| > 2\} \tag{4.53}$$

and

$$TP(GP3) = \{x \in \mathbb{R}^{|A|} : (4.2) - (4.4),\ \sum_{(i,j)\in C} x_{ij} + \sum_{k\in V_C} (x_{kp} + x_{pk}) \leq |C|$$

$$p \notin V_C,\ p \neq 1 \text{ for all circuits } D_C = (V_C, C),\ V_C \subseteq \{2, \ldots, n\},\ |C| \geq 2\}. \tag{4.54}$$

Both TP(GP2) and TP(GP3) are proper subsets of the circuit polytope P(CP), which is equivalent to TP(GP1). Hence, the GP2 and GP3 formulations are stronger than GP1 formulation and have larger LP bounds. Unfortunately, they are dominated by the DFJ formulation because P(DFJ) is a proper subset of both TP(GP2) and TP(GP3) [40].

In their fourth formulation, which we refer as the GP4 formulation, Gouveia and Pires consider constraints (4.51) and (4.52) together with constraints (4.2)–(4.5), (4.48) and (4.49). The GP4 formulation is clearly stronger than the GP2 and GP3 formulations since it includes the constraints of both the GP2 and GP3 formulations. However, a clear dominance relation between the GP4 and CLAUS formulations does not exist. In fact it is possible to show that the GG and GP1 formulations are not

comparable by finding a feasible $(\overline{x}, \overline{v})$ vector of P(GP1) for which no $\overline{y}$ exists such that $(\overline{x}, \overline{y})$ vector is included in P(GG), and vice-versa [40].

The DL formulation and the GP1 – GP4 formulations are incomparable. The fact that (4.12) is one of the subtour elimination constraints of the DL formulation, and its derivation from constraints (4.48), (4.49), (4.51) and linear transformation (4.50) may convince one to claim that the GP1 formulation is stronger than the DL formulation. Previous computational results support this claim, indeed [40,48]. Unfortunately, this claim is not true and the DL formulation and the GP1 – GP4 formulations are incomparable, in fact. We show this by means of two solution vectors one of which is feasible for P(DL) but infeasible for P(GP1) – P(GP4), and the other which is feasible for P(GP4) but infeasible for P(DL).

Consider

$$
\begin{aligned}
&x_{12} = 0.6, \quad x_{16} = 0.4, \\
&x_{25} = 1, \\
&x_{31} = 0.4, \quad x_{36} = 0.6, \\
&x_{42} = 0.4, \quad x_{43} = 0.6, \\
&x_{54} = 1, \\
&x_{61} = 0.6, \quad x_{63} = 0.4,
\end{aligned}
$$

$$
u_2 = 1.4, \quad u_3 = 3.2, \quad u_4 = 3.4, \quad u_5 = 2.4, \quad u_6 = 3.4,
$$

and all other variables are equal to zero. This is a feasible solution of P(DL). To show that no v_{ij} can exist such that (x, v) is included in P(GP1) let us write inequalities

(4.48) and (4.49) respectively for $(i,j) = (5,4)$ and $(i,j) = (4,5)$:

$$\begin{aligned} x_{54} &\leq v_{54}, \\ x_{54} + v_{45} &\leq 1, \\ x_{45} &\leq v_{45}, \\ x_{45} + v_{54} &\leq 1. \end{aligned}$$

Since $x_{54} = 1$ and $x_{45} = 0$, $v_{45} = 0$ and $v_{54} = 1$ follow. Now we consider (4.47) for $(i,j,k) = (2,5,4)$:

$$x_{25} + v_{42} - v_{45} \leq 1.$$

Using $v_{45} = 0$, $x_{25} = 1$ we obtain $v_{42} \leq 0$, which contradicts (4.48) for $(i,j) = (4,2)$, namely $x_{42} \leq v_{42}$, since $x_{42} = 0.4 \not\leq v_{42} \leq 0$. Therefore no v_{ij} value can exist for these x_{ij} values. Moreover, since P(GP2), P(GP3), and P(GP4) are proper subsets of P(GP1) no v_{ij} can exist for these x_{ij} values so that (x,v) vector is in P(GP2), P(GP3) and P(GP4) can exist. Hence, DL formulation is not stronger than GP1 – GP4 formulations.

Now consider

$$\begin{aligned} x_{13} &= 0.5, \quad x_{15} = 0.5, \\ x_{23} &= 0.5, \quad x_{24} = 0.5, \\ x_{32} &= 0.5, \quad x_{35} = 0.5, \\ x_{41} &= 0.5, \quad x_{42} = 0.5, \\ x_{51} &= 0.5, \quad x_{54} = 0.5, \end{aligned}$$

$$v_{23} = 0.5, \quad v_{24} = 0.5, \quad v_{25} = 0.5,$$
$$v_{32} = 0.5, \quad v_{34} = 0.5, \quad v_{35} = 0.5,$$
$$v_{42} = 0.5, \quad v_{43} = 0.5, \quad v_{45} = 0.5,$$
$$v_{52} = 0.5, \quad v_{53} = 0.5, \quad v_{54} = 0.5,$$

and all other variables are equal to zero. The corresponding (x, v) vector is included in the polyhedron P(GP4). To show that no u_i can exist for these x_{ij} values so that (x, u) vector is in P(DL). Let us write inequalities (4.13) for $i = 2, \ldots, 5$:

$$4 - 2x_{12} - x_{23} - x_{24} - x_{25} \geq u_2 \geq 1 + 2x_{21} + x_{32} + x_{42} + x_{52},$$
$$4 - 2x_{13} - x_{32} - x_{34} - x_{35} \geq u_3 \geq 1 + 2x_{31} + x_{23} + x_{43} + x_{53},$$
$$4 - 2x_{14} - x_{42} - x_{43} - x_{45} \geq u_4 \geq 1 + 2x_{41} + x_{24} + x_{34} + x_{54},$$
$$4 - 2x_{15} - x_{52} - x_{53} - x_{54} \geq u_5 \geq 1 + 2x_{51} + x_{25} + x_{35} + x_{45}.$$

They become

$$3 \geq u_2 \geq 2,$$
$$2 \geq u_3 \geq 1.5,$$
$$3.5 \geq u_4 \geq 3,$$
$$2.5 \geq u_5 \geq 2.5,$$

after replacing the above given x_{ij} values. Then we write inequalities (4.12) for $(i, j) = (2, 4)$ and $(i, j) = (4, 2)$:

$$u_2 - u_4 + 4x_{24} + 2x_{42} \leq 3,$$
$$u_4 - u_2 + 4x_{42} + 2x_{24} \leq 3.$$

Using $x_{24} = 0.5$ and $x_{42} = 0.5$ we have $u_2 = u_4$. Considering the lower bound of u_4 and the upper bound of u_2 we obtain $u_4 = u_2 = 3$. On the other hand inequalities (4.12)

for $(i,j) = (2,3)$ give

$$u_2 - u_3 + 4x_{23} + 2x_{32} \leq 3.$$

For $x_{23} = 0.5$, and $x_{32} = 0.5$ we obtain $u_2 \leq u_3$, which is a contradiction since $3 = u_2 \nleq u_3 \leq 2$. In addition, since P(GP4) is a proper subset of P(GP1), P(GP2) and P(GP3) this (x,v) vector is also included in them, and therefore not only the GP4 but also the GP1, GP2 and GP3 formulations are not stronger than DL formulation. This ends the discussion on the incomparability of the GP1 – GP4 formulations with the DL formulation.

The SD formulation and the GP1 – GP4 formulations are incomparable. Similar incomparability relations exist between the GP1 – GP4 and the SD formulations. Consider first

$$\begin{aligned} x_{13} &= 0.5, \quad x_{15} = 0.5, \\ x_{23} &= 0.5, \quad x_{24} = 0.5, \\ x_{32} &= 0.5, \quad x_{35} = 0.5, \\ x_{41} &= 0.5, \quad x_{42} = 0.5, \\ x_{51} &= 0.5, \quad x_{54} = 0.5, \end{aligned}$$

$$\begin{aligned} v_{23} &= 0.5, \quad v_{24} = 0.5, \quad v_{25} = 0.5, \\ v_{32} &= 0.5, \quad v_{34} = 0.5, \quad v_{35} = 0.5, \\ v_{42} &= 0.5, \quad v_{43} = 0.5, \quad v_{45} = 0.5, \\ v_{52} &= 0.5, \quad v_{53} = 0.5, \quad v_{54} = 0.5, \end{aligned}$$

and all other variables are equal to zero. This is a feasible solution of P(GP4). To show that no u_i can exist such that (x,u) is included in P(SD) let us write inequalities

(4.18) for $j = 2, 3, 4$

$$1 + (1 - x_{12}) + 2x_{21} \leq u_2 \leq 4 - 2x_{12} - (1 - x_{21})$$
$$1 + (1 - x_{13}) + 2x_{31} \leq u_3 \leq 4 - 2x_{13} - (1 - x_{31})$$
$$1 + (1 - x_{14}) + 2x_{41} \leq u_4 \leq 4 - 2x_{14} - (1 - x_{41}).$$

Then by using $x_{12} = 0$, $x_{13} = 0.5$, $x_{14} = 0$, $x_{15} = 0.5$, $x_{21} = 0$, $x_{31} = 0$, $x_{41} = 0.5$ and $x_{51} = 0.5$ we obtain

$$2 \leq u_2 \leq 3,$$
$$1.5 \leq u_3 \leq 2,$$
$$3 \leq u_4 \leq 3.5.$$

On the other hand, when we write (4.17) for $(i, j) = (3, 2)$, $(i, j) = (2, 3)$, $(i, j) = (4, 2)$ and $(i, j) = (2, 4)$,

$$u_3 + 3x_{23} - 4(1 - x_{32}) \leq y_{23} + y_{32} \leq u_3 - (1 - x_{32})$$
$$u_2 + 3x_{32} - 4(1 - x_{23}) \leq y_{32} + y_{23} \leq u_2 - (1 - x_{23})$$
$$u_2 + 3x_{42} - 4(1 - x_{24}) \leq y_{42} + y_{24} \leq u_2 - (1 - x_{24})$$
$$u_4 + 3x_{24} - 4(1 - x_{42}) \leq y_{24} + y_{42} \leq u_4 - (1 - x_{42})$$

and set $x_{23} = x_{24} = x_{32} = x_{42} = 0.5$ we obtain

$$u_3 - 0.5 \leq y_{23} + y_{32} \leq u_3 - 0.5$$
$$u_2 - 0.5 \leq y_{32} + y_{23} \leq u_2 - 0.5$$
$$u_2 - 0.5 \leq y_{42} + y_{24} \leq u_2 - 0.5$$
$$u_4 - 0.5 \leq y_{24} + y_{42} \leq u_4 - 0.5,$$

which imply $y_{23} + y_{32} + 0.5 = u_3 = u_2$ and $y_{24} + y_{42} + 0.5 = u_4 = u_2$. Because of the bounds for u_2 and u_3, $u_4 = u_2 = u_3 = 2$, which results in the contradiction

$3 \leq u_4 \leq 3.5$. Hence for the given x_{ij} values no u_i can exist such that (x, u) is in P(SD). Since P(GP4) is a subset of P(GP1) – P(GP3), this solution is also feasible for P(GP1) – P(GP3). Therefore the GP1 – GP4 formulations are not stronger than the SD formulation.

To show that the SD formulation is not stronger than the GP1 formulation, consider

$$
\begin{aligned}
&x_{14} = 0.25, \quad x_{15} = 0.75, \\
&x_{26} = 1, \\
&x_{37} = 1, \\
&x_{43} = 1, \\
&x_{52} = 1, \\
&x_{64} = 0.75, \quad x_{65} = 0.25, \\
&x_{71} = 1,
\end{aligned}
$$

$$
\begin{aligned}
&y_{26} = 2.25, \\
&y_{37} = 5, \\
&y_{43} = 4, \\
&y_{52} = 1.25, \\
&y_{64} = 3, \qquad y_{65} = 0.25,
\end{aligned}
$$

$$u_2 = 2.25, \quad u_3 = 5, \quad u_4 = 4, \quad u_5 = 1.25, \quad u_6 = 3.25, \quad u_7 = 6,$$

and all other variables are equal to zero. It is included in P(SD). Then circuit inequalities for the circuit $\{(5,2),(2,6),(6,5)\}$ yield the contradiction $2.25 \nleq 2$. Consequently since TP(GP1) is equivalent to the circuit polytope P(CP), and these x_{ij} values are not contained in P(CP) they can not be in TP(GP1) either. In other words there can not exist v_{ij} values so that the corresponding (x, v) vector is in P(GP1). On the other hand, there can not exist either v_{ij} values for these x_{ij} values such that the correspond-

ing (x, v) vector is in P(GP2) – P(GP4), since they are subsets of P(GP1). As a result, the SD formulation is not stronger than the GP1 – GP4 formulations. Therefore, the GP1 – GP4 formulations and the SD formulation are incomparable.

4.2.9.2. Generalization Based on Claus' Multi–Commodity Flow Formulation. In their latest work Gouveia and Pires consider an equivalent version of CLAUS formulation, which consists of the assignment constraints (4.2), (4.3), integrality restrictions (4.5), and the following subtour elimination constraints [41]:

$$\sum_{j=1,j\neq i}^{n} z_{kij} - \sum_{j=1,j\neq i}^{n} z_{kji} = 0 \quad i,k = 2,\ldots,n;\ i \neq k \tag{4.55}$$

$$\sum_{i=1}^{n} z_{jji} = 1 \qquad j = 2,\ldots,n \tag{4.56}$$

$$z_{kij} \leq x_{ij} \qquad i,j = 1,\ldots,n;\ k = 2,\ldots,n;\ i \neq j \tag{4.57}$$

$$z_{kij} \geq 0 \qquad i,j = 1,\ldots,n;\ k = 2,\ldots,n;\ i \neq j \tag{4.58}$$

Although both $O(n^3)$-formulations are equivalent and give the same LP bound, the order of indexing of commodity flow variables is different. They use z_{kij} instead of z_{ijk}, which only affects their interpretation. Namely, $z_{kij} = 1$ indicates that arc (i, j) is on the path from vertex $k = 2,\ldots,n$ to vertex 1.

Gouveia and Pires have shown that it is possible to obtain subtour elimination constraints (4.48), (4.49), and (4.52) of their GP3 formulation from constraints (4.57), (4.58), and the following weaker version of (4.55)

$$\sum_{j=1,j\neq i}^{n} z_{kij} \geq z_{kqi} + z_{kki} \quad i,k,q = 2,\ldots,n;\ k = 2,\ldots,n;\ i \neq k;\ q \neq k \tag{4.59}$$

by using the linear transformation

$$\sum_{j=1}^{n} z_{kij} = v_{ki} \quad i,k = 2,\ldots,n. \tag{4.60}$$

It is also interesting to observe that the left hand side of (4.60) counts the number of outarcs of vertex i on the path from vertex k to vertex 1. Besides, $v_{ki} = 1$ if k is on the path from vertex 1 to vertex i and $v_{ki} = 0$ otherwise.

Inequalities (4.59) are weaker than inequalities (4.55) since they are satisfied by any flow vector satisfying (4.55); but the converse is not always true. At the end of the derivation of constraints (4.52) from (4.57)–(4.59) and (4.60); Gouveia and Pires reach the conclusion that they have generated the GP3 formulation from the weaker constraints (4.59) suggesting that they may obtain stronger constraints if they add additional terms to the right-hand side of (4.59) and/or consider at the same time several constraints (4.59) such as

$$\sum_{j=1, j\neq i}^{n} z_{kij} \geq z_{kqi} + z_{kmi} + z_{kki}$$

$$i, k, m, q = 2, \ldots, n;\ k = 2, \ldots, n;\ q \neq k;\ m \neq k;\ i \neq k \quad (4.61)$$

and

$$\sum_{j=1}^{n} z_{kmj} \geq z_{kqm} + z_{kkm} \qquad m, q, k = 2, \ldots, n;\ m \neq k;\ q \neq k. \quad (4.62)$$

A proper use of these two inequalities with the transformation (4.60), (4.2) and (4.57) gives the two-path inequalities

$$x_{im} + x_{mj} + x_{ij} + x_{ik} + x_{km} + x_{mk} + x_{kj} + v_{ki} - v_{kj} \leq 2$$

$$i, j, k, m = 2, \ldots, n;\ k \neq i, j, m. \quad (4.63)$$

(4.63) can also be obtained from constraints (4.52) by summing those which represent triplets (i, m, k) and (m, j, k) side by side and lifting x_{ij} with a coefficient 1 in the aggregated constraint gives the two-path inequality for (i, m, j, k) [41]. When same arguments are applied to (4.55)–(4.58) the two-path inequalities (4.63) are further generalized to (4.64), where the central vertex represented with the index m is replaced

with clique $S \subseteq V$:

$$v_{ki} + \sum_{m \in S} (x_{im} + x_{km}) + \sum_{m \in S} (x_{mj} + x_{mk}) + \sum_{p,q \in S} x_{pq} + x_{ik} + x_{kj} + x_{ij} \leq 1 + |S| + v_{kj}$$
$$i, j, k = 2, \ldots, n;\ i, j \neq k;\ S \subset V \diagup \{1, i, j, k\}\,. \qquad (4.64)$$

Notice that for $|S| = 0$ and $|S| = 1$ we are respectively back to constraints (4.52) and (4.63).

Constraints (4.64) can also be derived from the subtour elimination constraints of CLAUS formulation and thus they are valid subtour elimination constraints [41]. Consequently, GP5 formulation is obtained by replacing inequalities (4.52) of the GP3 with inequalities (4.64) [41]. Then,

$$P(GP5) = \left\{(x, v) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2) - (4.4),\ (4.48),\ (4.49),\ (4.64)\right\}.$$

The derivation aggregates the following two weaker versions of constraints (4.55), namely

$$\sum_{j=1}^{n} z_{kij} \geq \sum_{m \in S} z_{kmi} + z_{kqi} + z_{kki}$$
$$i, k, q = 2, \ldots, n;\ i, q \neq k;\ S \subset V \diagup \{1, i, k, q\} \qquad (4.65)$$
$$\sum_{m \in S} \sum_{j=1}^{n} z_{kmj} \geq \sum_{m \in S} \left(\sum_{n \in S} z_{knm} + z_{kqm} + z_{kkm} \right)$$
$$k, q = 2, \ldots, n;\ q \neq k;\ S \subset V \diagup \{1, i, k, q\}\,, \qquad (4.66)$$

and applies transformation (4.60). In short, P(CLAUS) is included in P(GP5), which is, in its turn, properly included in P(GP3). It means CLAUS formulation is not weaker than the GP5 formulation, which is stronger than the GP3 formulation. The last statement follows from the fact that constraints (4.64) include constraints (4.52) as a special case for $|S| = 0$.

The addition of the two of inequalities (4.51), one for the triplet (i, m, k) and the other for the triplet (m, j, k) and the lifting of x_{ij} and x_{ji} with coefficients 1 results in

$$x_{im} + x_{mi} + x_{mj} + x_{jm} + x_{ij} + x_{ji} + v_{ki} - v_{kj} \leq 2$$
$$i, j, k, m = 2, \ldots, n;\ k \neq i, j, m. \qquad (4.67)$$

It can be noticed that at most two of the x_{pq} variables can be equal to one. In addition the arcs corresponding to these two variables form a path with two arcs, and $v_{ki} = 1$ if and only if $v_{kj} = 1$ because vertex k is not included in the path connecting vertices i and j. The fact that x_{pq} variables represent the arcs of the clique with $S = \{i, j, m\}$ suggest the generalization of constraints (4.67) for larger cliques [41]:

$$\sum_{\substack{p,q \in S \\ p \neq q}} x_{pq} + v_{ki} - v_{kj} \leq |S| - 1 \quad i, j, k = 2, \ldots, n,$$
$$S \subseteq \{2, \ldots, n\},\ |S| \geq 2;\ k \notin S;\ i, j \in S. \qquad (4.68)$$

When the value of the summation is less than $|S| - 1$, constraints (4.68) are trivially satisfied. If it is equal to $|S| - 1$ then the arcs associated with the variables which are equal to 1 form a path with $|S| - 1$ arcs, and $v_{ki} = 1$ if and only if $v_{kj} = 1$, because although vertices i and j are in S, vertex k is not. Hence inequalities (4.68) are valid. Besides, inequalities (4.51) and (4.67) are special cases obtained respectively for $|S| = 2$ and $|S| = 3$. Then Gouveia and Pires propose the GP6 formulation by replacing inequalities (4.51) of the GP2 formulation with inequalities (4.68) [41]. This formulation clearly has $O(2^n)$ constraints, and

$$P(GP6) = \left\{(x, v) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2) - (4.4),\ (4.48),\ (4.49),\ (4.68)\right\}.$$

Consider now two of the constraints (4.68); one for a given subset S and indices i, j and k, and the other for the same S but for indices j, i and k. Their addition gives clique inequalities (4.7) for subset S. This shows that it is possible to obtain clique inequalities by properly aggregating constraints (4.68) and hence the projection TP(GP6) of the

polyhedron P(GP6) into the space of x_{ij} variables is a subset of the polyhedron P(DFJ). Then the GP6 formulation can not be weaker than the DFJ formulation. On the other hand the authors conjecture that the converse of this statement is also true, namely both formulations are in fact equivalent, and P(GP6) is an extended characterization of P(DFJ). Observe that the polyhedron P(GP6) can not be obtained by transforming the polyhedron P(CLAUS) into the subspace of (x_{ij}, v_{ki}) variables under the linear transformation (4.60). This is, first of all, because it includes a special subset of constraints (4.68) (i.e. for $|S| = \{i, j\}$) which are constraints (4.51) belonging to the GP2 formulation. Moreover, the GP2 and GP3 formulations are shown to be incomparable.

As we have seen, the constraint set of the last two formulations by Gouveia and Pires [41] include the constraints of CLAUS formulation as a subset and they are stronger as a result. The derivation of inequalities (4.52) from constraints (4.55), (4.57) and (4.58) under linear transformation (4.60) shows that the same transformation cannot be used to generate inequalities (4.51) from constraints (4.55), (4.57) and (4.58). Therefore, inequalities (4.51) and linear transformation (4.60) are not redundant for the polyhedron P(CLAUS) and their addition to constraints (4.2)–(4.5), and (4.55)–(4.58) of CLAUS formulation results in the stronger GP7 formulation having $O(n^3)$ constraints. In other words

$$P(GP7) = \left\{(x, v) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2) - (4.4),\ (4.55) - (4.58),\ (4.51),\ (4.60)\right\}.$$

Recall that Padberg and Sung have shown TP(CLAUS), the projection of the polyhedron P(CLAUS) into the subspace of x_{ij} variables, is the polyhedron P(DFJ) [15]. As a consequence TP(GP7), the projection of the polyhedron P(GP7) into the subspace of x_{ij} variables, a proper subset of the set described by the clique inequalities (4.7), and therefore TP(GP7) is a proper subset of TP(CLAUS) and P(DFJ).

In their earlier work Gouveia and Pires demonstrate that the GP4 and CLAUS formulations are not comparable [40]. However, this is not the case for the GP7 formulation; it includes the constraints of CLAUS formulation from which inequalities (4.52)

can be derived by using linear transformation (4.60). It also includes constraints (4.51). As a consequence the projection of P(GP7) into the subspace of (x_{ij}, v_{ki}) variables is a proper subset of P(GP4), which implies that the GP7 formulation is stronger than the GP4 formulation.

The addition of constraints (4.51) for given indices j, i and k, to (4.64) for $\{i,j,k\}$ and for a subset S result in the lifted circuit inequalities

$$\sum_{m\in S}(x_{im}+x_{km})+\sum_{m\in S}(x_{mj}+x_{mk})+\sum_{p,q\in S}x_{pq}+x_{ik}+x_{kj}+2x_{ij}+x_{ji}\leq 2+|S|$$
$$i,j,k=2,\ldots,n;\ i,j\neq k;\ S\subset V\diagup\{1,i,j,k\}\,, \qquad (4.69)$$

which are special cases of the facet defining FD inequalities introduced by Balas and Fischetti [56]. It is possible to derive constraints (4.64) from a weaker version of (4.55)–(4.57) using (4.60). Besides, constraints (4.51) together with constraints (4.64) imply constraints (4.69) [41]. As a result, the GP7 formulation is a compact model which subsumes the exponential sized set of simple FD inequalities.

Recall that constraints (4.51) can be obtained from constraints (4.68) for $|S| = \{i,j\}$. Therefore it is possible to obtain a stronger formulation by adding constraints (4.68) and transformation (4.60) to CLAUS formulation. This gives the GP8 formulation, which is the eighth formulation due to Gouveia and Pires [41], where

$$P(GP8)=\left\{(x,v)\in\mathbb{R}^{|A|}\times\mathbb{R}^{|A|}:(4.2)-(4.4),\ (4.55)-(4.58),\ (4.68),\ (4.60)\right\}.$$

The GP8 formulation is stronger than both of their sixth and seventh formulations, the GP6 and GP7 formulations. The GP5, GP6 and GP8 formulations have $O(2^n)$ subtour elimination constraints while this number is $O(n^3)$ for the GP7 formulation.

4.2.10. Fox, Gavish and Graves's Formulation

Consider the following one-machine n-jobs scheduling problem, which is known as the time dependent TSP (TDTSP) in the literature. A set of $n-1$ jobs, denoted by $2, ..., n$, are to be performed in a single machine and setup cost c_{ijt} occurs when job j is processed in the t^{th} order immediately after job i. The machine is in the initial state which we denote by job 1. We assume that the machine will be in that state after processing all of the $n-1$ jobs. The problem is to find the cheapest sequence of performing all jobs.

The TDTSP is a generalization of the standard TSP where the cost of any given arc depends on its position in the tour. The TDTSP has several real-world applications. According to Pickard and Queyranne [57], the TDTSP is originally defined by Fox [58] and is illustrated by several examples from the brewing industry.

There are three time-dependent formulations proposed by Fox, Gavish and Graves [59]. The one we present below has four constraints and will be denoted as FGG4 in the sequel.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} c_{ijt}y_{ijt} \tag{4.70}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{t=1}^{n} y_{ijt} = 1 \qquad i = 1, \ldots, n \tag{4.71}$$

$$\sum_{i=1}^{n}\sum_{t=1}^{n} y_{ijt} = 1 \qquad j = 1, \ldots, n \tag{4.72}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} y_{ijt} = 1 \qquad t = 1, \ldots, n \tag{4.73}$$

$$\sum_{j=1}^{n}\sum_{t=1}^{n} t y_{ijt} - \sum_{j=1}^{n}\sum_{t=1}^{n-1} y_{jit} = 1 \qquad i = 2, \ldots, n \tag{4.74}$$

$$y_{ijt} \in \{0, 1\} \qquad i, j, t = 1, \ldots, n \tag{4.75}$$

Variable y_{ijt} is equal to 1 is vertex j is in the position number t immediately after vertex i, 0 otherwise. Constraints (4.71) - (4.73) are the assignment constraints. (4.71) and (4.72) ensure respectively that there exist exactly one leaving arc from vertex i and there exist exactly one incoming arc to vertex j. (4.73) state that there must be exactly one vertex in order t. Constraints (4.74) ensure that for each vertex i other than vertex 1, the position number of an arc leaving vertex i is exactly one more than the position number of an arc entering that vertex. Fox, Gavish and Graves [59] have noted that constraints (4.73) are not needed in showing the validity of the FGG4 and, they may be dropped from the FGG4. We will call, the formulation consisting of objective function (4.70) and, constraints (4.71), (4.72), (4.74) and (4.75) as the FGG3 formulation. Gouveia and Voβ [38] have shown that the FGG4 is stronger than the FGG3 by presenting an example which is feasible for the LP-relaxation of the FGG3 but infeasible for the LP-relaxation of the FGG4 formulation. In other words, P(FGG4) is a proper subset of P(FGG3).

Furthermore, they have also proposed a more compact version of FGG4 formulation, which will be denoted as the FGG2 formulation, by using

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} y_{ijt} = n \tag{4.76}$$

as an aggregation of constraints (4.71)-(4.73). For the validity of the FGG2 formulation, Fox, Gavish and Graves [59] assume that $y_{ij1} = y_{jin}$ for i=2,...,n, $y_{1it} = 0$ for t=2,...,n and $y_{i1t} = 0$ for t=1,...,n-1.

Now, consider the following example which is feasible for the LP relaxation of the FGG2 formulation but infeasible for the LP relaxation of the FGG3 formulation because of constraints (4.71)

$$y_{131} = 1,\ y_{214} = 1,\ y_{243} = 1/3,\ y_{322} = 1,\ y_{423} = 2/3 \tag{4.77}$$

Since the FGG2 formulation has aggregated assignment constraint (4.76) instead of

(4.71) - (4.73) and therefore every solution feasible for the LP relaxation of the FGG3 formulation is also feasible for the LP relaxation of the FGG2 formulation, this particular example shows that P(FGG3) is a proper subset of P(FGG2).

4.2.11. Pickard and Queyranne's Formulation

Pickard and Queyranne [57] have proposed the following three-indexed Time-Dependent Travelling Salesman Problem formulation:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} c_{ijt}y_{ijt} \tag{4.78}$$

$$\text{s.t.} \sum_{i=1}^{n} y_{1i1} = 1 \tag{4.79}$$

$$\sum_{j=1}^{n}\sum_{t=1}^{n} y_{jit} = 1 \qquad i = 1,\ldots,n \tag{4.80}$$

$$\sum_{j=1}^{n} y_{jit} = \sum_{j=1}^{n} y_{ij,t+1} \qquad i = 2,\ldots,n;\ t = 1,\ldots,n-1 \tag{4.81}$$

$$y_{ijt} \in \{0,1\} \qquad i,j,t = 1,\ldots,n \tag{4.82}$$

where $y_{ijt} = 1$ when city j is visited after city i in the t^{th} order. Constraint (4.79) state that city 1 must be left exactly once in the first order. Constraints (4.80) are the assignment constraints and constraints (4.81) denote the visit order of vertices.

Pickard and Queyranne [57] have discussed the application of this three-indexed formulation into the tardiness problem in one-machine scheduling problem. The visit order of the vertices for the TSP is now the processing order of the jobs in the machine. In the scheduling problem, setup cost is associated with each of n jobs to be processed. Moreover, the setup cost of each job depends not only on the job that precedes it, but also on its position (time) in the sequence.

4.2.12. Finke, Claus and Gunn's Formulation

Finke *et al.* [60] have proposed the first two–commodity flow formulation. In their formulation, the salesman leaves vertex 1 with $n-1$ units of one type of commodity, say commodity Y, and none of the other type, say commodity Z. At each vertex, he drops one unit of commodity Y and collects one unit of commodity Z. In other words he travels at all times with a total of $n-1$ units of commodities. Their formulation is as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\frac{(y_{ij}+z_{ij})}{(n-1)} \tag{4.83}$$

$$\text{s.t.} \sum_{j=2}^{n} y_{1j} - \sum_{j=2}^{n} y_{j1} = (n-1) \tag{4.84}$$

$$\sum_{j=2}^{n} z_{1j} - \sum_{j=2}^{n} z_{j1} = -(n-1) \tag{4.85}$$

$$\sum_{\substack{j=1\\ i\neq j}}^{n} y_{ij} - \sum_{\substack{j=1\\ i\neq j}}^{n} y_{ji} = -1 \qquad i=2,\ldots,n \tag{4.86}$$

$$\sum_{\substack{j=1\\ i\neq j}}^{n} z_{ij} - \sum_{\substack{j=1\\ i\neq j}}^{n} z_{ji} = -1 \qquad i=2,\ldots,n \tag{4.87}$$

$$\sum_{\substack{j=1\\ i\neq j}}^{n} y_{ij} + \sum_{\substack{j=1\\ i\neq j}}^{n} z_{ij} = n-1 \qquad i=1,\ldots,n \tag{4.88}$$

$$y_{ij}+z_{ij} \in \{0, n-1\} \qquad i,j=1,\ldots,n;\ i\neq j \tag{4.89}$$

$$y_{ij} \geq 0, z_{ij} \geq 0 \qquad i,j=1,\ldots,n;\ i\neq j. \tag{4.90}$$

Constraints (4.84)- (4.87) conserve the flow for each commodity at every vertex. Constraints (4.88) ensure the existence of $(n-1)$ commodities outgoing from each vertex.

Finally, we close this chapter with Figure 4.1, which illustrates the relative strength of the LP relaxations of the twenty two existing ATSP formulations which we have briefly explained so far. An arrow from A to B indicates that the LP relaxation of the B formulation is tighter than the LP relaxation of the A formulation, which

equivalently means the B formulation is stronger than the A formulation. Bidirectional arrows represent the equivalence of the A and B formulations. Dashed lines imply the incomparability of the A and B formulations.

The relations between the FGG3 and FCG, the FGG3 and FGG4 and, the FGG4 and PQ are shown by Gouveia and Voss [38]. The relation between the FGG2 and DFJ formulations is shown by Padberg and Sung [15]. The equivalence of the DFJ and WONG formulations and, the relations between the GG and DFJ and the MTZ and GG are shown by Wong [37]. The relations between the FCG and GG, the LANGEVIN and LOULOU, the LOULOU and DFJ, the WONG and CLAUS formulations are presented by Langevin *et al.* [14]. The relations between the GP1 and MTZ, the GP1 and GP2, the GP1 and GP3, the GP3 and GP2, the GP3 and GP4, the GP2 and GP4 formulations are shown by Gouveia and Pires [40]. Moreover, Gouveia and Pires have shown the relations between the GP3 and GP5, the GP2 and GP6, the GP4 and GP7, the GP6 and CLAUS, the GP5 and CLAUS, the CLAUS and GP7, the GP6 and GP8, the GP7 and GP8 formulations [41]. The relations between the FGG2 and FGG3, the GG and SD, the GG and DL, the SD and GP1, the SD and GP4, the DL and GP1, and the DL and GP4 formulations are our contributions.

One relation which we have not shown rigorously is between the SD and DFJ formulations (or CLAUS formulation equivalently). Clearly, the SD formulation is not stronger than the DFJ formulation. However, they may be incomparable, which indeed requires specific examples. Then Figure 4.1 will become complete.

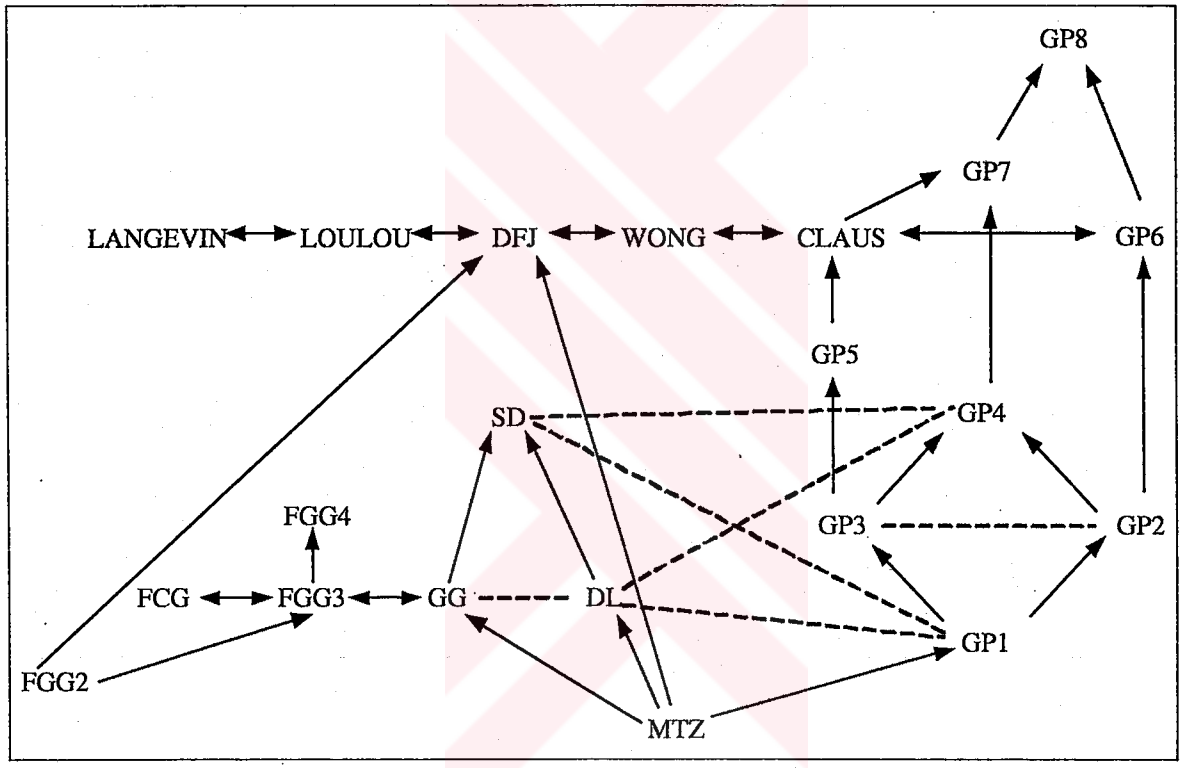

Figure 4.1. Relative strength of the twenty two existing ATSP formulations

5. NEW EXTENDED ASYMMETRIC TRAVELLING SALESMAN PROBLEM FORMULATIONS

5.1. Balanced Unidirectional Cyclic Layout Problem and a New Formulation for the Asymmetric Travelling Salesman Problem

In this chapter we will propose a new ATSP formulation based on the formulation of the BUCLP given by (2.15)–(2.21). Recall that binary decision variables are defined as

$$x_{ij} = \begin{cases} 1 & \text{if workstation } j \text{ immediately follows} \\ & \text{workstation } i \text{ in an optimal sequence} \\ 0 & \text{otherwise} \end{cases},$$

and d_{ij} denote the distance between the workstations i and j in the original formulation. For the sake of clearness we will represent once more their formulation:

$$\text{BUCLP:} \quad \min \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} f_{ij} d_{ij} \tag{5.1}$$

$$\text{s.t.} \sum_{j=1, j\neq i}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \tag{5.2}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{5.3}$$

$$d_{ij} + d_{ji} = 1 \qquad i, j = 1, \ldots, n; i \neq j \tag{5.4}$$

$$d_{ij} \geq d_{ik} + d_{kj} + x_{kj} - 1 \qquad i, j, k = 1, \ldots, n; i \neq j \neq k \tag{5.5}$$

$$0 \leq d_{ij} \qquad i, j = 1, \ldots, n; i \neq j \tag{5.6}$$

$$x_{ij} = 0, 1 \qquad i, j = 1, \ldots, n; i \neq j \tag{5.7}$$

In this formulation constraints (5.2) and (5.3) are exactly the assignment constraints (4.2) and (4.3) and ensure that any workstation has exactly one predecessor

and one successor. Constraints (5.4) and (5.5) define the properties of the distance matrix. (5.5) guarantees also $d_{ij} = d_{ik} + d_{kj}$ if j immediately follows k in the flow direction.

The new ATSP formulation can then be obtained after replacing objective function (5.1) with (4.1) and treating the workstations as cities the travelling salesman has to visit. The decision variables x_{ij} have the same meaning. Besides, circular distances d_{ij} become the additional variables which makes nonnegativity restrictions (5.6) redundant. Notice that, constraints (5.4) and (5.5) are the new subtour elimination constraints and the ones for vertex 1 do not have to be considered as before. Consequently, we drop vertex 1 from the sets (5.4) and (5.5), and refer the new ones as (5.4') and (5.5'). In short, the new formulation, which will be referred as NEW1 in the sequel, has objective (4.1) and constraints (5.2)–(5.7), (5.4') and (5.5').

5.1.1. The Validity of the New Formulation

In Section 2.2.2 we have stated important properties of the BUCLP formulation. In this section we will use some of the theoretical results derived in Section 2.2.2. Notice that, the ATSP's complete digraph is exactly Afentakis' traffic graph. The set of workstations can be considered as the set of cities, and the set of directed arcs (i, j) reflecting the part flow from workstation i to workstation j, can now be regarded as the arcs of the ATSP's complete digraph. The flow costs associated with each arc (i, j) of the traffic graph is now the distances c_{ij} from city i to city j.

As a result, we will directly use the consequences of Theorem 2.1 within the context of the ATSP. There are two direct consequences of Theorem 2.1. The first one, given as Corollary 2.1, implies now that NEW1 is a valid ATSP formulation. Corollary 2.2 points to an interesting feature of the new formulation: It remains feasible even vertex 1 is also included in the subtour elimination constraints. Namely, (5.4) and (5.5) eliminate any subtour without causing any violation for any Hamiltonian circuit, which is not the case for many extended ATSP formulations such as the MTZ formulation and its extensions the DL, SD and GP formulations. This is the second consequence.

5.1.2. The Strength of the New Formulation

Now we have enough material for discussing rigorously the strength of the NEW1 formulation. Corollary 5.1, the succeeding discussion, and Proposition 5.1 are also consequences of Theorem 2.1. We denote the feasible solution set of NEW1 formulation and its projection into the subspace of x_{ij} variables as P(NEW1) and TP(NEW1) respectively by using the notation we have introduced in Section 1.

Corollary 5.1 TP(NEW1) is a proper subset of TP(MTZ) for $n \geq 3$.

Proof. Let us add up side by side constraints (5.5') of the polyhedron P(NEW1) as it is done in the proof of Corollary 2.1. Then we obtain surrogate inequality

$$\sum_{(i,j)\in C} x_{ij} \leq |C| - \sum_{(i,j)\in C} d_{ij} \tag{5.8}$$

$\sum_{(i,j)\in C} d_{ij} = 1$ as a consequence of Theorem 2.1 and (5.8) becomes equivalent to a circuit inequality, which is tighter than its weaker version belonging to (4.11), since $(|C|-1) < |C| - \frac{|C|}{n-1}$. Hence for any $x \in$ P(CP) there exists $d \in \mathbb{R}^n$ such that $(x,d) \in$ P(NEW1), which means TP(NEW1) is a subset of P(CP). This completes the proof since P(CP) has been shown to be a proper subset of TP(MTZ) [15]. ∎

As a consequence of this corollary we can say that the NEW1 formulation has tighter LP relaxation than the one of the MTZ formulation, and can give larger LP bound. In addition, it is at least as strong as the GP1 formulation, Gouveia and Pires' first extension of the MTZ formulation explained in the previous section [40], since TP(NEW1) is a subset of P(CP), which is shown to be equivalent to TP(GP1) [40].

In fact the NEW1 formulation is stronger than the GP1 formulation. This can be shown by finding a feasible solution $\overline{x}$ of P(CP) for which no $\overline{d} \in \mathbb{R}^{|E|}$ can exist such that $(\overline{x}, \overline{d})$ is a feasible solution in P(NEW1). This shows that TP(NEW1) is in fact a proper subset of P(CP) and TP(GP1). Let us assume that $n \geq 4$, and consider $\overline{x}$ represented in the support graph given in Figure 5.1. This corresponds to

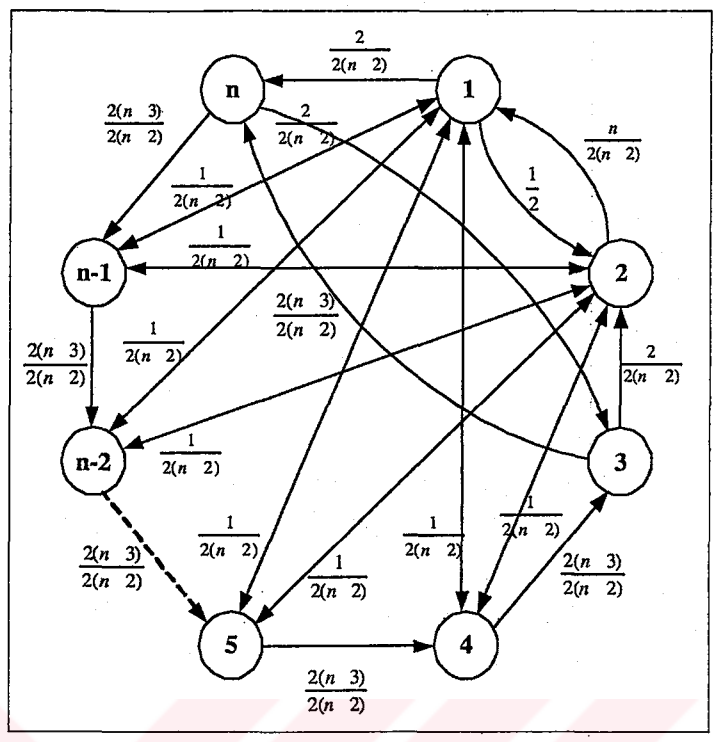

Figure 5.1. Example showing that TP(NEW1) is a proper subset of P(CP)

$$\overline{x}_{1,j} = \frac{1}{2(n-2)} \quad j = 4,5,...,n-1;$$
$$\overline{x}_{1,2} = \frac{1}{2};$$
$$\overline{x}_{1,n} = \frac{2}{2(n-2)};$$
$$\overline{x}_{2,1} = \frac{n}{2(n-2)};$$
$$\overline{x}_{2,j} = \frac{1}{2(n-2)} \quad j = 4,5,...,n-1;$$
$$\overline{x}_{3,2} = \frac{2}{2(n-2)};$$
$$\overline{x}_{j,1} = \frac{1}{2(n-2)} \quad j = 4,5,...,n-1;$$
$$\overline{x}_{j,2} = \frac{1}{2(n-2)} \quad j = 4,5,...,n-1;$$
$$\overline{x}_{j,j-1} = \frac{2(n-3)}{2(n-2)} \quad j = 4,5,...,n$$
$$\overline{x}_{n,3} = \frac{2}{2(n-2)};$$
$$\overline{x}_{3,n} = \frac{2(n-3)}{2(n-2)};$$

As it can be observed $\overline{x}$ satisfies the assignment constraints. The next step is to check the feasibility of $\overline{x}$ with respect to circuit inequalities (4.7). Recall that subtour elimination constraints for home city, namely city 1, are redundant and therefore disregarded in the previous work on the ATSP formulations as well as in this one. In fact the MTZ formulation and its extensions due to Gouveia and Pires [40, 41], Desrochers and Laporte [44], and Sherali and Driscoll [48] become infeasible when the subtour elimination constraints for city 1 are also considered. This is not true for the NEW1

formulation. However, as we have already mentioned we have dropped inequalities for vertex 1 from the sets (5.4) and (5.5) and use (5.4') and (5.5') instead. As a result we check the feasibility of $\overline{x}$ for circuits over vertices $\{2,\ldots,n\}$. All circuits with two arcs have vertex sequence $\{2,i,2\}$ $i=4,\ldots,n-1$ and $\{3,n,3\}$. For the first group

$$\overline{x}_{2i}+\overline{x}_{i2}=\frac{1}{2(n-2)}+\frac{1}{2(n-2)}=\frac{1}{n-2},$$

which is not larger than $|C|-1=1$ for $n\geq 3$. For the circuit $\{3,n,3\}$ similar result can be obtained. Namely,

$$\overline{x}_{3n}+\overline{x}_{n3}=\frac{2}{2(n-2)}+\frac{2(n-3)}{2(n-2)}=1.$$

The largest value of the summation $\sum_{(i,j)\in C}\overline{x}_{ij}$ with $3\leq|C|\leq n-2$, except the circuit with the vertex sequence $\{n,n-1,\ldots,4,3,n\}$ is

$$\frac{2}{2(n-2)}+\frac{1}{2(n-2)}+(|C|-2)\frac{2(n-3)}{2(n-2)}=(|C|-2)-\frac{2\,|C|-7}{2(n-2)},$$

which is less than $|C|-1$. In other words $\overline{x}$ with the support digraph of Figure 5.1 is included in TP(GP1).

Let us consider inequalities (5.5') of the NEW1 formulation for the circuit with vertex sequence $\{n,n-1,\ldots,4,3,n\}$ and the arc $(n,3)$. They are

$$\begin{aligned}
d_{2,n} &\geq d_{2,3}+d_{3,n}+\overline{x}_{3,n}-1\\
d_{2,n-1} &\geq d_{2,n}+d_{n,n-1}+\overline{x}_{n,n-1}-1\\
d_{2,n-2} &\geq d_{2,n-1}+d_{n-1,n-2}+\overline{x}_{n-1,n-2}-1\\
&\vdots\\
d_{2,3} &\geq d_{2,4}+d_{4,3}+\overline{x}_{4,3}-1\\
d_{4,3} &\geq d_{4,n}+d_{n,3}+\overline{x}_{n,3}-1.
\end{aligned}$$

When we add them up side by side we obtain

$$n-1 \geq (d_{3,n}+d_{n,3})+(d_{n,n-1}+\ldots+d_{4,n})+(\overline{x}_{3,n}+\overline{x}_{n,n-1}+\ldots+\overline{x}_{4,3}+\overline{x}_{n,3}).$$

As a consequence of equalities (5.4') $d_{3,n}+d_{n,3}=1$. From Theorem 2.1, $d_{n,n-1}+\ldots+d_{4,n}=1$ follows. Then we obtain

$$n-1 \geq 1+1+(n-2)\frac{2(n-3)}{2(n-2)}+\frac{2}{2(n-2)}=n-1+\frac{1}{n-2},$$

which is a contradiction. Hence for such $\overline{x}$ no $\overline{d}$ such that $(\overline{x},\overline{d})$ is included in P(NEW1) can exist. This clearly means that $\overline{x}$ is not included in the projected polyhedron TP(NEW1). However, it is in P(CP), which is equivalent to TP(GP1). Therefore TP(NEW1) is a proper subset of TP(GP1) and therefore the NEW1 formulation is stronger than the GP1 formulation.

Recall that Gouveia and Pires have strengthened their first extension of the MTZ formulation, namely the GP1 formulation, by replacing constraints (4.47) with (4.51) to obtain the GP2 formulation with constraint sets (4.2)–(4.4), (4.48), (4.49) and (4.51). They have also shown that the projection of the polyhedron P(GP2) into the subspace of x_{ij} variables is TP(GP2) given in Section 4.2.9. The following proposition shows that the NEW1 formulation is at least as good as the GP2 formulation.

Proposition 5.1 TP(NEW1) is a subset of TP(GP2)

Proof. Consider a circuit $D_C=(V_C,C)$. Then by adding the subtour elimination constraints

$$x_{ij}+d_{ki}+d_{ij} \leq d_{kj}+1,\ (i,j)\in C \text{ with } i,j,k=2,\ldots,n; i\neq j\neq k; k\notin V_C$$

and

$$x_{ji}+d_{pj}+d_{ji} \leq d_{pi}+1,\ (i,j)\in C \text{ with } i,j,p=2,\ldots,n; i\neq j\neq p;\ p\in V_C,$$

side by side and using circularity constraints

$$d_{ij} + d_{ji} = 1 \quad i \neq j;\ i,j = 2,\ldots,n$$

we obtain inequalities as a consequence of Theorem 2.1

$$\sum_{(i,j)\in C} x_{ij} + \sum_{\substack{(i,j)\in C \\ i,j\neq p}} x_{ji} \leq |C| - 1, \quad p \in V_C, V_C \subseteq \{2,\ldots,n\}, \tag{5.9}$$

they are demonstrated to be the inequalities describing the projection of the subtour elimination constraints of the GP2 formulation into the subspace of x_{ij} variables [40]. Therefore TP(NEW1) is a subset of TP(GP2). ∎

Moreover, as it can be observed from the computational results reported in Table 5.5 of the next section, there are ATSP instances, such as `ftv33`, `ftv35`, `ftv38`, `ftv47`, `ftv55`, `ftv64`, `ftv70`, `ft70` and `ft53`, for which the LP bound the NEW1 formulation gives, is strictly larger than the one the GP2 formulation does. When we combine this fact with Proposition 5.1 we can see that TP(NEW1) is in fact a proper subset of TP(GP2) and therefore the NEW1 formulation is stronger than the GP2 formulation.

5.2. Extensions of the New Formulation and Their Strengths

We start by pointing out that constraints (5.5') can not be lifted in x_{ij} variables. Notice that by adding up side by side two of the inequalities (5.5') which are given respectively for arcs (k,j) and (j,k), namely

$$d_{ij} \geq d_{ik} + d_{kj} + x_{kj} - 1. \tag{5.10}$$

and

$$d_{ik} \geq d_{ij} + d_{jk} + x_{jk} - 1, \tag{5.11}$$

we obtain $d_{ij} + d_{ik} \geq d_{ij} + d_{ik} + d_{kj} + d_{jk} + x_{kj} + x_{jk} - 2$ or equivalently

$$x_{kj} + x_{jk} \leq 1, \tag{5.12}$$

since $d_{kj} + d_{jk} = 1$ as a result of circularity inequalities (5.4'). The last inequality is the facet defining clique inequality (4.7) for $S = \{i, j\}$ [42]. If distance inequalities (5.5') could be lifted then the same argument would have resulted in an inequality which is stronger than a facet defining inequality. Therefore constraints (5.5') can not be lifted.

Let us suppose that additional variables d_{ij} are restricted to be binary. Then if vertex k immediately precedes vertex j in an optimal tour, $x_{kj} = 1$ and $x_{jk} = 0$, and the setting of $d_{ij} = d_{ik} = d_{jk} = 1$ and $d_{kj} = 0$ satisfy (5.10) and (5.11) as equalities. These values do not contradict Lemma 2.1, and give a decent interpretation. By letting d_{jk} indicate whether vertex k is on the path from vertex 1 to vertex j or not, one can observe that if $x_{kj} = 1$ then $d_{jk} = 1$, $x_{jk} = 0$ and $d_{kj} = 0$. Moreover, if $x_{kj} = 1$ then $d_{ik} = 1$ if and only if $d_{ij} = 1$. In other words if vertex k immediately precedes vertex j, then vertex j is on the path from vertex 1 to vertex i if and only if vertex k is on the path from vertex 1 to vertex i. This interpretation shows that additional variable d_{kj} used in our NEW1 formulation has the same meaning as the additional variable v_{jk} used in Gouveia and Pires' GP1 formulation [40]. Then constraints (5.5') ensure the ordering of cities, and constraints (5.4') state that either city i precedes city j or vice versa. We obtain the first extension of the NEW1 formulation, which is referred as the NEW2 formulation in the sequel, by adding the inequalities

$$x_{ij} - d_{ji} \leq 0 \quad i, j = 2, \ldots, n; i \neq j \tag{5.13}$$

to the NEW1 formulation's constraint set. In other words the NEW2 formulation consists of (4.1)–(4.5), (5.4') and (5.5') and (5.13). Then

$$P(NEW2) = \left\{(x, d) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2) - (4.4), (5.4'), (5.5'), (5.13)\right\}.$$

Notice the similarity between inequalities (5.13), and inequalities (4.48) of the GP1

formulation. Also, as it is shown below in Proposition 5.3; Inequalities (5.14) together with subtour elimination constraints (5.4') and (5.5') imply (4.51) with $v_{ki} = d_{ik}$ and $v_{kj} = d_{jk}$. The NEW1 formulation can be extended furthermore by adding the set

$$x_{ij} + x_{ik} + x_{kj} + d_{ik} - d_{jk} \leq 1 \quad i \neq j \neq k;\ i, j, k = 2, \ldots, n \tag{5.14}$$

to its constraint sets in order to obtain the NEW3 formulation (4.1)–(4.5), (5.4') and (5.5'), and (5.14) with the LP relaxation has the feasible solution set

$$P(NEW3) = \left\{(x, d) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2) - (4.4), (5.4'), (5.5'), (5.14)\right\}.$$

Observe that (5.14) becomes (4.52) for $d_{ki} = v_{ik}$ and $d_{kj} = v_{jk}$. Yet another extension is possible. The NEW4 formulation can be obtained just by adding both (5.13) and (5.14) to the NEW1 formulation: The NEW4 formulation consists of (4.1)–(4.5), (5.4') and (5.5'), (5.13) and (5.14). Consequently

$$P(NEW4) = \left\{(x, d) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2) - (4.4), (5.4'), (5.5'), (5.13), (5.14)\right\}.$$

Observe that each member of NEW formulation family has $O(n^3)$ constraints and $O(n^2)$ variables.

The NEW1, NEW2, NEW3 and NEW4 formulations can be ordered easily according to their strengths: The NEW2 formulation is at least as strong as the NEW1 formulation by definition since it includes the constraints of the GP1 formulation and inequalities (5.13), in addition. Also, the NEW3 formulation is at least as good as the NEW1 formulation, since it includes the constraints of the GP1 formulation and inequalities (5.14) in addition. Finally the NEW4 formulation is at least as good as the NEW2 and NEW3 formulations since it includes the constraints of these two formulations. Previously, we have shown that the NEW1 formulation is stronger than both the GP1 and GP2 formulations. We continue this line of analysis also in this section and compare the NEW2, NEW3 and NEW4 formulations with the GP2, GP3 and GP4 formulations.

Proposition 5.2 The NEW3 formulation is at least as strong as GP3 the formulation

Proof. Adding constraints (5.14) for $(i,j) \in C$ and $p \notin V_C$ which is

$$x_{ij} + x_{ip} + x_{pj} + d_{ip} \leq d_{jp} + 1 \tag{5.15}$$

with C being the arc set of a given circuit $D_C = (V_C, C)$ such that $|C| \geq 2$ and $1 \notin V_C$ we obtain

$$\sum_{(i,j)\in C} x_{ij} + \sum_{k\in V_C} (x_{kp} + x_{pk}) \leq |C| \,. \tag{5.16}$$

This set of inequalities together with constraints (4.2)–(4.4) describe TP(GP3), which is the projection of P(GP3) into the subspace of x_{ij} variables [40]. Therefore, TP(NEW3) is a subset of TP(GP3) ∎

Moreover, as it can be observed from the computational results reported in Table 5.5 given in the next section, there are ATSP instances, such as `ftv35`, `ftv38`, `ftv47`, `ftv55`, `ftv64`, `ftv70`, `ft70`, and `ft53`, for which LP bound the NEW3 formulation has is strictly larger than the one of the GP3 formulation. This observation together with Proposition 5.2 implies that TP(NEW3) is in fact a proper subset of P(GP3) and therefore NEW3 formulation is stronger than the GP3 formulation.

In order to demonstrate that NEW4 formulation is stronger than the GP4 formulation we first show TP(NEW4), which is the projection of the polyhedron P(NEW4) into the subspace of x_{ij} variables, is a subset of TP(GP4), which is the projection of the polyhedron P(GP4) into the subspace of x_{ij} variables.

Proposition 5.3 TP(NEW4) is a subset of TP(GP4)

Proof. Adding constraint

$$x_{ji} + d_{kj} + d_{ji} - 1 \leq d_{ki}$$

with

$$x_{ij} - d_{ji} \leq 0$$

side by side results in the surrogate inequality

$$x_{ij} + x_{ji} + d_{kj} - 1 \leq d_{ki}.$$

Then

$$x_{ij} + x_{ji} + d_{ik} \leq d_{jk} + 1.$$

holds since $d_{ki} = 1 - d_{ik}$ and $d_{kj} = 1 - d_{jk}$. The latter inequality becomes equivalent to $x_{ij} + x_{ji} + v_{ki} \leq v_{kj} + 1$ as a consequence of identity

$$d_{ji} = v_{ij}, \tag{5.17}$$

between the additional variables d_{ij} and v_{ij} of the NEW and GP formulation families. Moreover, since $d_{ji} = 1 - d_{ij}$, constraint $x_{ij} - d_{ji} \leq 0$ becomes

$$x_{ij} + d_{ij} \leq 1,$$

which is again equivalent to $x_{ij} + v_{ji} \leq 1$ because of identity (5.17).

Similar arguments also show the equivalence between $x_{ij} + x_{ik} + x_{kj} + v_{ki} \leq v_{kj} + 1$ of the GP4 formulation and $x_{ij} + x_{ik} + x_{kj} + d_{ik} \leq d_{jk} + 1$ of the NEW4. Last inequality can be considered as a lifted version of $x_{ij} + v_{ki} \leq v_{kj} + 1$ of GP1 formulation as a consequence of (5.17). Therefore constraints of the GP4 formulation can be obtained

from the constraints of the NEW4 formulation by using the transformation (5.17). This shows the NEW4 formulation is at least as strong as GP4 formulation. ■

Notice that it is possible to say that the NEW4 formulation is as least as good as the GP4 formulation as a consequence of this proposition. In addition, in order to see that the NEW4 formulation is even stronger than the GP4 formulation, it is enough to solve the LP relaxations of both formulations on one of the instances `ftv35`, `ftv38`, `ftv55`, `ftv64`, `ftv70`, `ft70`, and `ft53`, and observe that the LP bound NEW4 formulation gives is strictly larger than the one GP4 formulation does. Table 5.5 of the next section provides examples.

Although they give very good LP bounds none of the extensions of the NEW1 formulation we have obtained so far is stronger than DFJ formulation or one of its multi-commodity flow equivalents, such as the CLAUS formulation. However, there is a fifth member of the NEW formulation family, which we call the NEW5 formulation with this property. Being inspired by the derivation of the seventh formulation of Gouveia and Pires [41], namely the GP7 formulation summarized in Section 2.6.2, we add to the constraint set of the NEW2 formulation multi-commodity flow constraints (4.55)–(4.58) and the affine transformation

$$\sum_{j=1}^{n} z_{kij} = d_{ik} \quad i,k = 2,\ldots,n. \tag{5.18}$$

In other words the NEW5 formulation consists of the constraint sets (4.1)–(4.5), (5.4') and (5.5'), (4.55)–(4.58) and (5.18), and the feasible solution set of its LP relaxation is

$$P(NEW5) = \left\{(x,d) \in \mathbb{R}^{|A|} \times \mathbb{R}^{|A|} : (4.2)-(4.4),(5.4'),(5.5'),(5.18),(4.55)-(4.58)\right\}.$$

It is not very difficult to see that the NEW5 formulation is not weaker than the GP7 formulation. This follows from three observations. First of all the GP7 formulation is obtained from the GP2 formulation by adding the multi-commodity flow constraints (4.55)–(4.58). Then similarly the NEW5 formulation is obtained from the NEW2

formulation by adding the same multi-commodity flow constraints, which means that the NEW5 formulation is not weaker than the NEW2 formulation. Finally, the NEW2 formulation is stronger than the GP2 formulation as a consequence of Proposition 5.1. At this step it is enough to provide a feasible $(x_{ij}, v_{ij}, z_{kij})$ vector of P(GP7), for which no (d_{ji}, z_{kij}) vector can exist such that $(x_{ij}, d_{ji}, z_{kij})$ is included in P(NEW5). Consider the following solution for a 5-vertex instance

$$
\begin{aligned}
&x_{13} = 2/3, \quad x_{15} = 1/3, \\
&x_{21} = 2/3, \quad x_{25} = 1/3, \\
&x_{34} = 2/3, \quad x_{35} = 1/3, \\
&x_{41} = 1/3, \quad x_{42} = 2/3, \\
&x_{52} = 1/3, \quad x_{53} = 1/3, \quad x_{54} = 1/3.
\end{aligned}
$$

Its feasibility for the GP7 formulation can be shown for the following (v_{ij}, z_{kij}) values:

$$
\begin{aligned}
&v_{24} = 1/3, \quad v_{25} = 1/3, \\
&v_{32} = 2/3, \quad v_{34} = 2/3, \quad v_{35} = 1/3, \\
&v_{42} = 2/3, \quad v_{43} = 1/3, \quad v_{45} = 1/3, \\
&v_{52} = 2/3, \quad v_{53} = 1/3, \quad v_{54} = 2/3, \\
&z_{221} = 2/3, \quad z_{225} = 1/3, \quad z_{241} = 1/3, \quad z_{254} = 1/3, \\
&z_{321} = 2/3, \quad z_{334} = 2/3, \quad z_{335} = 1/3, \quad z_{341} = 1/3, \quad z_{342} = 1/3, \\
&z_{352} = 1/3, \quad z_{421} = 2/3, \quad z_{435} = 1/3, \quad z_{441} = 1/3, \quad z_{442} = 2/3, \quad z_{453} = 1/3, \\
&z_{521} = 2/3, \quad z_{534} = 1/3, \quad z_{541} = 1/3, \quad z_{542} = 1/3, \quad z_{552} = 1/3, \quad z_{553} = 1/3, \quad z_{554} = 1/3.
\end{aligned}
$$

Consider MCF constraints (4.56)

$$
\begin{aligned}
&z_{221} + z_{223} + z_{224} + z_{225} = 1 \\
&z_{331} + z_{332} + z_{334} + z_{335} = 1 \\
&z_{441} + z_{442} + z_{443} + z_{445} = 1 \\
&z_{551} + z_{552} + z_{553} + z_{554} = 1
\end{aligned}
$$

and (4.57)

$$\begin{array}{llll} z_{221} \leq 2/3, & z_{223} \leq 0, & z_{224} \leq 0, & z_{225} \leq 1/3, \\ z_{331} \leq 0, & z_{332} \leq 0, & z_{334} \leq 2/3, & z_{335} \leq 1/3, \\ z_{441} \leq 1/3, & z_{442} \leq 2/3, & z_{443} \leq 0, & z_{445} \leq 0, \\ z_{551} \leq 0, & z_{552} \leq 1/3, & z_{553} \leq 1/3, & z_{553} \leq 1/3, \end{array}$$

in order to see that these values are in fact infeasible for P(NEW5). These two sets all together imply that

$$\begin{array}{ll} z_{221} = 2/3, & z_{225} = 1/3, \\ z_{334} = 2/3, & z_{335} = 1/3, \\ z_{441} = 1/3, & z_{442} = 2/3, \\ z_{552} = 1/3, & z_{553} = 1/3, \quad z_{554} = 1/3. \end{array}$$

Constraints (4.55) with these values and the transformation (5.18) become

$$d_{32} = z_{234} + z_{235} = z_{253} \tag{5.19}$$

$$d_{42} = z_{241} = z_{234} + z_{254} \tag{5.20}$$

$$d_{52} = z_{253} + z_{254} = 1/3 + z_{235} \tag{5.21}$$

$$d_{23} = z_{321} + z_{325} = z_{342} + z_{352} \tag{5.22}$$

$$d_{43} = z_{341} + z_{342} = 2/3 + z_{354} \tag{5.23}$$

$$d_{53} = z_{352} + z_{354} = z_{325} + 1/3 \tag{5.24}$$

$$d_{24} = z_{421} + z_{425} = 2/3 + z_{452} \tag{5.25}$$

$$d_{34} = z_{435} = z_{453} \tag{5.26}$$

$$d_{54} = z_{452} + z_{453} = z_{425} + z_{435} \tag{5.27}$$

$$d_{25} = z_{521} = z_{542} + 1/3 \tag{5.28}$$

$$d_{35} = z_{534} = 1/3 \tag{5.29}$$

$$d_{45} = z_{541} + z_{542} = z_{534} + 1/3 \tag{5.30}$$

Now, since $d_{35} = 1/3$ from (5.29) $d_{53} = 2/3$ directly follows by (5.4'), namely $d_{35} + d_{53} =$

1. From (5.24) we obtain, $z_{325} = 1/3$. Moreover, $z_{352} \leq 1/3$ and $z_{354} \leq 1/3$ by (4.44) and $z_{352} + z_{354} = d_{53} = 2/3$ by (5.24) imply $z_{352} = 1/3$ and $z_{354} = 1/3$. As a result $d_{43} = 1$ by (5.23), therefore $d_{34} = 0$ and $z_{435} = z_{453} = 0$ by (5.26). Then, we have $z_{341} = 1/3$ and $z_{342} = 2/3$ since $z_{341} \leq 1/3$, $z_{342} \leq 2/3$ by (4.44) and $z_{341} + z_{342} = d_{43} = 1$ by (5.23). We know that $z_{342} = 2/3$ and $z_{352} = 1/3$, hence $d_{23} = 1$ follows by (5.22). Therefore, $d_{32} = 0$ and $z_{234} = z_{235} = z_{253} = 0$ by (5.19). Hence, we end up with $d_{52} = 1/3$ and $z_{254} = 1/3$ by (5.21). Consequently, $d_{42} = z_{241} = 1/3$ by (5.20) and this gives $d_{24} = 2/3$ by (5.4') and considering (5.25) we obtain $z_{452} = 0$. Furthermore, since $z_{435} = z_{453} = 0$ by (5.26), $z_{425} = 0$ by (5.27). As a direct result $d_{54} = 0$. On the other hand, we have $z_{534} = 1/3$ by (5.29) which implies $d_{45} = 2/3$; but $d_{54} = 0$ from (5.27), and this contradicts (5.4'), since $d_{54} + d_{45} = 2/3 \neq 1$. Hence there exists an instance for which TP(NEW5) is a proper subset of TP(GP7). Therefore, the NEW5 formulation is stronger than the GP7 formulation.

We summarize our discussion on the relative strength of our five new formulations, with the diagram given in Figure 5.6. Again an arrow from the A formulation to the B formulation indicates that the LP relaxation of the B formulation is tighter than the LP relaxation of the A formulation, or equivalently the B formulation is stronger than the A formulation. Also, bidirectional arrows represent the equivalence of the A and B formulations. All of the formulations have $O(n^3)$ constraints and $O(n^2)$ variables with some exceptions: CLAUS, the GP7 and NEW5 formulations have $O(n^3)$ constraints but $O(n^3)$ variables instead, and the GP5, GP6 and GP8 formulations have $O(2^n)$ constraints and $O(n^3)$ variables.

Notice that there are two relations which have not been explained yet. We first explain why P(NEW5) is a proper subset of P(NEW4) and thus the NEW5 formulation is stronger than NEW4 formulation. Recall that the NEW4 formulation consists of assignment constraints (4.2) and (4.3), subtour elimination constraints (5.4') and (5.5'), (5.13) and (5.14), while the NEW5 formulation is defined by assignment constraints (4.2) and (4.3), multi-commodity flow constraints (4.55)–(4.58), subtour elimination constraints (5.4') and (5.5'), (5.13), and linear transformation (5.18). As we have mentioned in Section 2.6.2, Gouveia and Pires [41] have shown that it is possible to obtain

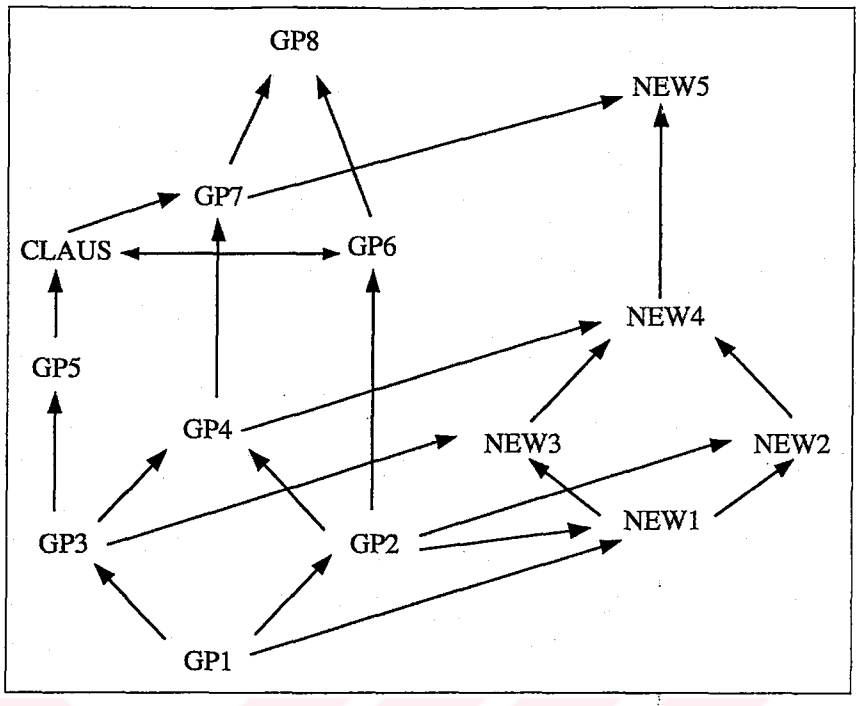

Figure 5.2. Relative strength of NEW formulation family

subtour elimination constraints (4.52) from constraints (4.57), (4.58) and a weaker version of (4.55) by using the linear transformation (4.60). For the sake of clarity, the derivation of (5.14), from a weaker version of (4.55) and linear transformation (5.18) is explained below. This weaker version of (4.55) is

$$\sum_{t=1}^{n} z_{kjt} \geq z_{kij} + z_{kkj} \qquad i,j,k = 2,\ldots,n; i \neq j \neq k.$$

It is equivalent to

$$d_{jk} = \sum_{t=1}^{n} z_{kjt} \geq z_{kij} + x_{kj} \qquad i,j,k = 2,\ldots,n; i \neq j \neq k.$$

Then by adding, $\sum_{t=1,t\neq j}^{n} z_{kit}$ to both sides and using the relation $\sum_{t=1,t\neq j}^{n} z_{kit} + z_{kij} = \sum_{t=1}^{n} z_{kit} = d_{ik}$ and considering (5.18) we obtain

$$\sum_{t=1,t\neq j}^{n} z_{kit} + d_{jk} \geq d_{ik} + x_{kj} \qquad i,j,k = 2,\ldots,n; i \neq j \neq k.$$

Since $x_{ij} \geq z_{kij}$ by (4.57) and z_{kik} is not defined, we have $\sum_{t=1,t\neq j,k}^{n} x_{it} \geq \sum_{t=1,t\neq j,k}^{n} z_{kit}$. Consequently,

$$\sum_{t=1,t\neq j,k}^{n} x_{it} + d_{jk} \geq d_{ik} + x_{kj} \qquad i,j,k = 2,\ldots,n; i \neq j \neq k$$

follows. On the other hand assignment constraints (4.2) imply $\sum_{t=1,t\neq j,k}^{n} x_{it} + x_{ij} + x_{ik} = 1$ resulting in

$$1 - x_{ij} - x_{ik} + d_{jk} \geq d_{ik} + x_{kj} \qquad i,j,k = 2,\ldots,n; i \neq j \neq k,$$

which are constraints (5.14). Hence, the NEW5 formulation, includes constraints of the NEW4 formulation, subtour elimination constraints (4.42)–(4.44), and the linear transformation (5.18). As a result, the NEW5 formulation is at least as strong as the NEW4 formulation. Now consider the following solution $(\overline{x}, \overline{d})$:

$$\begin{aligned}
&\overline{x}_{13} = 0.4, \quad \overline{x}_{16} = 0.6,\\
&\overline{x}_{23} = 0.2, \quad \overline{x}_{24} = 0.6, \quad \overline{x}_{25} = 0.2,\\
&\overline{x}_{31} = 0.8, \quad \overline{x}_{34} = 0.2,\\
&\overline{x}_{41} = 0.2, \quad \overline{x}_{42} = 0.4, \quad \overline{x}_{46} = 0.4,\\
&\overline{x}_{52} = 0.6, \quad \overline{x}_{53} = 0.4,\\
&\overline{x}_{64} = 0.2, \quad \overline{x}_{65} = 0.8.
\end{aligned}$$

$$\begin{aligned}
&\overline{d}_{23} = 0.4, \quad \overline{d}_{24} = 0.4, \quad \overline{d}_{25} = 0.6, \quad \overline{d}_{26} = 0.6,\\
&\overline{d}_{32} = 0.6, \quad \overline{d}_{34} = 0.6, \quad \overline{d}_{35} = 0.6, \quad \overline{d}_{36} = 0.6,\\
&\overline{d}_{42} = 0.6, \quad \overline{d}_{43} = 0.4, \quad \overline{d}_{45} = 0.6, \quad \overline{d}_{46} = 0.6,\\
&\overline{d}_{52} = 0.4, \quad \overline{d}_{53} = 0.4, \quad \overline{d}_{54} = 0.4, \quad \overline{d}_{56} = 0.8,\\
&\overline{d}_{62} = 0.4, \quad \overline{d}_{63} = 0.4, \quad \overline{d}_{64} = 0.4, \quad \overline{d}_{65} = 0.2.
\end{aligned}$$

It is in P(NEW4). Observe that, these x_{ij} values violate clique inequalities for the set $S = \{2,4,5,6\}$. In other words they are infeasible for P(DFJ). However, it is

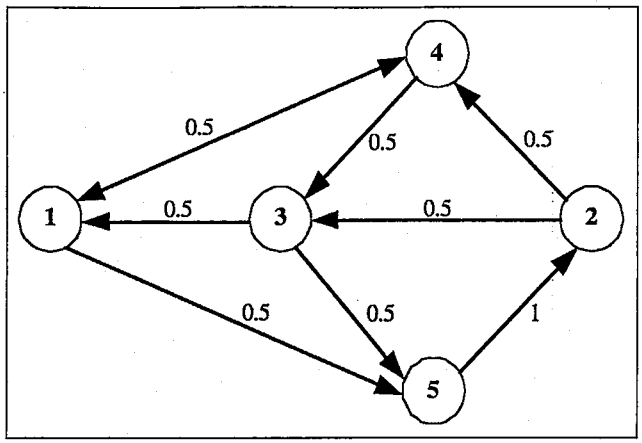

Figure 5.3. A solution which is feasible for P(GP4) but infeasible for P(NEW3)

known that P(DFJ) and TP(CLAUS), the projection of P(CLAUS) into the subspace of x_{ij} variables, are equivalent [15]. Thus no $\overline{z}$ can exist such that $(\overline{x}, \overline{z})$ vector is in P(CLAUS). Since the constraint set of the NEW5 formulation includes the constraint set of CLAUS formulation, P(NEW5) is a subset of P(CLAUS) and no $\overline{z}$ can exist such that $(\overline{x}, \overline{d}, \overline{z})$ vector is in P(NEW5). Therefore there exists an instance for which the projection of P(NEW5) into the subspace of (x_{ij}, d_{ij}) variables is a proper subset of P(NEW4), and the NEW5 formulation is stronger than the NEW4 formulation.

5.3. Formulations Incomparable with the New Ones

It is known that not all of the ATSP formulations are comparable. As for example, Gouveia and Pires provide solutions one of which is feasible for P(GP2) but infeasible for P(GP3), while the other is feasible for P(GP3) but infeasible for P(GP2) in its turn [40]. Examples showing DL and SD formulations are incomparable with the GP1 – GP4 formulations are provided in Chapter 4. Incomparabilities also exist between the members of NEW formulation family and the others explaining why some of the arrows are missing in Figure 5.6. We report results on this issue in the rest of this section.

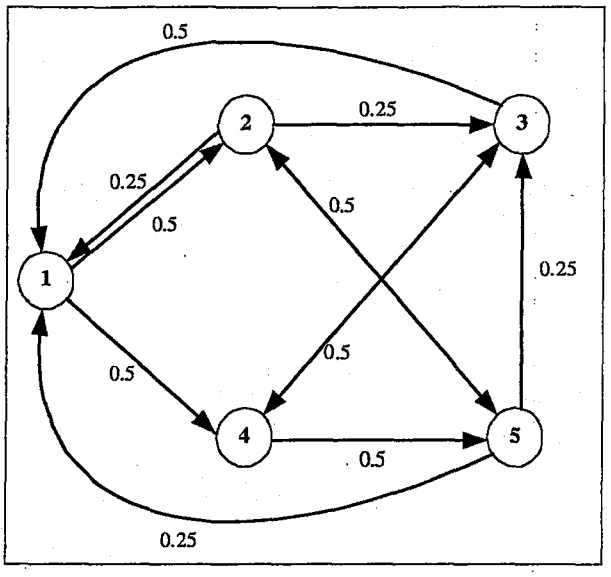

Figure 5.4. A solution which is feasible for P(NEW3) but infeasible for P(GP4)

5.3.1. Gouveia and Pires' Fourth Formulation

Consider the solution represented with the support digraph of Figure 5.3, which is a feasible solution in P(GP4). However, it is infeasible for NEW3, namely it is not in P(NEW3). To see this add inequalities

$$\begin{aligned} x_{23} + x_{24} + x_{43} + d_{24} &\leq d_{34} + 1 \\ x_{23} + d_{42} + d_{23} &\leq d_{43} + 1 \\ x_{32} + x_{35} + x_{52} + d_{35} &\leq d_{25} + 1 \\ x_{32} + d_{53} + d_{32} &\leq d_{52} + 1 \end{aligned} \tag{5.31}$$

side by side and use relations $d_{34} + d_{43} = 1$, $d_{25} + d_{52} = 1$, $d_{24} + d_{42} = 1$, $d_{23} + d_{32} = 1$, $d_{53} + d_{35} = 1$ to obtain the surrogate inequality

$$x_{23} + x_{24} + x_{43} + x_{23} + x_{32} + x_{35} + x_{52} + x_{32} \leq 3.$$

It is violated by the solution of Figure 5.3 since the left hand side of the above inequality adds up to 3.5, which implies that, in fact, this solution violates inequalities (5.31) belonging NEW3 formulation for a complete digraph of five vertices.

On the other hand, the solution whose support digraph given in Figure 5.4 is feasible for P(NEW3) but infeasible for P(GP4): Adding the following inequalities, which belong to the constraints of GP4 formulation,

$$x_{43} + x_{45} + x_{53} + v_{54} \leq v_{53} + 1$$
$$x_{43} + x_{34} + v_{53} \leq v_{54} + 1$$

we obtain

$$x_{43} + x_{45} + x_{53} + x_{43} + x_{34} \leq 2.$$

It is violated since its left hand side has a value of 2.25. As a result, based on the solutions whose support digraphs are given in Figure 5.3 and Figure 5.4 it is possible to say that the NEW3 and GP4 formulations are incomparable.

5.3.2. Claus' Formulation

Although we have not been able to achieve an algebraic expression for the projection of the NEW1 formulation and its extensions into the subspace of x_{ij} variables we have shown that the NEW1 formulation is stronger than GP1 formulation. Since P(GP1) is equivalent to P(CP). One may dare to claim that constraints of the three extensions of the NEW1 formulation, namely the NEW2, NEW3 and NEW4 formulations, correspond to some lifted circuit inequalities after P(NEW2), P(NEW3) and P(NEW4) are projected into the subspace of x_{ij} variables. This is the case with the NEW3 formulation: Its constraints imply one of the D_4 inequalities, which are shown to be facet defining for the ATSP polytope [42]. This can be seen by adding up constraints

$$x_{ki} + x_{kl} + x_{li} + d_{kl} \leq d_{il} + 1$$
$$x_{ik} + x_{ip} + x_{pk} + d_{ip} \leq d_{kp} + 1$$
$$x_{ik} + d_{pi} + d_{ik} \leq d_{pk} + 1$$
$$x_{ki} + d_{lk} + d_{ki} \leq d_{li} + 1$$

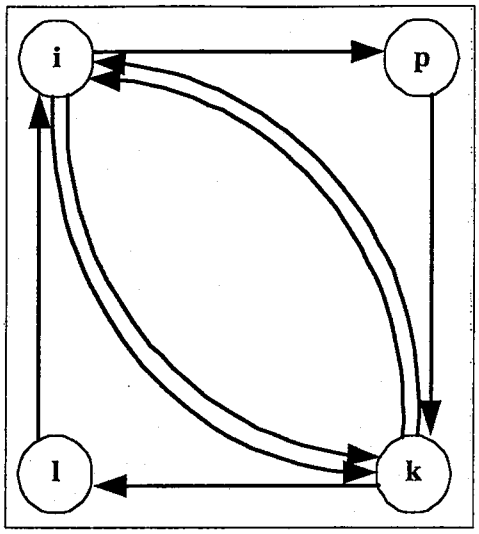

Figure 5.5. Support graph of a D_4 inequality for vertices i, k, l, and p

side by side and using equalities $d_{kl} + d_{lk} = 1, d_{ik} + d_{ki} = 1, d_{ip} + d_{pi} = 1, d_{il} + d_{li} = 1$ and $d_{pk} + d_{kp} = 1$ to obtain

$$x_{ip} + x_{pk} + x_{kl} + x_{li} + 2x_{ik} + 2x_{ki} \leq 3.$$

The last inequality is one of the D_4 inequalities, which is illustrated with the supporting digraph given in Figure 5.5.

In their early work Grötschel and Padberg have shown that D_k inequalities are not dominated by clique inequalities [42]. Therefore, the NEW3 formulation, and thus its stronger version the NEW4 formulation are not weaker than DFJ formulation (or equivalent to CLAUS formulation). However, there exists instances for which the LP bound CLAUS formulation gives larger than the ones the NEW3 and NEW4 formulations do. In short, CLAUS formulation, and the NEW3 and NEW4 formulations are not comparable.

5.3.3. Desrochers and Laporte's, and Sherali and Driscoll's Formulations

Similar to the GP1 – GP4 and DL formulations and the GP1 – GP4 and SD formulations, the NEW1 – NEW4 formulations and the DL and SD formulations are

incomparable. We first start by showing that DL formulation is not stronger than the NEW1 formulation. For this purpose consider the solution given in Section 4.2.9, which is feasible for P(DL) but infeasible for P(GP1). It is possible to see that it is also infeasible for P(NEW1). Side by side addition of inequalities (5.5') for $(i,j,k) = (2,5,6)$, $(i,j,k) = (5,4,6)$, $(i,j,k) = (4,2,6)$, and $(i,j,k) = (4,5,2)$ gives

$$1 + d_{65} \geq d_{62} + d_{25} + x_{25}$$
$$1 + d_{64} \geq d_{65} + d_{54} + x_{54}$$
$$1 + d_{62} \geq d_{64} + d_{42} + x_{42}$$
$$1 + d_{25} \geq d_{24} + d_{45} + x_{45}.$$

By using (5.4') for $(i,j) = (4,5)$ and $(i,j) = (2,4)$ results in the contradiction $4 \ngeq 4.4 = x_{25} + x_{54} + x_{42} + d_{42} + d_{24} + d_{45} + d_{54}$. Note that, this solution is also infeasible for P(NEW2) – P(NEW4) since P(NEW2) – P(NEW4) are proper subsets of P(NEW1). Therefore, the DL formulation is not stronger than the NEW1 – NEW4 formulations.

Now consider

$$x_{13} = 0.5, \quad x_{15} = 0.5,$$
$$x_{23} = 0.5, \quad x_{24} = 0.5,$$
$$x_{32} = 0.5, \quad x_{35} = 0.5,$$
$$x_{41} = 0.5, \quad x_{42} = 0.5,$$
$$x_{51} = 0.5, \quad x_{54} = 0.5,$$

$$d_{23} = 0.5, \quad d_{24} = 0.5, \quad d_{25} = 0.5,$$
$$d_{32} = 0.5, \quad d_{34} = 0.5, \quad d_{35} = 0.5,$$
$$d_{42} = 0.5, \quad d_{43} = 0.5, \quad d_{45} = 0.5,$$
$$d_{52} = 0.5, \quad d_{53} = 0.5, \quad d_{54} = 0.5,$$

and all other variables are equal to zero. It is feasible for P(NEW4) but infeasible for P(DL). Notice that x_{ij} values are the same as the solution given in Section 4.2.9.1

which is feasible for P(GP4) but infeasible for P(DL). Therefore, the DL formulation is not stronger than the NEW4 formulation. On the other hand, this solution is also feasible for P(NEW1) – P(NEW3) since P(NEW4) is a proper subset of these polyhedra. As a result, the NEW1 – NEW4 formulations are not stronger than DL formulation. Therefore, the DL formulation and the NEW1 – NEW4 formulations are not comparable.

To show that the NEW4 formulation is not stronger than the SD formulation consider

$$\begin{aligned} x_{13} &= 0.5, \quad x_{15} = 0.5, \\ x_{23} &= 0.5, \quad x_{24} = 0.5, \\ x_{32} &= 0.5, \quad x_{35} = 0.5, \\ x_{41} &= 0.5, \quad x_{42} = 0.5, \\ x_{51} &= 0.5, \quad x_{54} = 0.5, \end{aligned}$$

$$\begin{aligned} d_{23} &= 0.5, \quad d_{24} = 0.5, \quad d_{25} = 0.5, \\ d_{32} &= 0.5, \quad d_{34} = 0.5, \quad d_{35} = 0.5, \\ d_{42} &= 0.5, \quad d_{43} = 0.5, \quad d_{45} = 0.5, \\ d_{52} &= 0.5, \quad d_{53} = 0.5, \quad d_{54} = 0.5, \end{aligned}$$

and all other variables are equal to zero. It is feasible for P(NEW4). Recall that these given x_{ij} values are the same as the solution given above which is feasible for P(GP4) but infeasible for P(SD). Therefore we can perform the same operations to show its infeasibility for P(SD). Since, P(NEW4) is a proper subset of P(NEW1) – P(NEW3), this solution is also feasible for the polyhedra P(NEW1) – P(NEW3). Therefore, the NEW1 – NEW4 formulations are not stronger than the SD formulation.

Finally consider the solution given above which is feasible for P(SD) but infeasible for P(GP1) and the circuit polytope P(CP). Since P(NEW1) includes the circuit inequalities and P(NEW1) is a proper subset of P(GP1), this solution is also infeasible

for P(NEW1). Moreover, this solution is also infeasible for P(NEW2) – P(NEW4) since we know that the polyhedra P(NEW2) – P(NEW4) are proper subsets of P(NEW1). Hence, the SD formulation is not stronger than the NEW1 – NEW4 formulations. Therefore, we can conclude the NEW1 – NEW4 formulations and the SD formulation are incomparable.

One more missing relation is between the GP8 and NEW5 formulations. However, we believe that showing it does not have great importance from application point of view since GP8 formulation has exponential subtour elimination constraints while NEW5 formulation having only $O(n^3)$.

As a final issue, it may be interesting to question whether it is possible to combine the efficiency of the MTZ formulation with the strength of the NEW formulation family. One quick answer may be the aggregation of the distance inequalities (5.5) to obtain some new relatives of the MTZ subtour elimination constraints, which may increase the number of auxiliary variables.

5.4. New Aggregated Formulations

5.4.1. Aggregating the Assignment Constraints

In this section we show that it is possible to obtain a more compact formulation by replacing the $2n$ assignment constraints (5.2) and (5.3) by the equality

$$\sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} x_{ij} = n \tag{5.32}$$

obtained by aggregating the first n (or similarly the last n) of them. This new formulation will be denoted as NEW6. It consists of subtour elimination constraints (5.4) and (5.5), nonnegativity constraints (5.6), integrality constraints (5.7) and the equalities (5.32).

Recall that the ATSP digraph $D = (V, A)$ is simply a complete digraph over vertex set $V = \{1, \ldots, n\}$. Now, consider inequalities (5.5) for a vertex $i \in V$ and k of its inarcs $(j, i) \in A$. Since this is a complete digraph the total number of inarcs are $|V| - 1$. Assume without loss of generality these k inarcs are $(1, i)$, $(2, i)$, $\ldots$, (k, i), and these k inequalities have the following form:

$$\begin{aligned}
d_{2i} &\geq d_{21} + d_{1i} + x_{1i} - 1 \\
d_{3i} &\geq d_{32} + d_{2i} + x_{2i} - 1 \\
d_{4i} &\geq d_{43} + d_{3i} + x_{3i} - 1 \\
&\vdots \\
d_{ki} &\geq d_{kk-1} + d_{k-1i} + x_{k-1i} - 1 \\
d_{1i} &\geq d_{1k} + d_{ki} + x_{ki} - 1.
\end{aligned}$$

When we sum them up side by side we obtain

$$0 \geq \left(\sum_{i=1}^{k-1} d_{i+1i} + d_{1k} \right) + \sum_{j=1}^{k} x_{ji} - k. \tag{5.33}$$

because of cancellations. The expression within the parenthesis is to 1 as a consequence of Theorem 2.1 and inequality (5.33) reduces to

$$k - 1 \geq \sum_{j=1}^{k} x_{ji}.$$

We note that, the last inequality can be driven for any $k = 2, \ldots, n-1$, which implies

$$\sum_{j=1}^{k} x_{ji} \leq 1 \qquad i = 1, \ldots, n; i \neq j \tag{5.34}$$

As a consequence of this discussion we can state the following lemma.

Lemma 5.1 For any vertex $i \in V$, the subtour elimination constraints (5.4) and (5.5) with the integrality restrictions (5.7) allow at most one of the variables x_{ji} to be 1.

In other words every city can be entered by the travelling salesman at most one time. This result becomes stronger when leaving arcs considered instead of entering arcs. Let us apply similar argument to vertex i on any k of its leaving arcs; they can be assumed to be $(i,1)$, $(i,2)$, ..., (i,k) without loss of generality. Then, by adding the inequalities (5.5) side by side

$$\begin{aligned} d_{21} &\geq d_{2i} + d_{i1} + x_{i1} - 1 \\ d_{32} &\geq d_{3i} + d_{i2} + x_{i2} - 1 \\ d_{43} &\geq d_{4i} + d_{i3} + x_{i3} - 1 \\ &\vdots \\ d_{kk-1} &\geq d_{ki} + d_{ik-1} + x_{ik-1} - 1 \\ d_{1k} &\geq d_{1i} + d_{ik} + x_{ik} - 1 \end{aligned}$$

and by using $d_{ji} + d_{ij} = 1$, for $j = 1, \ldots, k$ we obtain

$$\left(\sum_{i=1}^{k-1} d_{i+1i} + d_{1k} \right) \geq k + \sum_{j=1}^{k} x_{ij} - k. \tag{5.35}$$

The expression within the parenthesis is equal to 1 as a consequence of Theorem 2.1 and inequality (5.35) reduces to

$$\sum_{j=1}^{k} x_{ij} \leq 1 \qquad i = 1, \ldots, n; i \neq j, \tag{5.36}$$

which completes the proof of the next lemma.

Lemma 5.2 For any vertex $i \in V$, the subtour elimination constraints (5.4) and (5.5) with the integrality restrictions (5.7) allow at most one of the variables x_{ij} to be 1.

In other words every city must be left no more than one time by the travelling salesman. We note that, this inequality is independent of the number of arcs. There are two major implications of Lemma 5.1 and Lemma 5.2. First of all constraints (5.4)

and (5.5) do not eliminate only subtours but also trees with vertices of more than 1 in- and outdegrees, when considered together with the integrality restrictions (5.7). Namely only paths are allowed. These constraints can be seen stronger than any other known subtour elimination constraints because of this property. However, it has an important drawback: it is not possible to benefit from any kind of tree relaxations in order to produce good lower bounds. The second implication is the redundancy of the assignment constraints: Subtour elimination constraints (5.4) and (5.5) allow at most one entrance and one exit for each city when considered with the integrality restrictions (5.7). However, dropping off the assignment constraints completely results in the trivial optimal solution where all variables have zero value. Hence, we have to force the formulation to have a Hamiltonian circuit in its solution, which can be done by considering equalities (5.32) with the constraints (5.4) – (5.7), which yields

$$\text{NEW6:} \qquad \min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} \qquad (5.37)$$

$$\text{s.t. (5.32), (5.4)–(5.7)}$$

as the new extended ATSP formulation. It has $n(n-1)^2 + 1$ constraints.

Theorem 5.1 Formulation (5.37) is a valid ATSP formulation.

Proof. Constraints (5.4) and (5.5) do not only eliminate subtours, but also force every city to have exactly one entrance and one exit with the aggregated equality (5.32) and the integrality restrictions (5.7) as a consequence of Lemma 2.1 and Lemma 2.2. ■

5.4.2. Aggregating the Subtour Elimination Constraints

Consider any subset S of the vertices V with size $|S| \geq 3$. If we add constraints (5.5) for a given pair (k, j) over all $i \in S/\{k, j\}$ we obtain

$$\sum_{i \in S/\{k,j\}} d_{ij} \geq \sum_{i \in S/\{k,j\}} d_{ik} + \sum_{i \in S/\{k,j\}} d_{kj} + \sum_{i \in S/\{k,j\}} x_{kj} - (|S| - 2).$$

In addition when we add side by side $1 = d_{kj} + d_{jk}$ and $0 = d_{kj} - d_{kj}$ to the last inequality we end up with

$$1+(|S|-2) \geq \sum_{i \in S/\{k,j\}} d_{ik}+d_{jk}+ \sum_{i \in S/\{k,j\}} d_{kj}+d_{kj}- \sum_{i \in S/\{k,j\}} d_{ij}-d_{kj}+d_{kj}+ \sum_{i \in S/\{k,j\}} x_{kj},$$

which is equivalent to

$$|S|-1 \geq \sum_{i \in S/\{k\}} d_{ik} - \sum_{i \in S/\{j\}} d_{ij} + |S|\, d_{kj} + (|S|-2)x_{kj}.$$

The last inequality becomes

$$u_{S_k} - u_{S_j} + (|S|-2)x_{kj} + |S|\, d_{kj} \leq |S| - 1 \tag{5.38}$$

for

$$u_{S_p} = \sum_{i \in S/\{p\}} d_{ip}.$$

Notice that, for $S \equiv V$ we have

$$u_{V_k} - u_{V_j} + (n-2)x_{kj} + nd_{kj} \leq n-1 \text{ for } k,j = 1,\ldots,n; k \neq j, \tag{5.39}$$

as the special form of (5.38). Summing up (5.38) for (k,j) on a given circuit $D_C = (V_C, C)$ with vertex set $V_C \subseteq S$ and arc set C we obtain

$$(|S|-2) \sum_{(k,j) \in C} x_{kj} + |S| \sum_{(k,j) \in C} d_{kj} \leq (|S|-1)\,|C|\,. \tag{5.40}$$

We can also consider reverse arcs (j,k) on the circuit vertices V_C and write in-

equalities

$$u_{S_j} - u_{S_k} + (|S| - 2)x_{jk} + |S|\, d_{jk} \leq |S| - 1. \tag{5.41}$$

They can be obtained as inequalities (5.38) by using the same argument. Then, it is possible to end up with

$$(|S| - 2) \sum_{(k,j)\in C} x_{jk} + |S| \sum_{(k,j)\in C} d_{jk} \leq (|S| - 1)\,|C| \tag{5.42}$$

by summing them up inequalities (5.41) on the circuit $D_C = (V_C, C)$. Since $x_{kj} = 1$ and $x_{jk} = 0$ because of the assignment constraints (5.40) and (5.42) reduce respectively to

$$(|S| - 2)\,|C| + |S| \sum_{(k,j)\in C} d_{kj} \leq (|S| - 1)\,|C| \tag{5.43}$$

and

$$|S| \sum_{(k,j)\in C} d_{jk} \leq (|S| - 1)\,|C|\,, \tag{5.44}$$

from which

$$\sum_{(k,j)\in C} d_{kj} \leq \frac{|C|}{|S|} \tag{5.45}$$

and

$$\sum_{(k,j)\in C} d_{jk} \leq \frac{|S| - 1}{|S|}\,|C|\,. \tag{5.46}$$

Meanwhile the aggregation of the circularity constraints (5.4) on arcs (k, j) for the circuit D_C results in

$$\sum_{(k,j)\in C} d_{jk} + \sum_{(k,j)\in C} d_{kj} = |C|\,. \tag{5.47}$$

Notice that inequalities (5.45) and (5.46) imply

$$\sum_{(k,j)\in C} d_{kj} + \sum_{(k,j)\in C} d_{jk} \leq \frac{|C|}{|S|} + \frac{|S|-1}{|S|}|C| = |C|\,,$$

which does not contradict (5.47).

Notice that

$$x_{ij} \leq d_{ji} \quad i,j = 2,\ldots,n; i \neq j \tag{5.48}$$

has been used to strengthen the NEW1 formulation. When we add them up over the arcs of the same circuit $D_C = (V_C, C)$, we end up with

$$\sum_{(k,j)\in C} x_{kj} \leq \sum_{(k,j)\in C} d_{jk},$$

implying

$$\sum_{(k,j)\in C} d_{jk} \geq |C| \tag{5.49}$$

since $x_{kj} = 1$ for all circuit arcs (k,j), which gives the contradiction $|S|-1 \geq |S|$ when considered together with inequality (5.46). A direct consequence of this discussion is the following formulation having exactly $2n+2n(n-1)+(n-1)(n-2) = 3n^2-3n+2$ constraints and $n(2n-1)$ variables:

$$\text{NEW7:} \qquad \min \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij}x_{ij} \tag{5.50}$$

$$\text{s.t. (5.2) – (5.3), (5.39), (5.48).}$$

Theorem 5.2 Formulation (5.50) is a valid ATSP formulation.

Proof. Follows immediately from the previous discussion. ■

5.5. Computational Results

In this section we report results on the quality of the LP bounds obtained with the NEW formulation family. Table 5.1 includes symmetric and asymmetric ATSP instances used in the computations and their optimum objective values. They are obtained from the TSPLIB [61] and used in similar earlier works as the test bed [40, 48]. Computational experiments are represented in two subsections. In the first one we report results obtained with formulations having $O(n^3)$ constraints. The second subsection includes the ones obtained with formulations having $O(n^2)$ constraints: The only exception is the NEW6 formulation. It has $O(n^3)$ constraints; but we still consider it as a compact formulation since it has single aggregated constraint instead of $2n$ assignment constraints. There are two types of tables in both sections. On the first group we present relative deviations from the optimal tour lengths. They are calculated according to the formula

$$100 \times \left(\frac{z^*_{IP} - z^*_{LP}}{z^*_{IP}} \right)$$

where z^*_{IP} is the length of an optimal tour and z^*_{LP} is the LP bound obtained by solving the LP relaxation of the models.

The second type of tables include CPU times spent for computing corresponding bounds with the barrier solver of Cplex ver. 7.0. Default options are selected. We prefer using barrier solver since the LP relaxations are quiet large and the use of interior point methods can decrease the solution time remarkably. However, barrier demands larger memory space which can cause problems for large linear programs as it is the case with the LP relaxation of the NEW5 formulation. For that reason there is no entry for ft53, ftv55, ftv64, ft70 and ftv70 in Table 5.3 and Table 5.5. For the same reason there is no entry for eil51, st70, eil76 and pr76 in Table 5.2 and Table 5.4. Therefore, the average of the NEW5 columns are taken over existing values and underestimates the true average in Tables 5.2 – 5.4. It is probably possible to obtain better memory

and CPU performances by tuning Cplex's parameters carefully. No value is reported with the GP5 – GP8 formulations since the CPU and memory resources they demand for computing the LP bounds are prohibitive. Our computations are realized on a Sun Microsystems Blade – 1000 with a 750 MHz Ultrasparc III CPU and 2 GByte RAM, working within SOLARIS 8 environment.

Each subsection includes also; one figure for symmetric instances and one for asymmetric instances. These are basically plots of average relative deviations versus average CPU times for each formulation. We believe they expose better the performances of the considered formulations.

5.5.1. Experiments with $O(n^3)$ Constrained Formulations

As it can be observed from Table 5.2 the CLAUS and NEW5 formulations give the best bounds. This is not surprising since NEW5 formulation is theoretically stronger than the CLAUS formulation which is theoretically as strong as the DFJ formulation in its turn. Notice that the average of NEW5 column is obtained by solving the LP relaxations of the first seven instances. We have not been able to solve the LP relaxations of NEW5 formulation on larger instances because of memory limitations. The NEW3, NEW4, GP3 and GP4 formulations are the second bests; They give the same relative deviations although the NEW3 and NEW4 formulations have been theoretically shown to be stronger than the GP3 and GP4 formulations. The strength of the new formulations become clearer on asymmetric instances. According to the values reported in the first five rows of Table 5.3, the NEW5 formulation gives the best LP bounds. This should not change even when the remaining instances are solved. The second best is the CLAUS formulation. This time the NEW3 and NEW4 formulations perform better than GP3 and GP4 formulations. We note again that the average of the NEW5 column is not directly comparable since it is obtained with only seven of the test instances. Despite this shortcoming, the NEW5 formulation gives better LP bounds that the CLAUS formulation does. Therefore we would expect an average LP bound lower than the one of the CLAUS formulation gives when all the instances are solved using the NEW5 formulation.

Table 5.1. Test instances and their optimum objective values

Symmetric Instances	Optimal Tour Length	Asymmetric Instances	Optimal Tour Length
gr21	2707	ftv33	1286
gr24	1272	ftv35	1473
bayg29	1610	ftv38	1530
bays29	2020	ftv44	1613
att48	10628	ftv47	1776
gr48	5046	ft53	6905
hk48	11461	ftv55	1608
eil51	426	ftv64	1839
eil76	538	ft70	38673
pr76	108159	ftv70	1950
st70	675	kroA124p	36230

Table 5.4 and Table 5.5 include CPU times spent for computing bounds. The NEW5 formulation is the slowest on both of the symmetric and asymmetric instances in the average. This is expected since the number of constraints it has is almost twice of any other model has. In short, regarding to bound-per-resource performance we can say that the NEW formulation family gives good LP bounds in a reasonable amount of time, which make them considerable for use in real life applications. This can be observed more clearly from Figure 5.6 and Figure 5.7. We must point out that better LP bounds can be obtained much faster by solving the LP relaxation of the DFJ formulation using constraint generation [62]. This decreases the probability that one of the models of the NEW and GP formulation families is used in the exact solution of the ATSP. However, they become more attractive for problems where additional variables also have costs and can not be removed. Gouveia and Pires provide examples for such cases [40].

Table 5.2. Relative per cent deviations for symmetric instances

Instance	NEW1	NEW2	NEW3	NEW4	NEW5	GP1	GP2	GP3	GP4	CLAUS
gr21	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
gr24	3.734	3.734	0.314	0.314	0.000	3.734	3.734	0.314	0.314	0.000
bayg29	5.186	5.186	2.112	2.112	0.124	5.186	5.186	2.112	2.112	0.124
bays29	4.752	4.752	2.921	2.921	0.322	4.752	4.752	2.921	2.921	0.322
att48	6.337	6.337	6.224	6.224	0.226	6.337	6.337	6.224	6.224	0.226
gr48	5.509	5.509	5.509	5.509	1.724	5.509	5.509	5.509	5.509	1.724
hk48	2.303	2.303	1.418	1.418	0.144	2.303	2.303	1.418	1.418	0.144
eil51	2.230	2.230	2.230	2.230	na	2.230	2.230	2.230	2.230	0.704
st70	7.704	7.704	3.704	3.704	na	7.704	7.704	3.704	3.704	0.148
eil76	0.000	0.000	0.000	0.000	na	0.000	0.000	0.000	0.000	0.000
pr76	8.473	8.473	4.442	4.442	na	8.473	8.473	4.442	4.442	2.810
Average	4.203	4.203	2.625	2.625	0.363	4.203	4.203	2.625	2.625	0.564

Table 5.3. Relative per cent deviations for asymmetric instances

Instance	NEW1	NEW2	NEW3	NEW4	NEW5	GP1	GP2	GP3	GP4	CLAUS
ftv33	4.426	4.426	0.000	0.000	0.000	4.768	4.646	0.000	0.000	0.000
ftv35	3.225	3.225	1.202	1.202	0.651	3.404	3.236	1.319	1.288	1.066
ftv38	2.912	2.912	1.157	1.157	0.641	3.201	3.088	1.270	1.240	1.024
ftv44	1.991	1.991	1.836	1.836	1.120	2.118	1.991	1.836	1.836	1.743
ftv47	2.365	2.365	1.858	1.858	0.779	2.528	2.412	1.896	1.858	1.542
ft53	12.210	12.210	9.870	9.870	na	12.548	12.439	11.019	10.676	0.000
ftv55	4.167	4.167	2.138	2.138	na	4.740	4.571	2.539	2.332	1.493
ftv64	3.800	3.800	2.678	2.678	na	4.001	3.982	3.144	3.144	1.713
ft70	1.171	1.171	1.107	1.107	na	1.271	1.252	1.146	1.144	0.053
ftv70	4.251	4.251	2.809	2.809	na	4.463	4.374	3.536	3.523	2.103
kroA124p	1.842	1.842	0.735	0.724	na	2.319	1.972	1.126	0.981	0.637
Average	3.851	3.851	2.308	2.307	0.638	4.124	3.997	2.621	2.547	1.034

Table 5.4. CPU times (in seconds) with barrier option for symmetric instances

Instance	NEW1	NEW2	NEW3	NEW4	NEW5	GP1	GP2	GP3	GP4	CLAUS
gr21	1.55	1.62	2.51	2.67	60.73	2.54	2.67	2.85	3.91	4.57
gr24	3.16	3.26	4.89	4.93	154.8	5.23	5.36	5.99	7.68	8.91
bayg29	8.51	9.88	15.21	15.86	621.2	16.42	20.41	21.20	26.41	24.27
bays29	8.39	8.41	13.24	13.13	603.7	15.30	18.80	19.30	21.07	22.87
att48	169.43	170.12	241.53	251.93	21326	335.52	397.36	379.30	440.35	364.43
gr48	167.39	168.50	210.07	221.97	19694	308.53	402.85	398.60	436.21	326.40
hk48	209.42	220.89	254.38	268.85	18897	367.46	432.08	401.30	438.27	340.59
eil51	235.06	253.80	268.13	268.15	na	446.77	480.20	475.60	507.60	434.10
st70	1632.40	1781.27	1786.24	1832.80	na	3006.20	3764.80	3704.90	3849.10	2784.6
eil76	2158.20	2227.50	2336.89	2534.86	na	4472.53	5524.50	4886.90	5010.40	3535.70
pr76	2278.61	2299.47	2335.10	3005.55	na	5281.70	5956.89	5864.70	5997.30	3987.34
Average	624.74	649.52	678.93	765.52	8765.4	1296.20	1545.99	1469.15	1521.66	1075.80

Table 5.5. CPU times (in seconds) with barrier option for asymmetric instances

Instance	NEW1	NEW2	NEW3	NEW4	NEW5	GP1	GP2	GP3	GP4	CLAUS
ftv33	25.96	24.28	49.71	49.78	2148.3	49.32	70.17	77.05	79.47	72.08
ftv35	39.99	37.24	72.14	69.01	5462.5	81.24	90.01	87.85	108.91	75.79
ftv38	95.92	94.37	100.7	104.34	7651.4	121.35	138.88	156.12	214.5	108.1
ftv44	147.48	135.21	176.56	186.9	17895	291.8	306.7	259.5	317.35	221.57
ftv47	196.3	208.1	252.2	256.7	35171	397.34	427.44	503.07	662.3	335.25
ft53	403.91	428.3	560.05	609.09	na	711.3	756.54	818.29	849.79	688.2
ftv55	469.15	760.75	941.15	943.5	na	1105.3	1534.5	961.4	963.04	801.64
ftv64	1093.53	1137.53	1791.67	1664.72	na	2354.54	2825.2	2741.08	3292.9	1806.35
ft70	2326.09	2460.04	2615.37	2350.93	na	4824.52	4969.22	4202.5	4220.8	2498.5
ftv70	2473.05	2730.65	3407.31	3431.51	na	3895.7	5679.59	5238.9	5340.4	2752.76
kroA124p	33568	40368.4	46627.4	53689.6	na	39984.5	44876.4	51248.2	56368.4	48534.3
Average	3712.7	4398.62	5144.92	5759.64	13683.6	4892.25	5606.79	6026.72	6583.44	5263.14

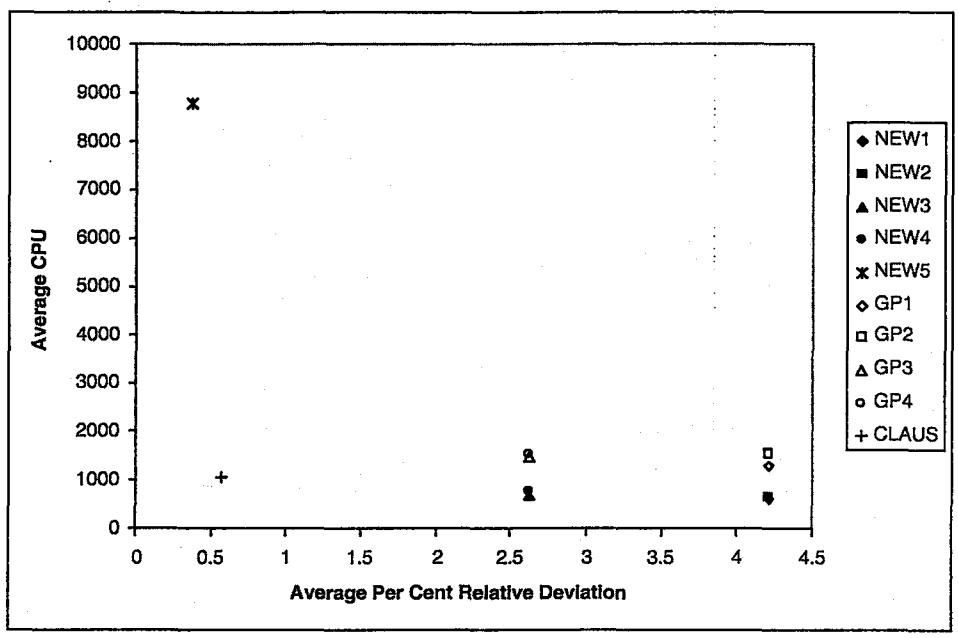

Figure 5.6. The performance of $O(n^3)$ formulations on symmetric instances

Recall that we have disregarded subtour elimination constraints for vertex 1 upto now while comparing the members of the NEW formulation family, even they do not cause an infeasibility as it is the case with the MTZ formulation and its extensions such as the DL, SD and GP formulations. The reason is to provide a platform for a fair comparison. The addition of these constraints increase their strengths since the related polyhedra become smaller.

Figure 5.6 and Figure 5.7 illustrate the performances of $O(n^3)$ constrained formulations for symmetric and asymmetric instances. In both of the figures CLAUS formulation appears to be the best since it has the lowest average CPU time requirement (the highest speed) with the lowest average per cent relative deviation (the highest average accuracy). Although the highest accuracy is obtained with NEW5, it has the highest average CPU time requirement. For example, for symmetric instances the average per cent deviation of the NEW5 is 0.363 and this value is 0.564 for the CLAUS. However, the CPU time requirements of the NEW5 and CLAUS are respectively, 8765.4 and 1075.8. For asymmetric instances, the accuracy gap between the NEW5 and CLAUS becomes clearer. However, the CLAUS' CPU time requirement is much less than the one of NEW5. Hence, NEW5 may not be preferable to the CLAUS in practice, unless

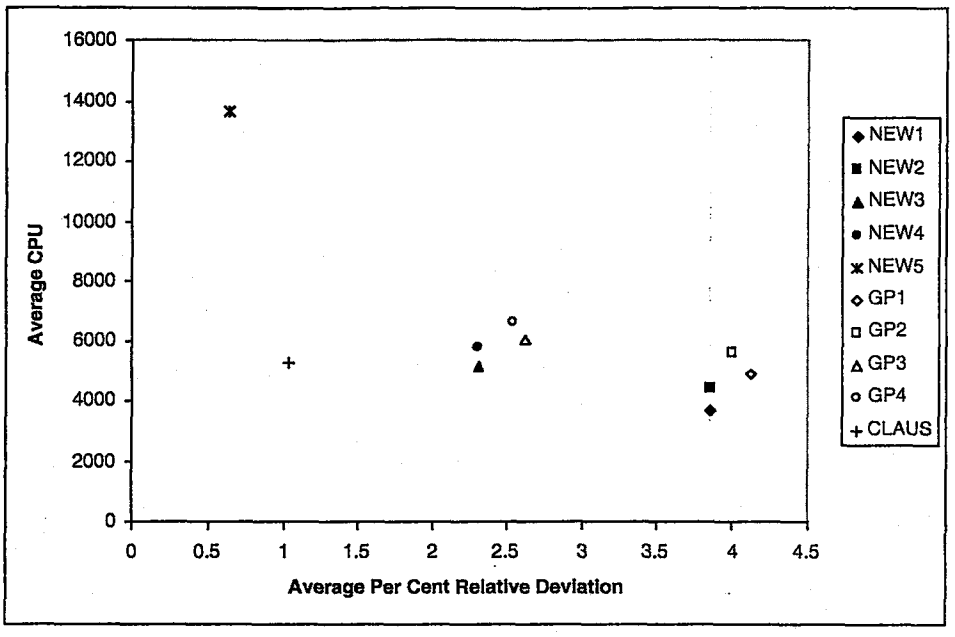

Figure 5.7. The performance of $O(n^3)$ formulations on asymmetric instances

the accuracy becomes much more important than the efficiency. The GP1, GP2, NEW1 and NEW2 formulations, and the GP2,GP4, NEW3 and NEW4 formulations are separately grouped close to each other in both of Figure 5.6 and Figure 5.7. For both of the figures, we can observe that the GP2, GP3, NEW3 and NEW4 formulations are closer to the origin $(0, 0)$ than the GP1, GP2, NEW1 and NEW2 formulations. That is to say, the performance of the former group of formulations (the GP2, GP4, NEW3 and NEW4) is better than the latter group of formulations (the GP1, GP2, NEW1 and NEW2). In both of the figures, GP1 and GP2 seem to be the worst formulations. They have the worst accuracy with relatively higher CPU time requirements comparing to the ones of the NEW1 and NEW2. Similarly, it is possible to observe that the NEW3 and NEW4 formulations are better than GP3 and GP4. In other words, comparing the NEW formulation family with the GP formulation family it is possible to say the performance of the NEW formulation family is better than the one of the GP formulation family. This observation becomes clearer on asymmetric instances.

5.5.2. Experiments with More Compact Formulations

Recall that all of the formulations considered on the experiments of this section have $O(n^2)$ constraints, except the NEW6 formulation. It has $O(n^3)$ subtour elimination constraints.

Table 5.6 and Table 5.7 present relative per cent deviations respectively for symmetric and asymmetric instances. Table 5.8 and Table 5.9 include CPU times spent for computing these bounds. We have used barrier procedure of Cplex ver. 7.0. In all of the tables the first column include the instances. The second column is for the MTZ, the third column is for the DL, the fourth column is for the SD and, the fifth and sixth columns are for NEW6 and NEW7 respectively. The average relative per cent deviations of the SD formulation are 3.41 for symmetric instances and 4.28 for asymmetric instances. These are the lowest average per cent relative deviations of Table 5.6 and Table 5.7. On the other hand, the average CPU time requirements of the SD formulation are 4.09 for symmetric instances and 5.66 for asymmetric instances. These are the lowest average CPU time requirements considering Table 5.4, Table 5.5, Table 5.8 and Table 5.9. Therefore, among all the formulations considered in this chapter, the SD has the highest efficiency. However, the accuracy of SD is not better than the ones of the GP3, GP4, NEW3, NEW4, NEW5, and CLAUS formulations, for symmetric instances. Moreover, the accuracy of SD is not better than any $O(n^3)$ constrained formulations for asymmetric instances.

Figure 5.8 and Figure 5.9 illustrate the performances of the more compact formulations. The performance of the SD formulation is the best which means that the SD formulation give the lowest average per cent relative deviation with the lowest average CPU time. On the other hand the performance of the NEW6 formulation is the worst. This is not surprising since NEW6 does not include assignment constraints. The NEW7 formulation is the second best. Its performance is very close to the one of SD. Recall that, both of NEW7 and SD formulations have n^2 variables. Then comes the DL and MTZ formulations. The performance of the DL formulation is better than the MTZ formulation since it is a lifted version of the MTZ formulation.

Table 5.6. Relative per cent deviations for symmetric instances

Instances	MTZ	DL	SD	NEW6	NEW7
gr21	9.59	0.00	0.00	37.90	0.00
gr24	16.19	3.73	3.20	30.90	3.73
bayg29	10.20	3.98	3.69	18.39	3.98
bays29	12.14	3.76	3.59	23.66	3.76
att48	20.13	5.52	5.25	40.42	5.52
gr48	17.56	5.49	5.17	33.02	5.49
hk48	13.50	2.30	1.76	26.67	2.30
eil51	11.16	2.23	2.13	13.62	2.23
st70	21.92	7.70	6.26	26.52	6.81
eil76	9.07	0.00	0.00	17.84	0.00
pr76	28.20	8.47	6.49	31.34	8.47
AVERAGE	15.42	3.93	3.41	27.30	3.84

Table 5.7. Relative per cent deviations for asymmetric instances

Instances	MTZ	DL	SD	NEW6	NEW7
ftv33	7.65	5.35	4.78	33.44	5.35
ftv35	6.12	4.04	3.90	35.95	4.04
ftv38	5.87	3.45	3.26	34.54	3.48
ftv44	5.56	2.43	2.43	27.12	2.43
ftv47	6.77	2.83	2.75	30.13	2.84
ft53	14.04	12.93	11.39	51.41	12.94
ftv55	10.56	6.05	5.89	27.45	6.05
ftv64	6.32	4.24	4.00	30.83	4.24
ftv70	9.26	4.69	4.64	28.21	4.69
ft70	1.77	0.88	0.80	17.89	0.90
kroA124p	6.13	3.46	3.23	13.96	3.46
AVERAGE	7.28	4.58	4.28	30.08	4.58

Table 5.8. CPU times (in seconds) with barrier option for symmetric instances

Instances	MTZ	DL	SD	NEW6	NEW7
gr21	0.45	0.53	2.08	1.43	0.35
gr24	0.84	0.92	4.80	4.68	0.36
bayg29	2.10	2.67	0.61	12.86	0.78
bays29	1.98	2.12	0.57	9.75	0.84
att48	49.84	58.60	0.59	191.74	6.00
gr48	45.60	53.23	2.24	198.56	5.86
hk48	54.13	62.37	2.39	193.79	6.48
eil51	64.40	68.20	3.48	62.74	6.86
st70	410.59	392.90	8.76	832.84	7.66
eil76	667.05	753.86	10.03	366.60	14.57
pr76	924.04	757.62	9.42	1173.04	15.57
AVERAGE	201.91	195.73	4.09	277.09	5.94

Table 5.9. CPU times (in seconds) with barrier option for asymmetric instances

Instances	MTZ	DL	SD	NEW6	NEW7
ftv33	5.34	6.39	1.15	20.21	17.2
ftv35	6.13	8.64	1.08	28.72	7.6
ftv38	10.79	17.20	1.59	42.20	15.0
ftv44	28.05	36.02	1.79	133.29	29.7
ftv47	53.10	64.50	2.30	182.57	49.4
ft53	93.70	111.04	3.11	224.39	1.7
ftv55	133.72	143.50	4.47	380.87	1.8
ftv64	246.38	334.07	5.63	884.65	2.8
ft70	411.50	590.80	6.54	1161.31	3.5
ftv70	407.70	592.30	10.43	1448.31	3.4
kroA124p	139.64	7.54	24.20	91.58	11.2
AVERAGE	139.64	173.82	5.66	418.01	13.03

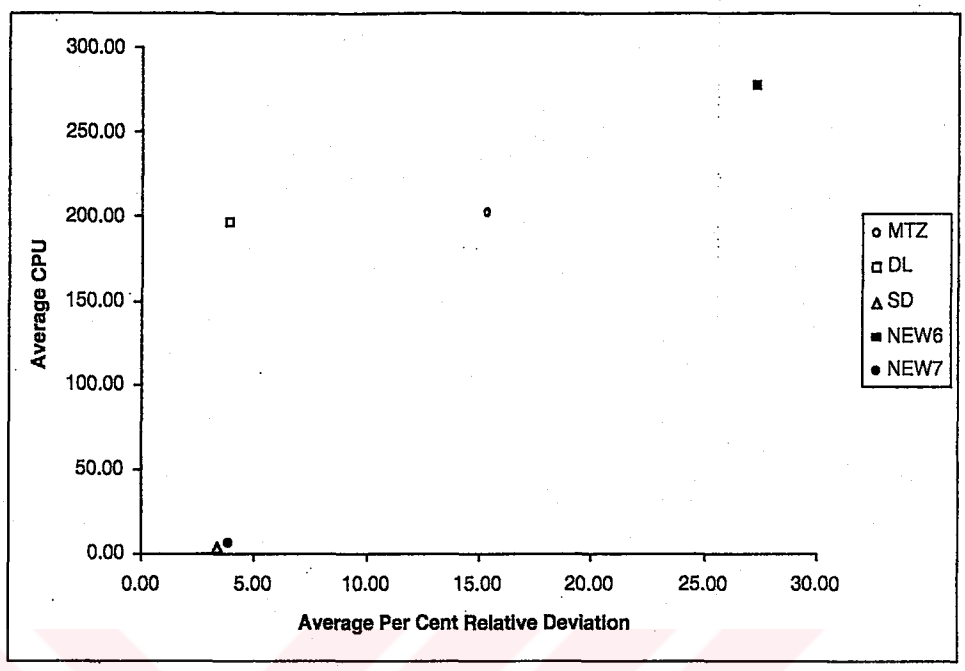

Figure 5.8. The performance of more compact formulations on symmetric instances

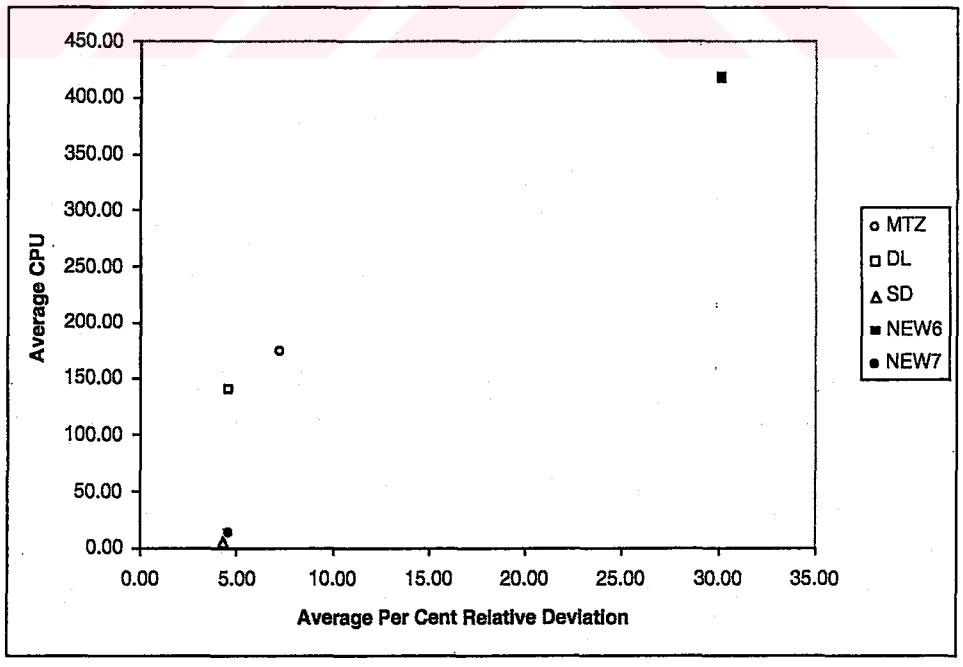

Figure 5.9. The performance of more compact formulations on asymmetric instances

6. A NEW ENHANCEMENT OF CLARKE AND WRIGHT'S SAVINGS HEURISTIC

6.1. Introduction

The CVRP is a difficult combinatorial optimization problem and known to be NP_hard [10]. The largest problem which has been solved to optimality contains 50 customers [63]. As a result of the inadequacy of exact methods, heuristics are widely used in practice. CVRP heuristics can be classified as classical heuristics and metaheuristics. Most of the works realized during the 30 – year period starting from 1960 are within the first family. Starting from 1990 the interest has been shifted towards metaheuristics and mostly new members for this family have been produced. Classical heuristics consist of three classes: Tour construction heuristics, tour improvement heuristics and two – phase methods. Tour construction heuristics gradually build a feasible solution either by merging existing routes using a *saving criterion* or assigning vertices to vehicle routes using an *insertion cost.* However, tour improvement heuristics attempt to improve a feasible solution by means of edge and vertex exchanges within or between vehicle routes. Two – phase methods are the result of two approaches. Methods based on the first group combine customers into feasible clusters first and obtains a vehicle route for each cluster then. These are known as *cluster – first, route – second* heuristics. Heuristics based on the second approach construct first a travelling salesman tour for all customers, which is then divided into separate parts to form vehicle routes with total customer demand not exceeding vehicle capacity D. They are known as *route – first, cluster – second methods.* For complete studies on classical CVRP heuristics see for example the recent works by Laporte *et al.* [64] and Laporte and Semet [65].

Metaheuristics intensively use neighborhood search methods to explore solution space without necessarily improving the objective function and sometimes allowing infeasible moves. Simulating Annealing (SA) [66–68] and Tabu Search (TS) [69,70], are

the most widely used metaheuristics to solve the CVRP. Both methods start with an initial solution and move each step from the current solution to a new one selected from its neighborhood, not necessarily improving the objective function. Other metaheuristics include Genetic Algorithms (GA) [71, 72], Ant Systems (AS) [73, 74] and Neural Networks (NN) [75–77]. GA examines at each step a new set of possibly infeasible solutions. The set of new solutions is the new population; it is derived from the previous one by combining its best member and discarding the worst. AS constructs possibly more than one new solutions at each step by using information collected on the previous solutions. NN are mainly adaptive learning processes that continuously update some weights until an acceptable, namely feasible or "close" to feasible solution is reached. The rule that governs the dynamics of the learning process depends on the implementation. Compared with TS, GA and AS no information on the previous solution is collected. More details on the application of metaheuristics on the CVRP can be found in two recent surveys by Laporte *et al.* [64] and Gendreau, Laporte and Potvin [78].

Classical heuristics are unsophisticated and perform a limited exploration of the search space compared with metaheuristics. However, they are simple, which makes them easy to understand and easy to implement, and produce fairly good solutions very fast. Some of them are flexible and can be extended easily to handle many variants of the CVRP. The performance of the metaheuristics is usually much higher than the ones of classical heuristics, but they require much more computational effort to have their parameters finely tuned – up. The large set of parameters they have, increase their flexibility; but they also make them context dependent and difficult to extend to other situations.

In their very recent work Cordeau *et al.* [79] introduce *accuracy*, *speed*, *simplicity* and *flexibility* as what they believe the most important attributes of a good heuristic and compare well – known classical heuristics and best available metaheuristics according to these four criteria. They report that none of the classical heuristics is as accurate and flexible as any one of the metaheuristics; but the CW heuristic is very fast and simple to implement, which probably explains its popularity. As a consequence,

how one can improve the accuracy of CW without harming its speed and simplicity very much becomes an interesting question. A quick intuitive answer could be the consideration of additional information in its savings criterion.

All of the savings method proposed for the CVRP use only distance information. However, the work by Vigo on the Asymmetric Capacitated Vehicle Routing Problem (ACVRP) [80], and the works by Salhi and Nagy [81] and Wade and Salhi [82] on the single and multi–depot CVRP with backhauling have shown that the use of vehicle overloads, which is implicitly related to customer demands, in addition to distances in the traditional insertion costs can improve the solution quality. Motivated by their results we propose a new parallel savings heuristic which combines distances and customer demands in its saving criterion.

6.2. Clarke and Wright's Savings Heuristic and Its Enhancements

Clarke and Wright's heuristic is not only one of the earliest methods proposed for the solution of the CVRP, but also probably the most widely used one in commercial routing packages. Initially every customer is visited by a separate vehicle. This is not clearly a feasible solution since a fleet of n vehicles (one vehicle per customer) is required. Subsequently, routes are combined repeatedly by considering the *saving* in the routing cost, which is obtained by using one vehicle instead of two for the same set of customers. Then the savings obtained by merging routes $(0, ..., i, 0)$ and $(0, j, ..., 0)$ into the route $(0, ..., i, j, ..., 0)$ is

$$\begin{aligned} s_{ij} &= (c_{0i} + c_{i0} + c_{0j} + c_{j0}) - (c_{0i} + c_{ij} - c_{j0}) \\ &= c_{i0} + c_{0j} - c_{ij} \end{aligned} \tag{6.1}$$

In short, at every iteration the feasible combination of two routes that leads to the largest saving in the routing cost is performed. The heuristic stops when a feasible merge of routes that leads to a saving is no longer possible. The *best feasible merge* version is also known as the parallel version of CW. In the sequential version the first

saving s_{ki} or s_{jl} that can feasibly be used to extend the current route $(0, i, \ldots, j, 0)$ by merging with another route containing arc or edge $(k, 0)$ or containing arc or edge $(0, l)$, is determined first. Then merge operation is implemented and repeated with the current route. The procedure stops when no feasible merge is possible. In their recent work on the CVRP heuristics Laporte and Semet [65] report that the parallel version dominates the sequential version, based on their experiments on the symmetric instances of Christofides *et al.* [83].

Saving defined as in formula (6.1) basically measures the gain when two customers are visited by the same vehicle instead of being visited separately by two vehicles. Savings become higher when the distance between customers i and j is smaller relative to their distance to the depot. As a consequence the original CW tends to produce good routes at the beginning. Gaskell [84] and Yellow [85] addressed this weakness in their early works and proposed the following parameterized saving expression:

$$s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}. \tag{6.2}$$

Here λ, which can only take positive values, is called *route shape* parameter. It prevents the formation of circumferenced routes which tend to be created by the original CW. As it increases, greater emphasis is placed on the distance between customer i and j rather than their position relative to the depot. The search for the best route structure realized by changing λ provides better heuristic solutions with an additional search effort for the best parameter value.

Another way to improve the performance of the CW algorithm is to extend the parametric saving expression (6.2) to consider more information about the spatial distribution of the customers. One approach can be the use of asymmetry between customers i and j with respect to their distances to the depot. This was Paessens motivation who introduced one new term and parameter to saving expression (6.2) in order to obtain the following one [86]:

$$s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu \left| c_{0i} - c_{j0} \right|. \tag{6.3}$$

This new enhancement of the CW can be run with different values of the parameters λ and μ to obtain different solutions. The result is a considerable increase in the solution quality, but also in the computational effort. However, Paessens also proposes an efficient approach to compute maximum savings [86]. Hence Paessens work contributes to both the *accuracy* and the *speed* of the savings algorithms.

Two other enhancements of the CW are due to Golden *et al.* [87] and Nelson *et al.* [88]. They mainly contribute to its speed. Computing the maximum saving value is computationally the most expensive part of the algorithm. Golden *et al.* consider Gaskell–Yellow savings criterion (6.2) and use a limited sort, instead of a full sort of the savings, which they implement by means of heap data structure. Their work is based on the idea of considering a subset of all possible savings in order to decrease computing time and memory requirements of the algorithm. Nelson *et al.*'s basic idea is quiet similar. However, they use more complex abstract data structures, such as hash functions implemented by using several smaller heaps instead of one large heap, in computing maximum savings calculated according to CW's original formula (6.1). Recalling the level nowadays computer technology has been reached it is possible to say that these enhancements may only be useful for very large instances.

The storage of the complete savings set allows the consideration of different selection strategies. One such example is Daskin's work [89]. At each iteration the original CW myopically chooses the route pairs. That is why several researchers have tried to introduce route shape parameter into the savings formula. Taking into consideration this observation, Daskin has come up with the randomized saving approach. He proposes to run the standard CW but at the route merging step, the k^{th} best saving is selected from the top of the saving list where k is a random variable between 1 and the *depth*, which is set to a value at the beginning. The randomized algorithm is then repeated for a *number of times*, which is another parameter fixed at the beginning with the *depth*, and the best solution is recorded.

6.3. New Enhancement

In this section we introduce a new enhancement of the original CW. While running the original CW and its enhancements, especially towards the end, the merge of routes with equal or very close savings occur often. Then it may be more interesting to consider also customer demands while calculating savings. In fact, the VRP consists of two problems: The Multiple Travelling Salesman problem (m–TSP) and the Bin Packing Problem (BPP). Hence, a saving expression which somehow combines the solution strategies proposed for them can increase the chance of obtaining higher improvements. For this purpose we have adopted the well known *first fit decrease* idea of Martello and Toth [90], which was originally used for the BPP: *put first larger items.* In short we propose the following new savings criterion:

$$s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu \left| c_{0i} - c_{j0} \right| + \nu \frac{d_i + d_j}{\overline{d}}. \tag{6.4}$$

Here d_i is the demand of customer i, $\overline{d}$ is the average demand used to normalize, c_{ij} is the distance between customers i and j, c_{i0} and c_{0j} are respectively distances between customers i and j and the depot, and ν is the new parameter. Since it also includes enhancements due to Gaskell [84], Yellow [85] and Paessens [86], saving expression (6.4) is more general. We consider demands normalized with the average demand $\overline{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$ because we prefer to include the relative importance of the customers according to their demands on a vehicle's capacity.

The situation given in Figure 6.1 illustrates the effect of customer demands, namely the third term of the new saving expression (6.4). There are eight customer points located equidistantly from each others on a circle whose center is the depot. Namely, $c_{i,i+1} = c_{i+1,i+2}$ for $i = 1, ..., 6$, and $c_{0,i} = c_{0,i+1}$ for $i = 1, ..., 7$. The given numbers in this figure present customer demands. Assume also that the fleet consists of identical vehicles with capacity 100 units. Observe that when customer demands are ignored, namely Paessens' saving expression (6.3) is used, any two adjacent customer

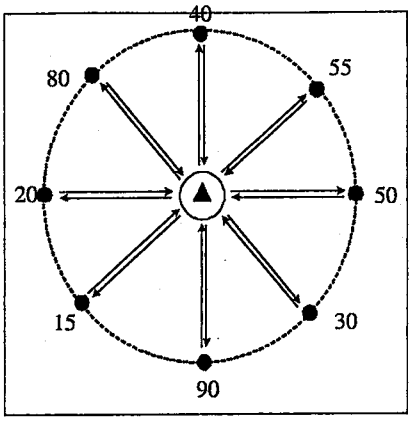

Figure 6.1. Effect of customer demand in merging

points can be combined unless they violate capacity restriction. However, this is different for the situation when demands are also considered: For any positive ν, customer points with demands 20 and 80 will have priority in our example.

The new method is also a parallel heuristic and starts with n vehicles each visiting one customer. Then gradually combines routes $(0, i, \ldots, 0)$ and $(0, \ldots, j, 0)$ into $(0, \ldots, i, j, \ldots, 0)$ if the saving criterion (6.4) has the largest value between all possible such route merges.

Another strategy for introducing demand information into the saving criterion may be the consideration of the remaining vehicle capacities. Very briefly, one might be interested in the saving expression

$$s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu \left| c_{0i} - c_{j0} \right| + \gamma \frac{(D - d_i) + (D - d_j)}{D - \bar{d}}. \tag{6.5}$$

Here the new parameter γ weights the relative importance given to customers with smaller demands, which results in higher remaining capacities. In other words the larger is γ, the smaller are the items put first, this time. As a result of extensive computational study we can say that its performance is not better than the one with the saving criterion (6.4). Hence we have preferred not to report the related computational results here.

While proposing the new heuristic, our main goal is to increase the performance of the original CW as much as we can with minimal sacrifice in its simplicity and speed. As a result we have not included on purposely a sophisticated tuning phase, during which several edge/node interchanging strategies can be used in order to improve the final solution obtained by the new savings method. Computational results reported in the next section indicate that the new approach causes considerable improvements in the accuracy of the CW without decreasing its speed and simplicity.

6.4. Computational Results

In order to demonstrate the improvement obtained by considering customer demands we conducted experiments with the original CW, its enhancements due to Gaskell [84], Yellow [85] and Paessens [86], and the new heuristic with savings criterion (6.4) on the same test bed. In our implementations we use Method 4 by Nelson *et al.* [88], which is efficient and not very complicated to implement. In this method savings are stored in a heap represented as a $n \times (n+1)/2$ dimensional array. They also use an additional linear array of size $n \times (n+1)/2$ to link saving s_{ij} to customer pair (i, j) and vice versa. At each step, the root of the heap, which contains the largest saving value s_{ij}, is removed from the heap and the routes passing through customers i and j are merged. Saving values which are associated to customers on these two merged routes are also removed from the heap. Besides, if it is not possible to merge the routes due to the capacity restrictions, then the corresponding value is also removed from the heap. Nelson *et al.* [88] propose four more methods to make CW more efficient. Method 1 – Method 3 are less efficient than Method 4. Method 5 is more efficient, but unfortunately the used data structure requires certain maturity in computer programming. At sum, Method 4 is very efficient and simpler to programme. In an earlier version of this work [91] we used a matrix to store the saving values. This is the most simple data structure to implement and require only very basic programming skills. In other words, we search the whole savings matrix starting from the very first cell and keep the best saving with the smallest indices i and j. Then, we try to merge the routes which contain customers i and j. We join them if the total demand of these two routes does not exceed vehicle capacity. In the other case, when the capacity is

exceeded, the saving value s_{ij} is set to 0. We also update the savings matrix after the merge operation is performed: The savings belonging to the pairs of customers who are on the new route are also set to 0. The algorithm terminates when there is no more positive saving in the savings matrix. However, the search effort spent on the determination of largest saving value was prohibitive; we ended up with CPU times approximately fifteen times larger than the ones reported here. All of our codes are written in C^{++} programming language.

In sophisticated implementations of savings heuristics, a tour improvement step is repeated after every merge operation to decrease the length of the new route. This is usually realized by using local improvement heuristics suggested originally for the TSP. As for example Daskin uses Or–opt [32], which is then followed by 2–opt after every merge and reports considerable improvement in the accuracy of his randomized savings heuristic [89]. This is also Laporte and Semet's motivation for using 3–opt with the best improvement strategy after every merge in their reputed parallel CW implementation [65]. For only the verification of our codes we compare our implementation of the CW with theirs on the seven symmetric instances of Christofides *et al.* [83]; they are the ones with capacity restrictions. The results are summarized in Table 6.1. The entries of the first column are the instances. The letter C denotes that there is a restriction on the vehicle capacity. Following digits denote the number of customers; the depot is not counted. Second column is adopted directly from Laporte *et al.*'s work [64]. Notice that we have two implementations. The third and fourth columns include final total route lengths computed by our plain and sophisticated implementations. Our sophisticated implementation has also local improvement steps. Once two routes are merged we first use Or–opt with strings of up to four vertices; sequences of four, three, two and one consecutive vertices (customers) are relocated in the new route so that it becomes shorter. The string whose relocation results in the best improvement (largest decrease of all possible relocations) is relocated. Then 4–opt [31] is utilized on the same route: Four edges are replaced with four others so that the length of the new route decreases. We again do the interchange which results in the best improvement. However, our plain implementation does not include any local improvement procedure.

Table 6.1. Total routing costs obtained by different CW implementations

Instance	Laporte–Semet	Altınel-Öncan (Plain)	Altınel–Öncan
C50	578.56	584.6	572.29
C75	888.04	907.3	888.04
C100a	878.70	1035.0	880.62
C150	1128.24	1140.4	1128.24
C199	1386.84	1395.7	1370.11
C120	1048.53	1068.1	1046.5
C100b	824.42	833.5	824.42

As it can be observed, our plain implementation finds larger values. This should be expected since it does not include any local improvement step. However, our sophisticated implementation results in equal or slightly lower values except instance $C100a$: The value our implementation computes is slightly higher. This is also expected since we spend more effort on local improvement than Laporte and Semet do. In short it seems we can rely on our CW implementation and use it for benchmarking. While proposing the new heuristic, our main goal is to increase the performance of the original CW as much as we can with minimal sacrifice in its simplicity and speed. As a result we prefer to use mainly our plain implementation in our computational experiments. We also believe this exposes the contribution introduced by demand information better.

Our test bed does not only consist of the instances by Christofides *et al.* [83], but also the ones by Christofides and Eilon [92] and Augerat *et al.* [93]. They can be downloaded from [94, 95]. The results reported in Table 6.2 are obtained with local improvement steps (Or–opt followed by 4–opt) by using the mentioned symmetric and capacitated instances of Christofides *et al.*'s. However, the results reported in Table 6.3 are obtained without local improvement steps. The first column of the tableau shows the instances. The second and third columns are obtained by using our CW implementation. The fourth and sixth columns include values calculated by our implementation of Gaskell–Yellow CW enhancement (CW1). The numbers in each cell are respectively route–shape parameter λ, the relative per cent deviations from the

best known values and total CPU times; they are the best of the results of 20 runs each of which is realized with a value of λ between 0.1 and 2. Initially λ is set to 0.1 and incremented by 0.1 every time. The entries of the sixth column are the total of CPU times in seconds of 20 runs. This format is repeated in columns 7 – 9 for our implementation of Paessens' enhancement (CW2) and in columns 10 – 12 for the new heuristic (CW3) with (λ, μ) pairs and (λ, μ, ν) triplets respectively. In the last column we present the best known values. The search effort becomes higher with the increase in the number of parameters. For our Paessens implementation λ changes within $[0.1, 2]$ with an increment of 0.1, while μ is taking values starting from 0 up to 2 with an increment of 0.1. In other words, the numbers in columns 7 and 8 are the best and the numbers in column 9 are the averages of $20 \times 21 = 420$ values. The search effort for the new heuristic is even higher because of the third parameter ν; it is assigned values starting from 0 up to 2 with an increment of 0.1. As for λ and μ, the search intervals are kept the same, namely $[0.1, 2]$ for λ and $[0, 2]$ for μ with an increment of 0.1. Hence the number in columns 10 – 11 are respectively the best of $20 \times 21 \times 21 = 8820$ values. As for example, for instance `C50`, it takes CW3 95.05 seconds (1.5 min.), to obtain a relative per cent deviation of 3.46 and parameters $(\lambda, \mu, \nu) = (0.6; 1.5; 0.4)$ as the best of 8820 values. Note that same local improvement strategy is activated after every merge in all implementations. Namely, first Or–opt then 4–opt and best improvement are executed. For the same instance, namely `C50`, when the local improvement strategy is not activated, it takes CW3 24.93 seconds (0.41 min.), to obtain a relative per cent deviation of 5.90 and parameters $(\lambda, \mu, \nu) = (1.4; 0.9; 0.3)$ as the best of 8820 values.

Tables 6.4 – 6.11 are prepared with the same purpose for different instances. For Table 6.4 and Table 6.5 the instances are chosen from Christofides and Eilon's [92] test bed, while the test bed from Augerat *et al.* [93] is being used for Tables 6.6 – 6.11. Table 6.4, Table 6.6, Table 6.8 and Table 6.10 present results with local improvement steps,namely first Or–opt, then 4–opt this time. However, Table 6.5, Table 6.7, Table 6.9 and Table 6.11 present results without local improvement steps. At the first look at the last rows of the tables we can say that local improvement steps have resulted in slight improvements of the relative deviations with a considerable increase in the CPU times.

In all these tables last row includes column averages of relative per cent deviations and total CPU seconds. Averaged relative per cent deviations give clues about the improvement in the accuracy. For Augerat *et al.*'s and Christofides and Eilon's test set we use the best known results reported at [94] in the calculations. The best known results for Christofides *et al.*'s instances are obtained from the work of Rochat and Taillard [70].

The reported values of the parameters inform us implicitly about the contribution of their corresponding term to the savings. As for example when $\lambda = 1$ CW1 becomes CW. Similarly for $\lambda = 1$, $\mu = \nu = 0$ CW3's behavior is equivalent to CW's. Therefore, whether the consideration of customer demand is a good idea or not can be determined how often ν is different than zero. For 137 out of 174 values reported in column 11 of Table 6.2 and Table 6.11 (79 per cent of the reported best values) ν is different than zero, which also points the contribution of CW3.

Table 6.2. Best relative per cent deviations obtained with local improvements on Christofides *et al.*'s test set

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
C50	9.09	0.00	0.9	9.09	0.22	0.8;1.2	4.88	4.70	0.6;1.5;0.4	3.46	95.05	524.6
C75	6.32	0.02	1.1	3.85	0.25	1.2;0.5	3.54	5.56	1.2;0.2;0.8	2.27	108.22	835.3
C100	6.59	0.05	1.6	3.55	1.06	1.6;0.0	3.55	22.87	1.0;1.6;0.2	2.14	467.34	826.1
C150	9.71	0.09	1.3	7.59	1.84	1.7;0.7	5.42	40.33	1.8;1.3;1.0	4.45	869.89	1028.4
C199	6.09	0.14	1.9	5.70	2.75	1.9;0.0	5.70	58.31	1.6;0.8;1.5	4.41	1394.30	1291.5
C120	0.42	0.14	1.0	0.38	2.73	1.0;0.0	0.38	57.42	1.0;0.0;0.0	0.38	1276.86	1042.1
C100b	0.59	0.03	1.1	0.17	0.62	1.1;0.0	0.17	14.23	1.1;0.0;0.0	0.17	322.03	819.6
Average	5.5	0.1		4.33	1.4		3.38	29.1		2.56	647.7	909.7

Table 6.3. Best relative per cent deviations obtained without local improvements on Christofides *et al.*'s test set

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
C50	11.44	0.02	1.3	11.22	0.06	1.0;0.3	8.42	1.17	1.4;0.9;0.3	5.90	24.93	524.6
C75	8.64	0.02	1.1	3.72	0.13	1.0;0.1	3.72	3.17	1.2;0.2;0.8	2.99	63.59	835.3
C100	7.35	0.02	1.3	6.44	0.22	1.3;0.8	4.95	5.50	1.4;0.3;0.3	4.27	117.94	826.1
C150	10.89	0.03	1.1	8.48	0.63	1.6;0.4	6.91	13.73	1.6;0.3;0.4	6.25	283.03	1028.4
C199	8.08	0.06	1.1	6.95	1.34	1.4;0.2	6.09	28.59	1.3;0.0;1.1	5.29	627.70	1291.5
C120	2.50	0.02	1.0	2.50	0.41	1.3;0.3	2.34	8.45	1.1;0.1;0.3	1.51	182.96	1042.1
C100b	1.70	0.02	1.1	1.30	0.27	1.3;0.3	0.97	5.63	1.3;0.3;0.6	0.63	119.72	819.6
Average	7.23	0.02		5.80	0.44		4.77	9.46		3.83	202.8	909.7

Table 6.4. Best relative per cent deviations obtained with local improvements on Christofides and Eilons' test set

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
E-n22-k4	3.67	0.01	0.8	3.34	0.02	1.4;0.6	0.07	0.31	1.2;0.2;1.1	0.07	6.62	375.0
E-n23-k3	-0.08	0.00	0.8	-0.08	0.14	0.8;0.0	-0.08	2.56	0.8;0.0;0.0	-0.08	52.83	569.0
E-n30-k4	1.02	0.00	1.6	0.75	0.17	1.9;1.0	0.40	3.23	1.0;0.0;1.1	0.40	60.25	503.0
E-n33-k4	0.93	0.00	1.0	0.93	0.14	0.9;1.7	0.65	2.65	0.9;1.4;1.1	0.65	54.33	835.0
E-n76-k14	5.08	0.00	1.1	4.40	0.16	1.1;0.1	3.61	3.02	1.3;0.0;0.4	2.85	65.71	1021.0
E-n76-k8	6.89	0.02	1.1	6.68	0.39	0.9;0.5	4.06	7.34	1.3;0.1;1.3	3.67	145.38	735.0
E-n76-k7	5.95	0.02	0.7	5.95	0.31	0.7;1.3	3.42	5.94	0.7;1.3;0.0	3.42	121.57	682.0
E-n101-k14	6.14	0.02	1.8	4.86	0.38	1.6;1.2	4.68	7.08	0.7;0.6;1.3	4.21	145.53	1071.0
Average	3.70	0.01		3.36	0.21		2.10	4.02		1.90	81.53	723.9

Table 6.5. Best relative per cent deviations obtained without local improvements on Christofides and Eilons' test set

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
E-n22-k4	3.67	0.00	0.8	3.34	0.00	1.5;0.6	0.07	0.22	1.0;0.9;1.6	0.07	3.43	375.0
E-n23-k3	9.15	0.00	1.6	2.78	0.00	1.8;0.5	0.71	0.22	1.7;0.5;0.9	0.71	4.94	569.0
E-n30-k4	6.25	0.00	1.6	1.97	0.02	1.3;0.3	0.73	0.38	1.3;0.3;0.0	0.73	7.14	503.0
E-n33-k4	0.97	0.00	1.0	0.97	0.02	1.0;0.0	0.97	0.47	1.0;0.0;0.0	0.97	9.14	835.0
E-n76-k14	3.29	0.00	1.1	3.28	0.16	1.1;0.1	3.07	3.30	1.3;0.0;1.0	2.35	64.17	1021.0
E-n76-k8	8.13	0.02	1.4	7.01	0.16	1.7;1.1	6.34	3.13	1.6;1.3;0.7	5.46	62.91	735.0
E-n76-k7	8.23	0.00	1.5	6.33	0.14	1.6;0.8	5.23	3.12	1.7;0.8;0.1	4.35	62.53	682.0
E-n101-k14	6.36	0.02	1.8	5.99	0.28	0.7;0.5	5.88	5.74	0.8;0.6;0.3	5.17	119.40	1071.0
Average	5.76	0.00		3.96	0.10		2.88	2.07		2.48	41.71	723.9

Table 6.6. Best relative per cent deviations obtained with local improvements on Augerat *et al.*'s test set A

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
A-n32-k5	5.95	0.00	0.8	5.95	0.05	0.3;0.7	4.36	1.22	0.0;0.4;0.6	4.36	26.28	784.0
A-n33-k5	7.69	0.00	1.1	4.59	0.05	1.8;1.9	4.17	1.11	1.2;0.8;0.4	4.09	21.64	661.0
A-n33-k6	4.58	0.00	0.4	3.86	0.05	0.7;0.4	1.86	1.02	1.0;1.2;1.0	1.86	17.27	742.0
A-n34-k5	4.17	0.02	0.8	1.89	0.05	0.7;0.1	1.89	1.56	1.2;0.6;0.3	1.85	22.12	778.0
A-n36-k5	3.69	0.00	0.9	0.97	0.08	0.7;0.1	0.97	1.75	0.8;0.0;0.1	0.97	36.58	799.0
A-n37-k5	3.95	0.00	0.8	3.95	0.11	1.0;0.7	3.70	2.33	1.1;1.5;0.3	3.69	46.08	669.0
A-n37-k6	3.70	0.00	0.9	2.54	0.03	1.0;0.1	2.07	1.36	1.5;0.6;0.8	1.62	26.43	949.0
A-n38-k5	4.91	0.00	0.3	3.32	0.06	0.2;0.9	1.46	1.52	0.0;0.4;0.0	1.46	29.02	730.0
A-n39-k5	9.64	0.00	1.3	5.78	0.08	2.4;1.0	3.29	1.91	1.1;1.6;1.2	2.17	38.87	822.0
A-n39-k6	3.12	0.00	1.0	3.12	0.08	2.9;1.9	2.04	1.94	0.0;3.0;1.8	1.46	39.01	831.0
A-n44-k7	3.26	0.00	0.9	3.27	0.09	1.9;0.9	1.93	2.14	0.8;1.4;1.6	1.93	41.64	937.0
A-n45-k6	6.62	0.00	1.7	3.57	0.09	1.1;0.1	1.38	2.25	1.1;0.1;0.0	1.38	44.41	944.0
A-n45-k7	4.55	0.00	0.7	4.45	0.06	2.0.1.0	1.28	1.78	2.0;1.8;0.4	1.22	35.58	1146.0
A-n46-k7	2.55	0.00	1.4	1.94	0.11	1.3;0.5	1.09	2.25	0.4;1.4;0.6	1.09	45.65	914.0

Table 6.6. Best relative per cent deviations obtained with local improvements on Augerat *et al.*'s test set A (continued)

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
A-n48-k7	2.81	0.00	0.7	2.81	0.13	0.7;0.0	2.81	2.71	1.8;0.6;0.2	2.59	56.34	1073.0
A-n53-k7	7.31	0.02	0.5	5.35	0.19	1.4;0.6	1.85	4.11	0.0;1.4.0.6	1.85	81.93	1010.0
A-n54-k7	1.83	0.00	1.0	1.83	0.16	1.9;0.9	0.65	3.64	0.0;2.0;1.0	0.65	68.31	1167.0
A-n55-k9	2.37	0.00	1.0	2.37	0.08	1.0;0.0	2.37	2.27	1.6;0.8;0.4	2.13	45.78	1073.0
A-n60-k9	2.48	0.00	1.2	0.75	0.14	1.2;0.0	0.75	3.17	0.0;1.2.0.0	0.71	64.57	1354.0
A-n61-k9	7.27	0.00	1.1	1.65	0.14	1.2;0.1	1.65	3.30	0.3.1.8;1.0	1.65	61.38	1034.0
A-n62-k8	4.70	0.00	1.0	4.70	0.22	1.1;0.2	4.33	5.44	1.9;1.4;0.2	4.33	101.06	1288.0
A-n63-k10	2.93	0.00	1.1	1.99	0.14	1.9;1.1	1.39	3.35	0.0;2.0;1.2	1.39	70.50	1314.0
A-n64-k9	5.72	0.00	1.2	4.26	0.23	1.4;0.5	2.03	5.31	0.0;1.0;0.6	2.03	99.90	1401.0
A-n63-k9	4.19	0.00	0.7	3.09	0.14	0.9;0.9	1.90	3.76	0.2;1.0;1.0	1.90	76.96	1616.0
A-n65-k9	5.79	0.02	1.0	5.79	0.17	1.0;0.2	4.19	3.74	0.0;1.0;0.2	2.26	75.69	1174.0
A-n69-k9	2.90	0.02	1.3	2.11	0.23	1.3;0.0	2.11	4.97	0.0;1.4;0.0	2.11	97.27	1159.0
A-n80-k10	5.14	0.02	0.7	3.71	0.31	3.0;2.6	2.20	8.49	0.0;3.0;1.6	2.20	171.62	1763.0
Average	4.59	0.01		3.32	0.12		2.16	2.90		1.99	57.11	1041.9

Table 6.7. Best relative per cent deviations obtained without local improvements on Augerat *et al.*'s test set A

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
A-n32-k5	7.61	0.00	1.2	6.79	0.02	0.8;0.6	5.70	0.39	0.8;0.6;0.0	5.70	8.50	784.0
A-n33-k5	7.72	0.00	1.1	4.59	0.03	1.2;0.8	2.83	0.45	1.2;0.8;0.0	2.83	9.69	661.0
A-n33-k6	4.62	0.00	1.2	1.92	0.02	1.9;1.1	0.65	0.47	1.2;0.0;1.0	0.16	9.33	742.0
A-n34-k5	1.93	0.02	0.8	1.93	0.03	0.7;0.1	1.93	0.45	0.6;0.3;1.2	1.93	10.01	778.0
A-n36-k5	3.69	0.00	0.9	0.97	0.02	0.9;0.0	0.97	0.66	0.8;0.0;0.1	0.97	10.82	799.0
A-n37-k5	5.80	0.02	1.3	5.43	0.03	0.9;0.9	4.38	0.62	1.5;0.3;1.1	3.80	11.80	669.0
A-n37-k6	2.91	0.02	1.0	2.91	0.02	1.0;0.1	2.85	0.64	1.1;0.1;0.3	2.69	12.04	949.0
A-n38-k5	5.22	0.00	1.0	5.22	0.05	1.6;0.1	4.12	0.64	1.4;0.3;0.2	3.58	12.51	730.0
A-n39-k5	9.73	0.02	1.4	3.56	0.05	1.1;0.1	3.19	0.72	1.1;0.1;0.0	3.19	13.35	822.0
A-n39-k6	3.86	0.00	0.9	3.35	0.05	0.8;0.2	2.23	0.64	0.8;0.2;0.0	2.23	13.22	831.0
A-n44-k7	4.17	0.00	0.9	4.17	0.05	1.9;0.8	3.40	0.92	1.6;0.4;1.8	2.39	18.21	937.0
A-n45-k6	6.62	0.02	1.1	3.77	0.05	1.2;0.2	2.31	0.94	1.0;0.0;1.4	1.38	18.37	944.0
A-n45-k7	4.71	0.00	0.9	4.71	0.03	2.0;1.0	2.01	1.00	1.8;0.2;1.8	1.84	18.97	1146.0
A-n46-k7	2.82	0.00	1.3	2.29	0.05	1.1;0.1	2.15	1.00	1.1;0.1;0.0	2.15	19.86	914.0

Table 6.7. Best relative per cent deviations obtained without local improvements on Augerat *et al.*'s test set A (continued)

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
A-n48-k7	3.71	0.02	1.2	3.02	0.05	1.6;0.7	2.91	1.13	1.6;0.7;0.0	2.91	22.11	1073.0
A-n53-k7	8.86	0.02	0.7	7.92	0.06	1.5;0.6	3.56	1.34	1.9;1.2;1.6	3.43	27.56	1010.0
A-n54-k7	2.93	0.00	1.0	2.93	0.06	1.7;0.9	2.11	1.44	1.1;0.1;0.9	0.86	28.62	1167.0
A-n55-k9	2.50	0.00	1.0	2.50	0.08	1.3;0.2	2.48	1.59	1.1;0.1;1.0	2.38	30.56	1073.0
A-n60-k9	5.01	0.00	1.1	3.65	0.09	1.9;0.8	2.33	1.84	1.4;0.0;0.9	1.64	37.88	1354.0
A-n61-k9	6.60	0.00	1.1	1.68	0.08	1.1;0.0	1.68	1.94	1.1;0.0;0.1	1.65	38.39	1034.0
A-n62-k8	5.03	0.00	1.0	5.03	0.09	1.2;0.2	4.90	1.98	1.0;0.0;0.2	4.65	40.50	1288.0
A-n63-k10	3.42	0.00	1.1	2.07	0.08	1.3;0.2	2.07	2.09	1.4;0.4;1.3	1.94	41.25	1314.0
A-n64-k9	6.13	0.00	1.4	4.40	0.11	1.4;0.5	2.99	2.15	1.4;0.6;0.1	2.50	42.53	1401.0
A-n63-k9	4.45	0.00	0.8	4.26	0.09	1.6;0.6	2.08	2.09	1.6;0.6;0.1	2.05	41.57	1616.0
A-n65-k9	5.57	0.02	0.9	4.78	0.09	0.9;0.3	4.42	2.12	0.9;0.1;0.3	2.39	43.26	1174.0
A-n69-k9	4.47	0.02	1.3	2.25	0.11	1.3;0.0	2.25	2.53	1.3;0.0;0.0	2.25	50.45	1159.0
A-n80-k10	5.56	0.02	0.8	3.82	0.17	1.8;0.6	3.26	3.53	1.5;0.7;1.6	2.95	71.03	1763.0
Average	5.02	0.01		3.70	0.06		2.81	1.31		2.46	26.01	1041.9

Table 6.8. Best relative per cent deviations obtained with local improvements on Augerat *et al.*'s test set B

	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
B-n31-k5	12.07	0.00	0.9	0.61	0.03	0.9;0.0	0.61	0.88	0.9;0.0;0.0	0.61	18.68	672.0
B-n34-k5	5.48	0.00	1.2	0.23	0.05	1.2;0.0	0.23	1.22	1.2;0.0;0.0	0.23	26.50	788.0
B-n35-k5	1.64	0.00	0.7	1.64	0.06	0.7;0.0	1.64	1.33	1.0;0.0;1.9	1.34	27.21	955.0
B-n38-k6	7.16	0.00	0.8	2.56	0.06	1;0.2.0	1.97	1.70	1.6;0.8;1.7	1.85	38.08	805.0
B-n39-k5	2.89	0.00	1.0	2.89	0.11	1.0;0.0	2.89	2.92	1.0;0.0;1.7	2.60	59.78	549.0
B-n41-k6	8.02	0.00	0.7	6.04	0.06	0.4;0.7	2.86	1.40	1.0;1.2;1.5	2.49	28.05	829.0
B-n43-k6	4.98	0.01	0.3	1.38	0.08	0.3;0.0	1.38	2.08	0.4;0.2;0.6	0.95	43.72	742.0
B-n44-k7	4.04	0.01	1.0	3.11	0.09	1.8;0.8	2.50	2.17	1.8.0.8;0.0	2.50	37.83	909.0
B-n45-k5	0.58	0.01	1.0	0.58	0.14	1.0;0.0	0.58	3.08	1.1;0.0;0.8	0.51	64.79	751.0
B-n45-k6	7.16	0.01	1.1	6.25	0.09	0.5;0.3	5.47	2.09	0.3;1.2;0.9	1.48	41.76	678.0
B-n50-k7	0.97	0.01	1.1	0.91	0.13	1.1;0.0	0.91	3.50	0.9;0.0;0.2	0.45	69.39	741.0
B-n50-k8	3.00	0.01	0.8	2.73	0.09	1.4;0.6	2.12	2.18	1.8;0.7;1.3	1.58	44.18	1312.0

Table 6.8. Best relative per cent deviations obtained with local improvements on Augerat *et al.*'s test set B (continued)

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
B-n51-k7	9.97	0.01	1.2	-0.80	0.11	1.2;0.0	-0.80	2.38	1.2;0.0;0.8	-0.99	44.83	1032.0
B-n52-k7	1.71	0.01	1.0	1.71	0.12	1.9;1.1	1.43	3.72	1.0;0.0;0.8	1.30	75.92	747.0
B-n56-k7	4.74	0.01	0.7	2.10	0.19	1.0;0.1	2.00	4.66	0.9;0.7;1.0	1.84	89.53	707.0
B-n57-k7	7.42	0.01	1.3	0.55	0.16	1.6.0.7	-0.43	3.77	1.4;0.3;1.9	-0.62	73.59	1153.0
B-n57-k9	8.71	0.01	0.9	0.83	0.13	0.9;0.0	3.96	2.90	0.9;0.0;0.0	3.96	57.53	1598.0
B-n63-k10	7.34	0.02	0.9	4.10	0.14	0.8;0.1	4.10	3.59	0.8;0.1;0.0	4.10	72.61	1496.0
B-n64-k9	6.56	0.01	1.0	6.56	0.12	1.5;1.0	6.39	3.83	1.2;0.7;1.9	5.77	73.77	861.0
B-n66-k9	7.09	0.02	0.5	4.89	0.16	1.4;0.4	3.28	4.44	1.4;1.0;1.0	1.97	88.78	1316.0
B-n67-k10	6.19	0.01	1.0	6.19	0.16	0.9;0.3	3.97	3.88	0.8;0.1;0.3	3.31	79.20	1032.0
B-n68-k9	3.48	0.02	0.8	3.47	0.20	1.8;1.4	3.40	5.08	1.0;0.0;0.2	3.31	101.20	1272.0
B-n78-k10	3.57	0.02	1.0	3.24	0.28	1.0;0.3	2.67	6.86	1.0;0.3;0.3	2.67	144.95	1221.0
Average	5.42	0.01		2.69	0.12		2.31	3.03		1.88	60.9	963.7

Table 6.9. Best relative per cent deviations obtained without local improvements on Augerat *et al.*'s test set B

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
B-n31-k5	1.36	0.00	0.9	1.11	0.02	1.9;1.4	1.05	0.40	0.9;0.0;0.1	0.79	7.70	672.0
B-n34-k5	0.80	0.00	1.2	0.23	0.03	1.2;0.0	0.23	0.51	1.2;0.0;0.0	0.23	9.80	788.0
B-n35-k5	2.44	0.00	1.0	2.44	0.03	1.5;1.2	2.26	0.52	1.1;0.1;1.8	2.14	10.55	955.0
B-n38-k6	3.37	0.00	0.9	3.15	0.03	1.4;0.4	2.36	0.69	1.5;0.5;0.2	2.28	12.54	805.0
B-n39-k5	3.22	0.00	1.1	1.12	0.05	1.4;0.3	1.09	0.70	1.4;0.3;0.0	1.09	13.41	549.0
B-n41-k6	8.33	0.00	0.8	6.61	0.05	0.7;0.6	5.94	0.73	0.8;0.7;0.2	3.94	15.08	829.0
B-n43-k6	5.39	0.00	0.8	2.15	0.03	1.2;0.4	1.74	0.80	0.9;0.1;0.4	1.74	16.62	742.0
B-n44-k7	3.16	0.02	1.0	3.16	0.05	1.8;0.8	2.86	0.94	1.9;0.9;1.8	2.83	18.07	909.0
B-n45-k5	0.82	0.02	1.0	0.82	0.03	1.0;0.0	0.82	0.84	1.1;0.0;0.8	0.49	18.27	751.0
B-n45-k6	7.35	0.00	1.1	6.26	0.05	0.9;0.6	5.20	0.89	0.9;0.7;0.8	5.12	18.65	678.0
B-n50-k7	1.05	0.02	1.1	0.93	0.06	1.1;0.1	0.59	1.22	1.0;0.0;0.2	0.59	23.40	741.0
B-n50-k8	3.20	0.02	0.9	3.06	0.06	1.5;0.6	2.46	1.23	1.9;1.0;0.3	2.01	24.92	1312.0

Table 6.9. Best relative per cent deviations obtained without local improvements on Augerat *et al.*'s test set B (continued)

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
B-n51-k7	8.65	0.02	1.2	-0.33	0.06	1.2;0.1	-0.42	1.22	1.2;0.0;0.8	-0.60	23.83	1032.0
B-n52-k7	2.40	0.00	1.0	2.40	0.05	1.0;0.0	2.40	1.34	1.3;0.0;1.5	1.33	25.73	747.0
B-n56-k7	3.78	0.00	0.7	2.37	0.08	1.3;0.5	2.28	1.53	0.8;0.0;0.2	2.21	30.71	707.0
B-n57-k7	7.53	0.00	1.3	1.25	0.06	1.7;0.7	0.33	1.72	1.1;0.0;0.5	-0.35	32.39	1153.0
B-n57-k9	3.47	0.00	0.9	1.36	0.08	0.9;0.0	1.36	1.64	0.9;0.0;0.0	1.36	32.92	1598.0
B-n63-k10	6.83	0.00	0.9	4.45	0.11	1.8.0.8	4.01	2.08	1.8;0.8;0.0	4.01	41.85	1496.0
B-n64-k9	7.03	0.00	1.0	7.03	0.11	1.8;0.9	6.76	2.08	1.1;0.7;1.8	5.82	41.92	861.0
B-n66-k9	7.63	0.02	0.8	5.53	0.11	1.9;1.2	2.90	2.30	1.7;0.9;0.9	2.67	45.31	1316.0
B-n67-k10	6.58	0.02	1.2	6.46	0.13	1.3;0.5	3.83	2.28	1.2;0.4;0.3	3.01	46.82	1032.0
B-n68-k9	3.60	0.02	1.0	3.60	0.11	1.0;0.0	3.60	2.39	1.0;0.0;0.2	3.47	49.17	1272.0
B-n78-k10	3.57	0.02	1.0	3.57	0.17	1.3;0.4	3.46	3.36	1.0;0.1;0.9	3.31	67.77	1221.0
Average	4.42	0.01		2.99	0.07		2.48	1.37		2.15	27.28	963.74

Table 6.10. Best relative per cent deviations obtained with local improvements on Augerat *et al.*'s test set C

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
P-n16-k8	6.39	0.00	1.3	5.28	0.00	1.8;0.6	0.43	0.13	1.7;0.4;1.4	0.43	3.75	450.0
P-n19-k2	18.75	0.00	1.3	4.08	0.05	0.3;0.9	3.90	0.95	0.6;1.1;0.5	2.23	19.04	212.0
P-n20-k2	8.33	0.00	0.1	8.33	0.05	0.7;1.2	1.82	1.11	0.6;1.1;0.4	1.82	23.00	216.0
P-n21-k2	11.94	0.00	0.1	11.94	0.06	0.6;1.7	0.81	1.45	0.4;1.6;1.8	0.81	29.35	211.0
P-n22-k2	10.88	0.00	0.1	10.88	0.08	0.6;1.7	0.86	1.89	0.4;1.6;1.8	0.86	37.48	216.0
P-n22-k8	-2.05	0.00	0.5	-2.36	0.02	1.5;0.0	-2.36	0.19	0.4.0.0;1.7	-2.36	4.33	603.0
P-n23-k8	1.98	0.00	1.2	1.46	0.02	1.4;0.2	1.46	0.28	1.4;0.2;0.0	1.46	5.03	529.0
P-n40-k5	13.18	0.00	1.2	10.49	0.13	0.2;1.2	1.55	2.41	0.2;1.2;0.0	1.55	49.02	458.0
P-n45-k5	14.26	0.00	1.2	4.38	0.16	1.9;0.7	2.20	3.39	1.9;0.7;0.0	2.20	73.82	510.0
P-n50-k10	5.51	0.00	1.6	3.81	0.06	1.2.0.1	2.41	1.50	1.2;0.1;0.0	2.41	29.49	696.0
P-n50-k7	6.12	0.00	1.7	4.63	0.11	1.7;0.8	3.01	2.39	1.7;0.7;0.3	3.01	49.36	554.0

Table 6.10. Best relative per cent deviations obtained with local improvements on Augerat *et al.*'s test set C (continued)

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
P-n50-k8	6.89	0.00	1.5	3.06	0.08	1.4;0.2	2.46	1.64	1.2;0.2;0.7	2.46	33.18	631.0
P-n51-k10	6.74	0.00	1.7	4.04	0.06	1.6;1.4	1.54	1.58	1.9;0.7;0.5	1.52	30.52	741.0
P-n55-k10	5.50	0.00	1.5	4.45	0.09	1.6;1.4	2.58	2.06	1.6;1.4;0.0	2.58	40.75	694.0
P-n55-k15	-1.11	0.00	1.0	-1.11	0.05	1.6;0.9	-2.99	1.67	1.6;0.9;0.0	-2.99	33.45	989.0
P-n55-k7	8.04	0.01	1.7	3.51	0.19	0.6;0.8	2.35	3.53	0.6;0.7;0.5	2.35	75.34	568.0
P-n55-k8	10.34	0.01	1.4	3.66	0.17	0.6;0.7	3.38	3.08	0.6;1.0;1.8	1.86	65.64	576.0
P-n60-k10	8.36	0.01	1.0	7.02	0.13	0.5;0.9	4.34	2.59	0.6;0.9;1.0	3.53	51.67	744.0
P-n60-k15	5.25	0.00	1.2	4.34	0.09	1.5;0.2	3.32	2.06	0.4;1.1;1.3	2.78	41.54	968.0
P-n70-k10	7.30	0.01	1.2	4.74	0.19	1.8;0.4	2.66	4.42	0.8;0.7;1.1	2.18	84.63	834.0
P-n76-k4	14.92	0.06	1.6	8.43	1.59	1.4;1.6	2.79	31.79	1.4;1.6;0.0	2.79	661.25	593.0
P-n76-k5	9.53	0.05	1.8	9.26	1.14	1.5;1.6	3.61	19.66	1.5;1.6;0.0	3.61	429.41	627.0
Average	8.05	0.01		5.20	0.20		1.92	4.08		1.69	85.05	573.6

Table 6.11. Best relative per cent deviations obtained without local improvements on Augerat *et al.*'s test set C

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
P-n16-k8	6.39	0.00	1.3	5.28	0.00	1.9;0.5	0.43	0.14	1.8;0.3;1.7	0.43	2.07	450.0
P-n19-k2	12.21	0.00	1.3	4.08	0.02	0.6;0.7	4.08	0.16	0.6;0.7;0.0	4.08	2.76	212.0
P-n20-k2	8.33	0.00	0.4	8.33	0.02	1.8;1.2	6.22	0.14	1.5;1.1;1.6	6.22	3.02	216.0
P-n21-k2	11.94	0.00	0.4	11.94	0.02	1.8;1.2	9.74	0.19	1.3;1.0;2.0	9.05	3.05	211.0
P-n22-k2	10.88	0.00	0.4	10.88	0.00	1.9;0.7	4.37	0.22	1.9;0.7;0.8	0.87	3.33	216.0
P-n22-k8	-2.05	0.00	0.9	-2.26	0.00	0.9;0.0	-2.26	0.19	0.9;0.0.0.0	-2.26	4.11	603.0
P-n23-k8	1.98	0.00	1.2	1.46	0.15	1.5;0.2	1.46	0.22	1.4;0.2;1.3	1.46	4.60	529.0
P-n40-k5	13.18	0.00	1.3	7.93	0.05	1.0;1.0	2.23	0.70	0.9;1.0;0.4	2.23	13.67	458.0
P-n45-k5	12.34	0.00	1.3	4.16	0.05	1.9.0.7	2.73	0.92	1.5;0.1;0.7	2.43	17.92	510.0
P-n50-k10	6.30	0.00	1.5	3.99	0.06	1.2;0.1	2.41	1.19	1.2;0.1;0.0	2.41	24.08	696.0
P-n50-k7	7.77	0.00	1.7	6.07	0.05	1.8;0.5	4.58	1.17	1.7;0.5;1.6	4.28	23.35	554.0

Table 6.11. Best relative per cent deviations obtained without local improvements on Augerat *et al.*'s test set C (continued)

Instance	CW		CW1			CW2			CW3			Best
	% Dev.	CPU	(λ)	% Dev.	CPU	(λ;μ)	% Dev.	CPU	(λ;μ;ν)	% Dev.	CPU	Known
P-n50-k8	6.87	0.00	1.5	3.06	0.06	1.4;0.2	2.46	1.22	1.2;0.2;0.7	2.46	23.83	631.0
P-n51-k10	6.74	0.00	1.7	4.61	0.08	1.5;0.6	2.15	1.28	1.0;0.3;0.1	1.89	25.00	741.0
P-n55-k10	6.12	0.02	1.9	4.20	0.08	1.4;0.3	3.18	1.50	1.2;0.1;1.7	3.06	30.67	694.0
P-n55-k15	-1.11	0.02	1.0	-1.11	0.08	1.6;0.9	-2.60	1.60	1.9;0.8;0.7	-2.78	32.25	989.0
P-n55-k7	8.92	0.00	1.2	3.79	0.06	1.4;0.4	3.20	1.51	2.0;0.8;1.7	2.99	29.33	568.0
P-n55-k8	9.66	0.02	1.4	3.96	0.08	1.3;0.3	3.21	1.55	1.3;0.3;1.9	2.09	28.84	576.0
P-n60-k10	7.55	0.00	0.9	7.14	0.09	1.9;0.6	4.74	1.84	2.0;0.5;0.3	2.74	36.16	744.0
P-n60-k15	5.06	0.00	0.8	4.02	0.09	0.8;0.0	4.02	1.86	0.9;0.0;0.5	3.59	37.97	968.0
P-n70-k10	7.54	0.00	0.8	4.47	0.13	0.6;0.4	2.39	2.58	0.6;0.4;0.0	2.39	52.26	834.0
P-n76-k4	15.49	0.01	1.1	10.29	0.16	1.7;0.8	6.79	2.91	1.7;0.8;0.0	6.79	61.74	593.0
P-n76-k5	13.14	0.02	2.0	8.10	0.16	1.6;0.8	5.14	3.06	2.0;0.5;1.5	3.77	61.79	627.0
Average	7.97	0.00		5.20	0.85		3.21	23.16		2.74	23.72	573.6

7. NEW ENHANCEMENTS OF ESAU AND WILLIAMS' SAVINGS HEURISTIC

7.1. Introduction

Many telecommunication networks, such as time sharing, on line banking, reservation, or registration, require remote terminals to be connected to a central processor. Terminals have specified demands for information that must flow between the central processor and them. Almost none of them use transmission lines continuously. They transmit information intermittently, which result in very small transmission times. A reasonable approach to increase the utilization is to connect several terminals into one transmission line, which gives the name multipoint. Each multipoint transmission line is in fact a tree spanning the vertices (terminals) sharing this line, and connected to the central vertex (computer center) only by one edge (single link). Then the problem becomes the design of minimal – cost multipoint transmission lines subject to capacity restrictions.

As we have mentioned before, the CMSTP is a difficult combinatorial optimization problem. It has been shown that even for the unit demand case it is NP-complete when $2 < Q < n/2$ [12]. In fact, many variants of the MST problem belong to the same class [96,97]. Largest problem instances solved to optimality have respectively at most 200 and 50 vertices for the homogenous and nonhomogeneous demand cases [98]. As a result of the inadequacy of exact methods, heuristics are widely used in practice. The CMSTP heuristics can be classified as classical heuristics and metaheuristics. Most of the early works realized during the 30 – year period starting from 1965 are within the first family. Starting from 1995 the interest has been shifted towards metaheuristics and mostly new members for this family have been produced. Current best heuristics for the CMSTP in term of the quality of the solutions produced belong to this family. The recent work by Amberg *et al.* provide an excellent survey on the exact algorithms and heuristics upto 1996 [99]. More recent developments are due to Sharaiha *et*

al. [100], Ahuja *et al.* [101,102], Patterson *et al.* [103] and, Patterson and Pirkul [104].

Classical heuristics consist of two classes: Construction heuristics and improvement heuristics. Construction heuristics gradually build a feasible solution by merging existing subtrees. They differentiate in their starting solution and merging criteria. The first group starts with a spanning forest feasible with respect to capacity constraints and apply a *greedy* rule to select the subtrees to be merged. The feasibility remains invariant and the number of connected components decreases throughout the iterations. They stop when a feasible spanning tree is obtained. The second group starts with a feasible trivial spanning tree, e.g. a star tree and improve this solution by merging multipoint transmission lines according to a *saving criterion.* This is repeated until no savings can be obtained anymore. The third group starts with a cheap but infeasible solution, e.g. a minimum spanning tree, and improves it gradually by decreasing the infeasibility and increasing the cost according to an *insertion cost.* Improvement heuristics attempt to improve a feasible solution. The first group use local edge exchange strategies to move from a spanning tree with lower infeasibility but higher cost. However the second group iteratively apply one of the construction heuristics with different initial conditions, namely by forcing some of the edges to be into or out of the final solution. Amberg *et al.* provide a detailed survey most of the classical heuristics with an emphasis on their complexity in their work where they propose a new vertex exchange procedure [99].

Metaheuristics intensively use neighborhood search methods to explore solution space without necessarily improving the objective function and sometimes allowing infeasible moves. Tabu Search (TS) [69, 99–101] and Adaptive Reasoning Technique (ART) [103, 105] are the most widely used metaheuristics to solve the CMSTP. Both methods start with an initial solution and move each step from the current solution to a new one selected from its neighborhood, not necessarily improving the objective function. Other metaheuristics are Simulated Annealing (SA) [66, 99, 106], Neural Networks (NN) [75, 104, 106] and Greedy Randomized Adaptive Search Procedure (GRASP) [101, 107, 108]. Similar to TS, SA also starts with an initial solution and move to a new one not necessarily improving the objective function, but no informa-

tion on the previous solution is collected. NN are mainly adaptive learning processes that continuously update some weights until an acceptable, namely a feasible or close to feasible solution is reached. GRASP is a neighborhood search algorithm that applies a local improvement scheme many times with different starting feasible solutions each of which is generated using some greedy randomized procedure.

Classical heuristics are unsophisticated and perform a limited exploration of the search space compared with metaheuristics. However, they are simple, which makes them easy to understand and easy to implement, and produce fairly good solutions very fast. Some of them are flexible and can be extended easily to handle many variants of the CMSTP. The performance of the metaheuristics is usually much higher than the ones of classical heuristics, but they require much more computational effort to have their parameters finely tuned – up. The large set of parameters they have, increase their flexibility; but they also make them context dependent and difficult to extend to other situations. Moreover, they have exponential running time times in the worst case [109, 110]. These probably explains why some of the classical and elementary heuristics, such as the EW heuristic [18], are very popular in practice. EW is extremely modest in its computational requirements and always ends up with a feasible solution. Amberg *et al.* point the superiority of EW's average performance in comparison with other heuristics in the literature with similar computation time [99]. They also remarked that EW has a worst case running time of $O(n^2 \log n)$ where n is the total number of nodes. Hence, it is embedded in many metaheuristics as a slave local search procedure [99,101,103,104], and used as a benchmark by several researchers. As a consequence, how one can improve the accuracy of EW without harming its speed and simplicity very much becomes an interesting question. It has been reported that the major weakness of EW is in its greedy behavior: An edge that is cheap early in the selection process may cause later edge selections resulting an expensive solution. A quick and intuitive remedy could be the consideration of additional information in its savings criterion, which can be introduced by means of additional parameters and/or new terms concerning the capacity restriction. This is the main motivation of this section; we propose new parallel savings heuristics which can control edge selections and combine distance and demand information in its saving criterion.

7.2. Esau and Williams' Savings Heuristic and Its Enhancements

The EW heuristic [18] is not only one of the earliest methods proposed for the solution of the CMSTP, but also probably the most widely used one in practice. Initially every vertex is directly connected to the central vertex and there are as many subtrees as the number of vertices. In other words, EW starts with a *star tree*. The star tree is a feasible topology, otherwise there is no feasible topology. Subsequently, EW attempts to reduce the cost as much as possible by modifying current solution without violating capacity constraints. This is done by first identifying two vertices i and j belonging to two different subtrees and yielding the largest saving so that total demand of the subtree of i and the subtree of j does not exceed Q. Then the edge $\{0, j\}$ is replaced with the edge $\{i, j\}$ resulting in the largest saving

$$\begin{aligned} s_{ij} &= (g_i + g_j) - (g_i + c_{ij}) \\ &= g_j - c_{ij}. \end{aligned} \tag{7.1}$$

Here g_i and g_j are the lengths of the gates connecting respectively the subtrees of vertices i and j to the central vertex. Finally the savings are updated according to the formula

$$s_{kl} = \begin{cases} s_{kl} - g_j + g_i & \text{for } k \in V \setminus (V_i \cup V_j \cup \{0\}) \text{ and } l \in V_j \\ 0 & \text{for } k, l \in V_j \text{ and } k \neq l \\ s_{kl} & \text{otherwise} \end{cases} \tag{7.2}$$

since the vertices in the subtree including vertex j now have edge $\{0, i\}$ as the new gate, whose length is g_i. We have implemented the parallel version of the algorithm rather than the sequential version. In the parallel version of the algorithm, the list of savings is tracked from the top in order to join the first pair of subtrees with the largest saving, without violating the feasibility. Whereas in the sequential version, the subtrees are considered first. For each subtree in turn, the saving list is tracked from the top in order to find the first feasible join with another subtree.

Although the feasibility of the final solution is guaranteed, the accuracy of EW is not always satisfactory. The low accuracy is mainly because of its greedy nature. At each step, it myopically merges two subtrees and can be caught at a local optimal solution. As an attempt to remedy this drawback, one can modify EW to take more control over the merging operations. For that purpose, Karnaugh [111] and Kershenbaum *et al.* [112] have proposed second order greedy algorithms (SOGA). A SOGA iteratively applies a construction procedure, namely a first order greedy algorithm (FOGA), such as EW, to different initial conditions forcing some of the edges into or out of the final solution. In each iteration, all possible modifications according to a given rule are tested. The best one is realized and the respective modifications are made permanent for the remaining iterations. These methods can also be seen as the early neighborhood search algorithms for the CMSTP, rather than enhancements of EW.

The earliest enhancement of EW appears as a special form of Kershenbaum and Chou's unified algorithm [113]. The authors propose a saving expression where g_i, vertex i's gate cost, is replaced with a parameterized weight in EW's saving expression (7.1). Another enhancement is due to Dai and Fujino [114]. They mainly contribute to the speed of EW. Computing the maximum saving value is computationally the most expensive part of EW. They employ component oriented saving computations instead the vertex – oriented one in order to increase the efficiency. They also modify the original heuristic to handle additional constraints on vertex order, degree and depth, which can also be seen as a demonstration for the flexibility of EW. They organize and maintain savings in a heap.

A limitation of EW is that a vertex, say vertex j, is always disconnected from the root by deleting its gate while it may have been more advantageous to remove the edge with the highest cost on the path linking vertex j to the root. In their recent work Bruno and Laporte have modified the EW algorithm to implement this idea [115]. This enhancement is easy to implement and the resulting algorithm is only slightly slower than the original method. However, the accuracy has been significantly improved especially on instances with large capacity bounds, Q.

The most recent attempt to enhance EW is due to Jothi and Raghavachari [109, 110]. They have modified the saving formula by taking into account the information coming from the total demand of the vertices belonging to the subtree to be connected.

7.3. New Enhancements

Although the feasibility of the final solution is guaranteed, the accuracy of EW is not always satisfactory. The low accuracy is mainly because of its greedy nature. At each step EW myopically merges two subtrees and can be caught at a local optimal solution. Once two vertices are joined it is not possible to disconnect them later. That is why one must be careful in joining the subtrees in order to escape from local minima. One remedy is to use EW as a local search procedure. As it can be remembered this was the motivation behind the second order greedy approaches by Karnaugh [111] and Kershenbaum *et al.* [112]. Another remedy can be the parametrization of the saving expression (7.1) similar to what Gaskell [84] and Yellow [85] propose to enhance the CW heuristic for the CVRP [17]. This results in the saving expression,

$$s_{ij} = g_j - \alpha \times c_{ij} \tag{7.3}$$

where α is the positive *tree shape* parameter. Notice that as it increases, greater emphasis is given to the distance between vertices i and j rather than their position relative to the central vertex. The search for the best network topology realized by changing α provides better near optimal solutions with an additional search effort for the best parameter value. Notice that, although the parametrization of EW's saving expression (7.1) seems similar to the one of Kershenbaum and Chou's earlier suggestion [113] it is in fact different. They parameterize EW's saving expression by means of vertex i's gate cost g_i. However, we consider the cost c_{ij} of connecting vertices i and j.

Another way to improve the performance of EW is to extend the parametric saving expression (7.3) to consider more information about the spatial distribution of the terminal vertices. One approach can be the use of asymmetry between the vertices

i and j with respect to the central vertex. A similar enhancement was previously proposed by Paessens for CW [86]. Parallel to his attempt for the CVRP it is possible to add one more term and parameter to saving expression (7.3) in order to obtain the following one

$$s_{ij} = g_j - \alpha \times c_{ij} + \beta \times |g_i - g_j| \,. \tag{7.4}$$

This new enhancement of EW can be run with different values of the parameters α and β to obtain different solutions. The effect of parameters α and β are computationally studied in the following section. The result is a considerable increase in the accuracy with a small additional computational cost.

While running the original EW and these two new enhancements, especially towards the end, the merge of subtrees with equal or very close savings occur often. Then it may be more interesting to consider also terminal demands while calculating savings. In general, CMSTP is a combination of two problems: Minimal Spanning Tree Problem (MSTP) and Bin Packing Problem (BPP). Hence, a saving expression which somehow combines the solution strategies proposed for them can increase the chance of obtaining higher improvements. For this purpose we adopt the well – known *first fit decrease* idea of Martello and Toth [90], which was originally used for the BPP: *put first larger items* into the bin. The inclusion of demand information provides additional control over the merge process. In short, we propose the following new savings criterion:

$$s_{ij} = g_j - \alpha \times c_{ij} + \beta \times |g_i - g_j| + \gamma \times (\sum_{k \in V_i} d_k + \sum_{k \in V_j} d_k)/\overline{d}. \tag{7.5}$$

Here d_i is the demand of terminal i, $\overline{d}$ is the average demand used for normalization, c_{ij} is the distance between terminals i and j, g_i and g_j are respectively the lengths of gates connecting the subtrees (multipoint lines) of terminals i and j and the central

vertex, and γ is the new parameter. Since it also includes previous enhancements, saving expression (7.5) is more general. We consider demands normalized with their average $\overline{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$, because we want to include the effect of the relative impact of terminal demands on the savings.

The use of demand effect in the saving formula is also proposed by Jothi and Raghavachari in their very recent works [109, 110]. They have proposed the use of the following expression for the Esau-Williams heuristic

$$s_{ij} = (g_j - c_{ij}) \times (\sum_{k \in V_i} d_k)^{\delta} \tag{7.6}$$

where δ is a fixed parameter between 0 and 1. Notice that, (7.6) is different than the one we proposed as the third term in (7.5). Jothi and Raghavachari use the demand information prioritizing the subtrees with greater total demand to grow bigger and bigger [109, 110]. Different than their enhancement, we include into the saving formula the demand information coming from both of the subtrees to be connected. In fact, at merging step, we give priority to pairs of subtrees with greatest sum of their vertices total demands.

7.4. Computational Results

In order to demonstrate the improvement in the accuracy, we conducted experiments with the original EW and the three new enhancements on the same test bed. Although a heap can provide a more efficient data organization we use the most simple data structure, namely a matrix, to store the savings in our implementation. In other words, we search the whole savings matrix starting from the very first cell and keep the best saving with the smallest indices i and j. Then, we try to merge the subtrees including terminals i and j. We join them if the total demand of their subtrees does not exceed capacity bound Q. In the other case, when the capacity is exceeded, the saving value s_{ij} is set to 0. We also update the savings matrix after the merge operation is performed: The savings belonging to the pairs of customers who are in the new subtree

are also updated according to (7.2).

All of our codes are written in C^{++} programming language. For only the verification of our codes we compared our implementation of the EW with the implementation of Sharaiha *et al.* [100] on seven sets of standard test problems: `tc40`, `tc80`, `te40`, `te80`, `cm50`, `cm100`, and `cm200`. Each set has 15 instances, which makes 105 test problem at sum. This is the test bed we use in our experiments. The sets which are identified as `tcX-Y QZ` have the central vertex (root of the final tree) centrally located. This is what the letter `c` means. The central vertex is eccentric for the sets `teX-Y QZ` as meant by the letter `e`. `X` is the number of terminals, `Y` is an instance label and the `Z` of `QZ` is the capacity bounds for the subtrees. As for the `cmX--Y QZ` instances, `X`, `Y` and `Z` have precisely the same meaning, but the capacity bounds are significantly larger. All of them can be downloaded from Beasley's Operational Research library [116]. There can be more than one edge with the same largest saving prior to a merge operation. Since it determines the two subtrees to be joined this selection has a direct influence on the final solution. We implement two strategies for a demonstration: *Last best* (LB) and *First best* (FB). Last best chooses the last one of edges with the highest saving encountered during the search in the current savings matrix. However, first best does the opposite; the first edge with the highest saving determines the two subtrees to be merged in this case. Our results are reported in the LB and FB columns of the Table 7.1 and Table 7.2. "Instance" column includes the standard codes describing the test problems. The columns with header SEW consist of the results due to Sharaiha *et al.* [100]. As it can be observed the type of the strategy can effect final solutions although it is not possible to say one is superior than the other. However, we can say that our implementation gives slightly better results for almost all problem instances. The overall averages of the final costs are respectively 884.26, 904.63, and 913.25 for LB, FB and SEW. Last best (LB) is a better strategy in the average. We prefer to use our EW implementation with the first best (FB) saving selection strategy for benchmarking since it gives an average cost closer to the one of SEW. One may suggest to improve final results further by going through a local improvement phase after every merge. This can be achieved by determining "locally" a MST spanning the vertices of the new subtree. Similar local improvement phases increase the accuracy of the saving

heuristics for the CVRP. During our preliminary computational tests we have observed marginal improvement only for a very few instances and preferred not to report them here in detail.

The results reported in Tables 7.3 – 7.9 are obtained by using the mentioned 7 test sets. The first column shows the instances. The second and third columns are obtained by using our implementation of the original EW. We run our implementation with the first best saving selection strategy. The values are relative per cent deviations from the best known values and the CPU seconds spent for its calculation. The fourth and fifth columns include values calculated with our first enhancement (EW1). The numbers in each cell are respectively tree – shape parameter α and the total costs; they are the best of the results of 19 runs each of which is realized with a value of α between 0.1 and 2. Initially α is set to 0.1 and incremented by 0.1 at every step. The entries of the sixth column are the averages of 19 CPU times in seconds each of which is obtained for one value of α. This format is repeated in columns $7-9$ for the second enhancement (EW2) and in columns $10-12$ for the third enhancement (EW3) with (α, β) pairs and (α, β, γ) triplets respectively. Notice that, in the original EW's update equation (7.1), $(\alpha = 1.0, \beta = 0.0, \gamma = 0.0)$ The best known values are given in the last columns. They are reported by Ahuja *et al.* [101, 102], Patterson *et al.* [103], Sharaiha *et al.* [100] and Amberg *et al.* [99]. The search effort becomes higher with the increase in the number of parameters. For both enhancements α, β and γ are chosen in the intervals $(0.1, 2)$, $(0, 2)$ and $(0, 2)$ respectively. The increment is selected 0.1 for all parameters. In other words, we have solved EW2, $20 \times 21 = 420$ times and EW3, $20 \times 21 \times 21 = 8820$ times and recorded the best of 420 and 8820 relative per cent deviations and corresponding parameter values in columns 7 and 8 for EW2 and columns 10 and 11 for EW3. The CPU times given in columns 9 and 12 are the averages of 420 and 8820 for EW2 and EW3 respectively. As an example, for instance cm50-3 Q200, it takes third enhancement EW3 1368 seconds (22.8 min.), which makes 0.155 seconds in the average, to obtain a total cost of 1208 as the best of 8820 values and corresponding parameters $(\alpha, \beta, \gamma) = (1.5, 0.9, 2.0)$. In all these tables last row includes column averages for relative per cent deviations and average CPU seconds. Averaged total costs give clues about the improvement in the accuracy.

We observe 9.083, 6.464 and 5.841 per cent relative deviations as the average of the whole test set (105 test instances) for our new heuristics EW1, EW2 and EW3 respectively. According to the results listed in Tables 5.3 – 5.9 the most important parameters seem to be the tree shape parameter α, and β. They assume non zero final values more frequently than γ does. This fact can also be deduced from the three average relative deviations. The one of EW2, which has α and β, is close to the one of EW3, which has α, β and γ as parameters. The capacity term in the savings expression becomes more important (γ has higher tendency to have non zero final values) for small capacity bounds Q, which is reasonable since this resource is scarce and there is not many subtrees that can satisfy this bound. In the contrary, when Q is large, the resource is not scarce and less care is given to capacities in the savings criterion, which result in a larger number of zero γ values.

Our final remark is on the determination of the intervals for (α, β, γ) parameters. This is a crucial point which may greatly affects the accuracy of EW1, EW2 and EW3. It may not be possible to improve EW when the intervals are too tight. On the other hand the required computational effort becomes excessive when the intervals are unnecessarily wide. In order to find a balance between the speed and the accuracy we have conducted additional experiments with EW3 where (α, β, γ) are respectively chosen within $[0.1, 3]$, $[0, 3]$, $[0, 3]$ and with increments 0.1. Wider intervals gave better results than the ones obtained with (α, β, γ) in $[0.1, 2]$, $[0, 2]$, $[0, 2]$ in only 26 of the 20010 $((31 \times 30 \times 30) - (21 \times 20 \times 20))$ additional runs. Besides, in all these 26 cases the values of β and γ were within $[0, 2]$ and α was within $[2, 3]$. This suggests that our choice of interval $[0, 2]$ for β and γ is reasonable and the additional computational effort spent for (α) in $[2, 3]$ is not worthless. Therefore, we believe that our choice of (α, β, γ) parameter intervals to be $[0.1, 2]$, $[0, 2]$, $[0, 2]$ is a good search strategy.

We have also experimented with the saving expression

$$s_{ij} = (g_j - \alpha \times c_{ij} + \beta \times |g_i - g_j|) \times (\sum_{k \in V_i} d_k)^{\delta}, \tag{7.7}$$

which is inspired by (7.6) of Jothi and Raghavachari [109,110]. In these computations, (α, β, δ) are chosen within the intervals $[0.1, 2]$, $[0, 2]$, $[0, 2]$ and with increments 0.1. For only 7 out of 105 instances we ended up with a nonzero δ. It means demand information contributes for only 6.66 per cent of test problems. Moreover, in any of these 7 instances we have observed better results than the ones we obtained with EW3. Therefore, we do not report them here.

Table 7.1. Costs by different EW implementations on tc40, tc80, te40 and te80 sets

Instance	LB	FB	SEW	Instance	LB	FB	SEW	Instance	LB	FB	SEW	Instance	LB	FB	SEW
tc40-1 Q3	774	775	777	tc80-1 Q5	1175	1201	1182	te40-1 Q3	1215	1203	1215	te80-1 Q5	2619	2635	2592
tc40-2 Q3	749	734	749	tc80-2 Q5	1157	1189	1170	te40-2 Q3	1158	1140	1144	te80-2 Q5	2624	2612	2631
tc40-3 Q3	728	743	728	tc80-3 Q5	1144	1140	1146	te40-3 Q3	1148	1143	1146	te80-3 Q5	2708	2680	2723
tc40-4 Q3	808	795	804	tc80-4 Q5	1154	1133	1160	te40-4 Q3	1151	1144	1156	te80-4 Q5	2619	2658	2630
tc40-5 Q3	758	766	760	tc80-5 Q5	1340	1398	1344	te40-5 Q3	1143	1145	1147	te80-5 Q5	2560	2504	2595
tc40-1 Q5	597	598	595	tc80-1 Q10	937	917	931	te40-1 Q5	857	885	857	te80-1 Q10	1716	1771	1735
tc40-2 Q5	588	616	588	tc80-2 Q10	928	925	917	te40-2 Q5	834	828	827	te80-2 Q10	1713	1729	1787
tc40-3 Q5	602	597	602	tc80-3 Q10	925	900	912	te40-3 Q5	820	831	820	te80-3 Q10	1781	1771	1828
tc40-4 Q5	627	632	645	tc80-4 Q10	916	930	924	te40-4 Q5	880	848	854	te80-4 Q10	1768	1786	1691
tc40-5 Q5	615	612	615	tc80-5 Q10	1075	1089	1092	te40-5 Q5	808	795	816	te80-5 Q10	1708	1725	1731
tc40-1 Q10	506	498	516	tc80-1 Q20	860	848	856	te40-1 Q10	639	645	649	te80-1 Q10	1308	1324	1336
tc40-2 Q10	501	490	505	tc80-2 Q20	850	842	856	te40-2 Q10	607	622	613	te80-2 Q20	1292	1344	1295
tc40-3 Q10	508	508	517	tc80-3 Q20	840	858	852	te40-3 Q10	616	590	596	te80-3 Q20	1342	1369	1340
tc40-4 Q10	518	512	524	tc80-4 Q20	832	838	860	te40-4 Q10	600	600	638	te80-4 Q20	1368	1371	1349
tc40-5 Q10	504	504	540	tc80-5 Q20	948	948	969	te40-5 Q10	593	593	597	te80-5 Q20	1251	1251	1271

Table 7.2. Costs by different EW implementations on cm50, cm100 and cm200 sets

Instance	LB	FB	SEW	Instance	LB	FB	SEW	Instance	LB	FB	SEW
cm50-1 Q200	1142	1167	1135	cm100-1 Q200	701	663	728	cm200-1 Q200	1327	1297	1427
cm50-2 Q200	1001	1007	1023	cm100-2 Q200	741	785	800	cm200-2 Q200	1700	1669	1839
cm50-3 Q200	1249	1226	1249	cm100-3 Q200	749	785	750	cm200-3 Q200	1781	1656	1840
cm50-4 Q200	801	809	834	cm100-4 Q200	553	582	625	cm200-4 Q200	1313	1323	1456
cm50-5 Q200	968	956	970	cm100-5 Q200	562	565	637	cm200-5 Q200	1319	1263	1369
cm50-1 Q400	718	718	731	cm100-1 Q400	317	307	375	cm200-1 Q400	553	508	606
cm50-2 Q400	658	658	642	cm100-2 Q400	332	337	376	cm200-2 Q400	620	742	752
cm50-3 Q400	739	739	741	cm100-3 Q400	302	336	363	cm200-3 Q400	717	671	801
cm50-4 Q400	579	571	583	cm100-4 Q400	289	296	322	cm200-4 Q400	486	555	534
cm50-5 Q400	638	639	643	cm100-5 Q400	286	252	323	cm200-5 Q400	515	508	579
cm50-1 Q800	514	511	550	cm100-1 Q800	211	205	255	cm200-1 Q800	339	301	329
cm50-2 Q800	546	545	531	cm100-2 Q800	198	197	235	cm200-2 Q800	353	373	422
cm50-3 Q800	541	560	565	cm100-3 Q800	194	197	238	cm200-3 Q800	400	379	460
cm50-4 Q800	491	491	514	cm100-4 Q800	202	196	248	cm200-4 Q800	301	319	357
cm50-5 Q800	513	512	515	cm100-5 Q800	217	202	232	cm200-5 Q800	351	330	413

Table 7.3. Best relative per cent deviations obtained on tc40 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	($\alpha;\beta$)	% Dev.	CPU	($\alpha;\beta;\gamma$)	% Dev.	CPU	Known
tc40-1 Q3	4.45	0.02	0.1	4.45	0.61	0.2;0.1	1.35	13.3	1.3;0.7;1.0	0.81	311.6	742
tc40-2 Q3	2.37	0.02	0.6	2.37	0.53	1.1;0.2	2.09	12.3	1.1;0.2;0.0	2.09	288.8	717
tc40-3 Q3	3.77	0.02	0.1	3.07	0.62	0.5;2.2	1.40	11.4	0.9;0.1;0.4	1.26	273.6	716
tc40-4 Q3	2.58	0.01	1.3	1.81	0.49	0.8;0.2	1.16	12.9	1.1;0.5;1.7	0.52	296.4	775
tc40-5 Q3	3.37	0.01	1.3	1.21	0.57	1.7;0.7	0.13	12.5	1.7;0.7;0.0	0.13	296.4	741
tc40-1 Q5	2.05	0.01	0.1	2.05	0.46	1.3;0.4	0.34	9.8	1.3;0.4;0.0	0.34	296.4	586
tc40-2 Q5	6.57	0.01	1.3	4.84	0.46	1.0;0.1	3.81	11.0	0.7;0.2;0.2	1.73	326.8	578
tc40-3 Q5	3.47	0.01	0.1	2.25	0.38	0.1;0.0	2.25	8.3	0.7;0.3;0.6	1.91	273.6	577
tc40-4 Q5	2.43	0.01	1.4	1.13	0.49	1.4;0.0	1.13	11.0	1.4;0.0;0.0	1.13	319.2	617
tc40-5 Q5	2.00	0.01	0.7	2.00	0.57	1.0;0.1	0.83	10.6	1.0;0.1;0.0	0.83	311.6	600
tc40-1 Q10	0.00	0.00	0.1	0.00	0.23	0.1;0.0	0.00	10.2	0.1;0.0;0.0	0.00	448.4	498
tc40-2 Q10	0.00	0.01	0.1	0.00	0.30	0.1;0.0	0.00	8.3	0.1;0.0;0.0	0.00	410.4	490
tc40-3 Q10	1.60	0.01	0.1	1.60	0.27	0.1;0.0	1.60	7.2	0.1;0.0;0.0	1.60	413.4	500
tc40-4 Q10	0.00	0.01	0.1	0.00	0.28	0.1;0.0	0.00	8.3	0.1;0.0;0.0	0.00	380.0	512
tc40-5 Q10	0.00	0.01	0.7	0.00	0.27	0.7;0.0	0.00	9.1	0.7;0.0;0.0	0.00	448.4	504
Average	2.31	0.01		1.79	0.44		1.07	10.4		0.82	339.7	610.2

Table 7.4. Best relative per cent deviations obtained on tc80 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	$(\alpha;\beta)$	% Dev.	CPU	$(\alpha;\beta;\gamma)$	% Dev.	CPU	Known
tc80-1 Q5	9.78	0.19	0.1	7.86	8.89	0.3;0.1	5.21	174.0	0.3;0.1;0.0	5.21	5411.2	1094
tc80-2 Q5	9.08	0.22	1	9.08	10.24	1.4;0.1	5.69	184.3	0.6;0.1;0.1	3.76	5510.0	1090
tc80-3 Q5	6.84	0.19	1.4	5.81	8.57	1.4;0.0	5.81	142.5	1.1;0.4;0.1	4.50	4757.6	1067
tc80-4 Q5	5.89	0.19	0.3	5.89	8.95	0.4;0.2	5.14	142.8	0.9;0.2;0.1	4.02	4628.4	1070
tc80-5 Q5	10.25	0.22	1.4	9.31	9.58	1.2;0.6	4.81	170.2	1.2;0.6;0.0	4.81	4810.8	1268
tc80-1 Q10	4.44	0.09	0.5	4.44	5.99	0.5;0.0	4.44	119.7	0.5;0.0;0.0	4.44	7136.4	878
tc80-2 Q10	5.71	0.15	0.1	5.71	7.24	0.9;0.1	4.57	131.1	0.9;0.2;0.5	2.51	7265.6	875
tc80-3 Q10	3.57	0.07	0.8	3.57	4.03	0.8;0.0	3.57	120.4	0.8;0.0;0.0	3.57	7539.2	869
tc80-4 Q10	7.76	0.14	0.8	7.76	8.08	0.8;0.3	5.10	130.7	1.9;0.9;0.1	3.94	7668.4	863
tc80-5 Q10	9.12	0.17	1.3	8.42	8.00	1.3;0.5	5.11	167.2	1.3;0.5;0.0	5.11	7311.2	998
tc80-1 Q20	1.68	0.07	0.1	1.20	2.81	0.4;0.1	0.96	46.3	0.4;0.1;0.0	0.96	13163.2	834
tc80-2 Q20	2.68	0.08	0.1	2.68	4.20	0.4;0.1	0.73	72.9	0.4;0.1;0.0	0.73	12821.2	820
tc80-3 Q20	3.62	0.13	0.1	3.62	7.30	0.5;0.1	0.48	70.3	0.5;0.1;0.0	0.48	14075.2	828
tc80-4 Q20	2.20	0.07	0.8	2.20	2.76	0.6;0.1	1.22	63.1	1.5;0.5;0.1	0.49	14014.4	820
tc80-5 Q20	3.49	0.09	0.8	3.49	7.13	0.8;0.3	3.28	111.7	0.8;0.3;0.0	3.28	12638.8	916
Average	5.74	0.14		5.40	6.92		3.74	123.1		3.19	8583.4	952.7

Table 7.5. Best relative per cent deviations obtained on te40 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	($\alpha;\beta$)	% Dev.	CPU	($\alpha;\beta;\gamma$)	% Dev.	CPU	Known
te40-1 Q3	1.09	0.05	0.8	1.09	1.14	1.5;0.8	1.01	25.4	0.4;0.1;0.2	0.59	524.4	1190
te40-2 Q3	3.35	0.06	0.1	2.27	1.08	0.3;0.2	1.72	22.8	0.3;0.2;0.4	0.54	487.1	1103
te40-3 Q3	2.51	0.06	1.2	1.52	1.10	0.2;0.1	1.43	25.1	1.2;0.0;1.1	0.90	532.0	1115
te40-4 Q3	1.06	0.05	1.6	0.97	1.03	1.9;0.9	0.35	22.8	1.9;0.9;0.0	0.35	486.4	1132
te40-5 Q3	3.71	0.05	0.3	1.45	1.08	0.3;0.0	1.45	23.6	1.1;0.2;0.3	1.00	500.8	1104
te40-1 Q5	6.63	0.06	0.8	6.63	1.33	0.9;0.1	4.22	24.3	1.8;1.3;0.1	3.61	585.2	830
te40-2 Q5	4.55	0.06	1.2	3.16	1.22	0.9;0.1	1.89	23.9	0.9;0.6;1.9	1.89	570.0	792
te40-3 Q5	4.27	0.06	0.1	3.64	1.24	0.1;0.0	3.64	25.1	1.3;0.6;0.0	2.51	592.8	797
te40-4 Q5	4.18	0.05	0.5	3.93	1.18	1.4;0.6	1.84	22.8	1.4;0.6;0.0	1.84	562.4	814
te40-5 Q5	1.40	0.05	0.9	1.40	1.06	0.9;0.0	1.40	21.3	0.9;0.0;0.0	1.40	524.4	784
te40-1 Q10	8.22	0.03	0.8	8.22	1.33	1.8;1.2	5.70	19.0	1.8;1.2;0.0	5.70	691.6	596
te40-2 Q10	8.55	0.03	0.7	8.55	1.18	1.1;0.6	6.28	17.4	0.9;0.4;0.1	6.11	668.8	573
te40-3 Q10	3.87	0.04	0.1	3.87	1.22	1.1;0.6	3.70	18.6	1.1;0.6;0.0	3.70	668.8	568
te40-4 Q10	0.67	0.05	0.5	0.67	1.03	0.5;0.0	0.67	15.2	0.5;0.0;0.0	0.67	570.0	596
te40-5 Q10	3.67	0.03	1	3.67	0.95	0.8;0.1	3.67	17.1	1.6;0.0;1.4	2.80	615.6	572
Average	3.85	0.05		3.40	1.14		2.60	21.6		2.24	572.1	837.7

Table 7.6. Best relative per cent deviations obtained on te80 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	($\alpha;\beta$)	% Dev.	CPU	($\alpha;\beta;\gamma$)	% Dev.	CPU	Known
te80-1 Q5	4.11	0.89	0.1	1.74	19.5	0.4;0.1	0.20	390.2	0.4;0.1;0.0	0.20	8633.6	2531
te80-2 Q5	3.57	1.09	1.2	2.93	21.8	1.2;0.0	2.93	455.2	1.3;0.7;0.5	2.74	9872.4	2522
te80-3 Q5	3.36	0.94	0.1	3.01	20.5	1.9;0.9	2.66	431.6	1.9;0.9;0.0	2.66	9317.6	2593
te80-4 Q5	4.69	1.00	1.3	3.58	20.6	0.5;0.2	2.44	422.9	1.0;0.3;1.6	2.13	9256.8	2539
te80-5 Q5	1.87	0.84	0.1	1.71	18.5	1.6;0.6	0.94	389.1	1.6;0.6;0.0	0.94	8618.4	2458
te80-1 Q10	8.58	0.79	1.4	8.15	21.9	1.2;0.1	4.90	326.0	1.2;0.1;0.0	4.90	10533.6	1631
te80-2 Q10	7.93	0.94	0.4	6.99	17.5	0.4;0.0	6.99	413.4	0.9;0.1;0.5	5.18	11582.4	1602
te80-3 Q10	6.69	0.91	1	6.69	31.0	1.1;0.1	4.64	420.2	1.1;0.1;0.0	4.64	11802.8	1660
te80-4 Q10	10.66	0.92	0.6	10.66	23.3	0.7;0.5	3.78	402.4	0.7;0.5;0.0	3.78	11643.2	1614
te80-5 Q10	8.76	0.78	0.1	8.76	20.2	1.7;0.7	6.43	338.9	1.9;0.6;0.1	6.18	10062.4	1586
te80-1 Q20	5.41	0.29	0.8	5.41	9.0	1.2;0.1	2.31	195.3	1.2;0.1;0.0	2.31	13414.0	1256
te80-2 Q20	11.91	0.69	1.9	11.74	20.2	0.6;0.1	5.66	241.6	0.6;0.1;0.0	5.66	14379.2	1201
te80-3 Q20	8.91	0.78	0.8	8.91	23.1	1.4;0.1	5.97	309.3	1.4;0.1;0.0	5.97	14873.2	1257
te80-4 Q20	9.94	0.67	0.6	9.94	23.1	0.8;0.5	8.18	302.4	1.8;0.5;0.1	6.42	16932.8	1247
te80-5 Q20	1.62	0.13	0.1	1.62	5.7	0.1;0.0	1.62	194.9	0.1;0.0;0.0	1.62	14858.0	1231
Average	6.53	0.78		6.12	19.7		3.98	348.9		3.69	11718.6	1795.2

Table 7.7. Best relative per cent deviations obtained on cm50 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	($\alpha;\beta$)	% Dev.	CPU	($\alpha;\beta;\gamma$)	% Dev.	CPU	Known
cm50-1 Q200	6.28	0.11	0.8	2.82	2.24	0.9;0.1	2.55	46.7	1.1;0.3;0.1	1.64	1140.0	1098
cm50-2 Q200	3.39	0.10	1	3.39	1.98	1.0;0.0	3.39	44.8	0.9;0.0;0.1	2.77	1132.4	974
cm50-3 Q200	3.37	0.12	1	3.37	2.49	1.1;0.1	3.20	56.2	1.5;0.9;2.0	1.85	1334.5	1186
cm50-4 Q200	1.13	0.06	1	1.13	1.50	1.1;0.1	0.88	28.8	1.1;0.1;0.1	0.13	790.4	800
cm50-5 Q200	3.02	0.09	0.8	0.97	1.84	0.8;0.1	0.97	38.7	0.8;0.0;0.0	0.97	965.2	928
cm50-1 Q400	5.43	0.08	0.8	4.85	2.15	1.4;0.9	2.94	39.9	1.9;1.2;1.1	2.20	1269.2	681
cm50-2 Q400	4.28	0.05	0.8	3.01	1.54	0.8;0.2	1.58	28.7	1.3;0.4;0.1	1.27	1041.2	631
cm50-3 Q400	0.96	0.11	1	0.96	2.43	1.0;0.0	0.96	50.1	1.0;0.0;0.0	0.96	1451.6	732
cm50-4 Q400	1.24	0.04	0.9	1.24	1.14	0.9;0.1	0.89	20.9	0.9;0.1;0.0	0.89	782.8	564
cm50-5 Q400	4.58	0.07	0.8	4.58	2.05	0.8;0.1	1.64	34.5	0.7;0.1;0.2	0.33	1109.6	611
cm50-1 Q800	3.23	0.06	1	3.23	1.14	1.0;0.0	3.23	24.7	1.0;0.0;0.0	3.23	1352.8	495
cm50-2 Q800	6.24	0.05	1.1	5.07	1.56	0.8;0.1	1.75	16.3	0.8;0.1;0.0	1.75	1162.8	513
cm50-3 Q800	5.26	0.08	0.5	5.26	2.19	0.5;0.1	1.69	34.5	0.5;0.1;0.0	1.69	1444.0	532
cm50-4 Q800	4.25	0.02	0.3	0.85	0.61	0.3;0.0	0.85	14.2	0.3;0.0;0.0	0.85	919.6	471
cm50-5 Q800	4.07	0.07	0.5	2.24	1.81	1.1;0.2	1.83	24.9	1.1;0.2;0.0	1.83	1261.6	492
Average	3.78	0.07		2.86	1.78		1.89	33.6		1.49	1143.8	713.8

Table 7.8. Best relative per cent deviations obtained on cm100 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	($\alpha;\beta$)	% Dev.	CPU	($\alpha;\beta;\gamma$)	% Dev.	CPU	Known
cm100-1 Q200	28.49	0.50	0.9	28.49	15.1	1.0;0.1	21.71	344.6	1.5;0.1;1.1	19.96	9591.2	516
cm100-2 Q200	31.71	0.53	0.6	31.71	16.3	0.8;0.1	21.98	416.8	1.5;0.1;0.5	17.79	10966.8	596
cm100-3 Q200	45.10	0.65	0.2	40.67	17.5	1.1;0.1	17.74	404.3	1.9;0.2;1.1	14.97	10845.2	541
cm100-4 Q200	33.18	0.38	0.2	27.69	11.3	0.5;0.1	27.23	303.6	1.5;0.1;1.2	19.22	8808.4	437
cm100-5 Q200	32.94	0.33	1.1	32.71	10.4	0.8;0.2	17.18	274.7	0.8;0.2;0.0	17.18	8261.2	425
cm100-1 Q400	21.83	0.29	0.5	19.84	9.1	0.5;0.0	19.84	137.9	0.5;0.0;0.1	15.08	10860.4	252
cm100-2 Q400	21.22	0.41	0.5	16.19	12.2	0.5;0.0	16.19	273.9	0.5;0.0;0.0	16.19	11985.2	278
cm100-3 Q400	42.37	0.44	1.1	29.66	12.1	1.9;0.1	21.61	232.5	1.4;0.0;0.2	19.92	10009.2	236
cm100-4 Q400	35.16	0.43	0.4	10.05	7.8	0.4;0.0	10.05	142.1	0.4;0.0;0.0	10.05	8428.4	219
cm100-5 Q400	13.00	0.20	1	13.00	7.1	1.8;0.1	11.66	137.9	1.8;0.1;0.2	10.31	8223.2	223
cm100-1 Q800	12.64	0.19	1.6	7.69	5.5	1.6;0.0	7.69	129.2	1.6;0.0;0.0	7.69	14470.4	182
cm100-2 Q800	10.06	0.21	0.6	10.06	6.9	1.8;0.1	8.94	177.1	1.8;0.1;0.0	8.94	20732.8	179
cm100-3 Q800	12.57	0.18	0.9	12.57	5.5	1.7;0.1	11.43	104.5	1.6;0.0;0.1	8.00	13262.0	175
cm100-4 Q800	7.10	0.11	0.9	7.10	3.8	1.5;0.1	5.46	91.9	1.5;0.1;0.0	5.46	15131.6	183
cm100-5 Q800	8.60	0.16	0.9	5.91	4.5	0.9;0.0	5.91	107.5	0.9;0.0;0.0	5.91	15587.6	186
Average	23.73	0.33		19.56	9.7		14.97	218.6		13.11	11810.9	308.5

Table 7.9. Best relative per cent deviations obtained on cm200 test set

Instance	EW		EW1			EW2			EW3			Best
	% Dev.	CPU	(α)	% Dev.	CPU	($\alpha;\beta$)	% Dev.	CPU	($\alpha;\beta;\gamma$)	% Dev.	CPU	Known
cm200-1 Q200	27.53	8.09	1.6	26.25	248.6	1.9;0.1	18.98	6420.8	1.2;0.1;0.3	16.13	178631.9	1017
cm200-2 Q200	36.69	9.72	0.6	34.97	285.2	1.6;0.4	21.21	7259.5	1.0;0.1;1.2	20.64	192622.0	1221
cm200-3 Q200	21.32	9.53	0.7	20.95	287.6	1.6;0.1	16.04	7465.8	1.6;0.1;0.5	13.19	196254.8	1365
cm200-4 Q200	42.72	8.41	1.6	41.75	256.2	1.9;0.2	22.65	5917.3	1.7;0.2;0.2	22.11	168233.6	927
cm200-5 Q200	30.88	6.97	0.9	30.88	232.2	1.2;0.2	19.27	5503.1	1.6;0.2;0.1	17.31	157919.6	965
cm200-1 Q400	27.96	4.31	1	27.96	161.8	1.4;0.1	23.93	3729.3	1.4;0.1;0.0	23.93	175031.8	397
cm200-2 Q400	55.23	9.81	1.8	53.77	311.6	1.5;0.1	26.99	5199.5	1.6;0.1;0.1	25.94	199538.0	478
cm200-3 Q400	19.82	5.93	0.1	19.82	203.3	1.8;0.1	19.64	5053.6	1.8;0.1;0.0	19.64	201255.6	560
cm200-4 Q400	41.58	6.09	0.9	41.58	213.2	1.3;0.1	21.68	3505.8	1.3;0.1;0.0	21.68	168674.4	392
cm200-5 Q400	20.95	3.83	1	20.95	138.7	1.5;0.1	18.33	3559.1	1.5;0.1;0.0	18.33	173606.8	420
cm200-1 Q800	18.50	2.21	0.1	16.14	70.6	0.1;0.0	16.14	1569.4	0.1;0.0;0.0	16.14	261029.6	254
cm200-2 Q800	26.87	4.38	1.1	25.17	158.0	1.5;0.1	21.77	3393.4	1.5;0.1;0.0	21.77	300830.8	294
cm200-3 Q800	4.99	3.06	0.6	4.99	113.4	0.6;0.0	4.99	3994.1	0.6;0.0;0.0	4.99	330805.2	361
cm200-4 Q800	16.00	3.29	1	16.00	145.1	1.0;0.0	16.00	2039.8	1.0;0.0;0.0	16.00	279398.8	275
cm200-5 Q800	13.01	2.74	1.1	11.64	104.8	1.1;0.0	11.64	2607.5	1.1;0.0;0.0	11.64	300352.0	292
Average	26.94	5.89		26.19	195.4		18.62	4481.2		17.96	218945.6	614.5

8. CONCLUSIONS

With this dissertation we first propose new heuristic algorithms for the UCLP based on the ideas originally proposed for two well-known combinatorial optimization problems: The Asymmetric Travelling Salesman Problem and The Linear Ordering Problem. One of the new heuristics, MR, appears to be much more efficient and accurate than the other considered known and new heuristics. Then, we have devised a branch and bound algorithm for a special case of the UCLP: Balanced UCLP. We have used MR as the upper bounding procedure and observed it is very efficient. Moreover, we have also developed a new move based dominance rule. The techniques used in our branch and bound algorithm may be exported for other problems with similar structure. Some of them are the LOP, the permutation flow–shop scheduling problem, the single machine scheduling problem, etc.

Second, we propose new ATSP formulations, with $O(n^3)$ subtour elimination constraints, based on a formulation originally devised for the balanced case of the UCLP. We then improve this new formulation to obtain tighter linear programming relaxations, while maintaining $O(n^3)$ subtour elimination constraints. Their relative strength is analyzed and compared with several major ATSP formulations both polyhedral standpoint and a computational testing. It turns out that one of $O(n^3)$ constrained ATSP formulations, is stronger than any known multi-commodity flow formulation. We have not only classified the new formulations with the existing ones, but also we have theoretically compared some of the known formulations.

All these new ATSP formulations have quite efficient subtour elimination constraints that one can also use them for the CVRP. Moreover, although we have not considered in this thesis, the classification of Capacitated Vehicle Routing Problems is also another important further research topic.

Third, we propose a new enhancement of the CW heuristic for the CVRP. Compared with the previous enhancements of the Clarke–Wright heuristic considers also

customer demands in its saving criterion. This is realized by introducing a new term and a multiplier.

One problem with similar structure to the CVRP, is the CMSTP. Similarly, we put forward new enhancements of the EW heuristic for the CMSTP. Both, CW and EW run in the same way, based on a similar saving criterion. For the EW, in addition to the demand information, we have adopted enhancements based on the ideas originally proposed for the CVRP by Gaskell [84], Yellow [85] and Paessens [86].

In their recent work on the CVRP heuristics Cordeau *et al.* [79] have introduced *accuracy*, *speed*, *simplicity*, and *flexibility* as what they believe the most important attributes of a good heuristic and compare well – known classical heuristics and best available metaheuristics according to these four criteria for the CVRP. They report that none of the classical heuristics is as accurate and flexible as any one of the metaheuristics; but the CW is very fast and simple to implement. These are probably the reasons of its popularity for the CVRP and explain its wide usage in commercial software. Analogously, the four criteria used by Cordeau *et al.* [79] to evaluate the CVRP heuristics can also be considered for the CMSTP heuristics. When this is done same verdict can be declared about the EW: it has low accuracy, very high speed, very high simplicity and low flexibility. Although, both CW and EW are forty years old algorithms, they are still used by many researchers and practitioners. As a consequence, how one can improve the accuracy of CW and EW without harming their speed and simplicity very much, becomes an important question whose answer will contribute to the practice of the CVRP and CMSTP, respectively.

Finally we should point out that all the computational tests of this dissertation are realized on a Sun Microsystems Blade – 1000 with a 750 MHz Ultrasparc III CPU and 2 GByte RAM, working within SOLARIS 8.0 environment.

APPENDIX A: PUBLICATIONS RELATED WITH THE DISSERTATION

Journal Articles

- Altınel, İ. K. and T. Öncan, "A New Enhancement of the Clarke and Wright Savings Heuristic for the Capacitated Vehicle Routing Problem", submitted to *Journal of the Operational Research Society*, accepted for publication.
- Altınel, İ. K., T. Öncan and A. T. Ünal, "New Extended Formulations for the Asymmetric Travelling Salesman Problem and The Strength of Their Linear Programming Relaxations", submitted to *European Journal of Operations Research*, first revision.
- Altınel, İ. K. and T. Öncan, "Enhancements of the Esau-Williams Heuristic for the Capacitated Minimum Spanning Tree Problem", submitted to *European Journal of Operations Research.*
- Altınel, İ. K. and T. Öncan, "The Design of Optimal Unidirectional Cyclic Layouts", submitted to *International Journal of Production Research.*
- Altınel, İ. K. and T. Öncan, "Aggregating the Assignment and Subtour Elimination Constraints", submitted to *Operations Research Letters.*

Conferences

- Altınel, İ. K. and T. Öncan, "New Enhancements of the Esau-Williams Heuristic for the Capacitated Minimum Spanning Tree Problem", *International Conference of Informatics*, İzmir, Turkey, September 2004.
- Öncan, T. and İ. K. Altınel, "New Heuristic Solution Methods for the Unidirectional Cyclic Layout Problem", in Turkish, *Operations Research and Industrial Engineering Conference (YA/EM 2004)*, pp. 73–75, Çukurova University, Turkey, June 2004.
- Altınel, İ. K. and T. Öncan, "A New $n(n-1)^2+1$ Constrained Asymmetric

Travelling Salesman Problem Formulation", in Turkish, *Operations Research and Industrial Engineering Conference (YA/EM 2004)*, pp. 545, Çukurova University, Turkey, June 2004.

- Öncan, T. and İ. K. Altınel, "A New Enhancement of the Clarke-Wright Savings Heuristic for the Capacitated Vehicle Routing Problem", *Euro/Informs Joint International Meeting*, İstanbul, Turkey, July 2003.
- Öncan, T. and İ. K. Altınel, "A New Travelling Salesman Problem Formulation and Its Linear Programming Relaxation Strength", in Turkish, *Operations Research and Industrial Engineering Conference (YA/EM 2002)*, Yeditepe University, Turkey, July 2002.
- Öncan, T. and İ. K. Altınel, "Two New Travelling Salesman Problem Formulations and Their Comparisons", in Turkish, *Operations Research and Industrial Engineering Conference (YA/EM 2000)*, East Mediterranean University, NCTR, July 2000.

Technical Reports

- Altınel, İ. K. and T. Öncan, *A New Enhancement of the Clarke and Wright Savings Heuristic for the Capacitated Vehicle Routing Problem*, Research Paper FBE-IE-12/2003-02, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Bebek, İstanbul, Turkey, February 2003.
- Altınel, İ. K. and T. Öncan, *Enhancements of the Esau-Williams Heuristic for the Capacitated Minimum Spanning Tree Problem*, Research Paper FBE-IE-03/2003-03, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Bebek, İstanbul, Turkey, April 2003.
- Altınel, İ. K., T. Öncan and A. T. Ünal, *New Extended Formulations for the Asymmetric Travelling Salesman Problem and The Strength of Their Linear Programming Relaxations*, Research Paper FBE-IE-05/2003-05, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Bebek, İstanbul, Turkey, August 2003.
- Altınel, İ. K. and T. Öncan, *The Design of Optimal Unidirectional Cyclic Layouts*, Research Paper FBE-IE-06/2004-09, Institute for Graduate Studies in Sci-

ence and Engineering, Boğaziçi University, Bebek, İstanbul, Turkey, May 2004.

- Altınel, İ. K. and T. Öncan, *Aggregating The Assignment and Subtour Elimination Constraints*, Research Paper FBE-IE-15/2004-22, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Bebek, İstanbul, Turkey, September 2004.

REFERENCES

1. Kouvelis, P. and M. W. Kim, "Unidirectional Loop Network Layout Problem in Automated Manufacturing Systems", *Operations Research*, Vol. 40, No. 3, pp. 533–550, 1992.

2. Papadimitriou, C., "The Euclidean Travelling Salesman Problem is NP-Complete", *Theoretical Computer Science*, Vol. 4, pp. 237–244, 1977.

3. Punnen, A. P., "The Traveling Salesman Problem: Applications, Formulations and Variations", in G. Gutin and A. P. Punnen (eds.), *The Traveling Salesman Problem and Its Variations*, pp. 1–28, Kluwer Academic Publishers, Dordrecht, 2002.

4. Menger, K. "Das Botenproblem", *Epigenists Eines Mathematischen Kolloquiums*, Vol. 2, pp. 11–12, 1932.

5. Robinson, J. B., *On the Hamiltonian Game: A Traveling Salesman Problem*, RAND Research Memorandum RM-303, 1949.

6. Lawler, E. L., J. K. Lenstra, A. H. G. Rinooy Kan and D. B. Shmoys (editors), *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 1985.

7. Reinelt, G., *The Travelling Salesman Problem: Computational Solutions for TSP Applications*, Springer-Verlag, Berlin, 1994.

8. Gutin, G. and A. P. Shmoys (editors), *The Travelling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrecht, 2002.

9. Toth, P. and D. Vigo (editors), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, SIAM Publishing, Philadelphia, PA, 2001.

10. Lenstra, J. and A. Rinnooy Kan, "Complexity of Vehicle Routing and Scheduling Problems", *Networks*, Vol. 11, pp. 221–227, 1981.

11. Gavish, B., "Topological Design of Telecommunication Networks - Local Access Design Methods", *Annals of Operations Research*, Vol. 33, pp. 17–71, 1991.

12. Papadimitriou, C., "The Complexity of the Capacitated Tree Problem", *Networks*, Vol. 8, pp. 217–230, 1978.

13. Dantzig, G. B., D. R. Fulkerson and S. M. Johnson, "Solutions of a Large Scale Traveling Salesman Problem", *Operations Research*, Vol. 2, pp. 363–410, 1954.

14. Langevin, A., F. Soumis and J. Desrosiers, "Classification of Travelling Salesman Problem Formulations", *Operations Research Letters*, Vol. 9, pp. 127–132, 1990.

15. Padberg, M. and T. Sung, "An Analytical Comparison of Different Formulations of the Travelling Salesman Problem", *Mathematical Programming*, Vol. 52, pp. 315–357, 1991.

16. Kıran, A. S., A. T. Ünal and S. Karabatı, "A Location Problem on Unicyclic Networks: Balanced Case", *European Journal of Operational Research*, Vol. 62, pp. 194–202, 1992.

17. Clarke, G. and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Depots", *Operations Research*, Vol. 12, pp. 568–581, 1964.

18. Esau, L. and K. Williams, "On Teleprocessing System Design. Part II- A method for Approximating the Optimal Network", *IBM Systems Journal*, Vol. 5, pp. 142–147, 1966.

19. Afentakis, P., "A Loop Layout Design Problem for Flexible Manufacturing Systems", *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 2, pp. 175–196, 1989.

20. Kouvelis, P., W. C. Chiang and A. S. Kıran, "A Survey of Layout Issues in the Flexible Manufacturing Systems", *OMEGA International Journal of Management Science*, Vol. 20, No. 3, pp. 375–390, 1992.

21. Tansel, C. B. and C. Bilen, "Move Based Heuristics for the Unidirectional Loop Network Layout Problem", *European Journal of Operational Research*, Vol. 108, pp. 36–48, 1998.

22. Bozer, Y. A. and S. C. Rim, *Exact Solution Procedures for the Circular Layout Problem*, Report No 89-33, University of Michigan, USA, 1989.

23. Kıran, A. S. and S. Karabatı, "Exact and Approximate Algorithms for the Loop Layout Problem", *Production Planning and Control*, Vol. 4, No. 3, pp. 253–259, 1993.

24. Cheng, R. and M. Gen, "Loop Layout Design Problem in Flexible Manufacturing Systems Using Genetic Algorithms", *Computers and Industrial Engineering*, Vol. 34, pp. 53–61, 1998.

25. Lee, S. D., K. S. Huang and C. P. Chiang,"Configuring Layout in Unidirectional Loop Manufacturing Systems", *International Journal of Production Research*, Vol. 39, No. 6, pp. 1183–1201, 2001.

26. Tansel, C. B. and C. Bilen, *Layout Problem in Flexible Manufacturing Systems*, Research Report 93-19, Department of Industrial Engineering, Bilkent University, Ankara 06533, Turkey, 1994.

27. Tansel, C. B. and C. Bilen, *Unidirectional Loop Network Layout Problem in Flexible Manufacturing Systems*, Research Report, Bilkent University, Turkey, 1993.

28. Francis, R. L, L. F. McGinnis and J. A. White, *Facility Layout and Location: An Analytical Approach*, 2^{nd} edition, Prentice-Hall, Englewood Cliffs, NJ, 1992.

29. Reinelt, G., *The Linear Ordering Problem: Algorithms and Applications*, Helder-

mann Verlag, Berlin, 1985.

30. Potts, C. N. and J. D. Whitehead, "Workload Balancing and Loop Layout in the Design of a Flexible Manufacturing System", *European Journal of Operational Research*, Vol. 129, pp. 326–336, 2001.

31. Lin, S., "Computer Solutions of the Traveling Salesman Problem", *Bell System Computer Journal*, Vol. 44, pp. 2245–2269, 1965.

32. Or, İ., *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking*, Ph.D. Dissertation, Northwestern University, Evanston, IL, 1976.

33. Becker, O., "Das Helmstädtersche Reihenfolgeoproblem - die Effizenz verschiedener, Näherungsverfahren", *Computer Uses in the Social Sciences Berichteiner Working Conference*, Wien, 1967.

34. Chanas, S. and P. Kobylanski, "A New Heuristic Algorithm Solving the Linear Ordering Problem", *Computational Optimization and Applications*, Vol. 6, pp. 191–205, 1996.

35. Boyd, S. and W. Cunningham, "Small Travelling Salesman Polytopes", *Mathematics of Operations Research*, Vol. 16, pp. 259–271, 1991.

36. Balas, E. and W. Pulleyblank, "The Perfectly Matchable Subgraph Polytope of a Bipartite Graph", *Networks*, pp. 486–516, 1983.

37. Wong, R., "Integer Programming Formulations of the Traveling Salesman Problem", *Proceedings of the IEEE International Conference of Circuits and Computers*, pp. 149–152, 1980.

38. Gouveia, L. and S. Voß, "A Classification of Formulations for the (Time-Dependent) Travelling Salesman Problem", *European Journal of Operational Research*, Vol. 83, pp. 69–82, 1995.

39. Orman, A. and H. Williams, *A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem*, Technical Report Preprint Series OR01, Faculty of Mathematical Studies, University of Southampton, Southampton, England, 1999.

40. Gouveia, L. and J. Pires, "The Asymmetric Travelling Salesman Problem and a Reformulation of the Miller-Tucker-Zemlin Constraints", *European Journal of Operational Research*, Vol. 112, pp. 134–146, 1999.

41. Gouveia, L. and J. Pires, "The Asymmetric Travelling Salesman Problem: on Generalizations of Disaggregated Miller-Tucker-Zemlin Constraints", *Discrete Applied Mathematics*, Vol. 112, pp. 129–145, 2001.

42. Grötschel, M. and M. Padberg, "Polyhedral theory", in E. L. Lawler, J. K. Lenstra, A. Rinnooy Kan, D. B. Shymoys (eds.), *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, 1985.

43. Miller, C. E., A. W. Tucker and R. A. Zemlin, "Integer Programming Formulations and Travelling Salesman Problems", *Journal of Association for Computing Machinery*, Vol. 7, pp. 326–329, 1960.

44. Desrochers, M. and G. Laporte, "Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints", *Operations Research Letters*, Vol. 10, pp. 27–36, 1991.

45. Laporte, G., Y. Nobert and M. Desrochers, "Optimal Routing Under Capacity and Distance Restrictions", *Operations Research*, Vol. 33, pp. 1050–1073, 1985.

46. Toth, P. and D. Vigo, "An Overview of Vehicle Routing Problems", in P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, SIAM, Pennsylvania, 2002.

47. Driscoll, P., *A New Hierarchy of Relaxation for 0-1 Mixed Integer Problems with Application to Some Specially Structured Problems*, Ph.D. Thesis, Department of

Industrial and Systems Engineering, Virginia Polytechnic Institute, Blacksburg, VA, 1995.

48. Sherali, H. and P. Driscoll, "On Tightening the Relaxations of Miller-Tucker-Zemlin Formulations for Asymmetric Traveling Salesman Problems", *Operations Research*, Vol. 50, pp. 656–669, 2002.

49. Sherali, H. and W. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Acedemic Publishers, Netherlands, 1999.

50. Gavish, B. and S. Graves, "The Travelling Salesman Problem and Related Problems", Working paper GR-078-78, Massachusetts Institute of Technology, 1978.

51. Gouveia, L., "A Result on Projection for the Vehicle Routing Problem", *European Journal of Operational Research*, Vol. 85, pp. 610–624, 1995.

52. Langevin, A., *Planification des Tournées de Véhicles*, Ph.D. Dissertation, École Polytechnique de Montréal, 1988.

53. Loulou, R. J., *On Multicommodity Flow Formulations for the TSP*, Working Paper, McGill University, Montréal, 1988.

54. Claus, A., "A New Formulation for the Travelling Salesman Problem", *SIAM Journal on Algebraic Discrete Methods*, Vol. 5, pp. 21–25, 1984.

55. Balas, E. and M. Fischetti, "Lifted Cycle Inequalities for the Asymmetric Travelling Salesman Problem", *Mathematics of Operations Research*, Vol. 24, pp. 273–292, 1999.

56. Balas, E. and M. Fischetti, "The Fixed-Outdegree 1-Arborescence Polytope", *Mathematics of Operations Research*, Vol. 17, pp. 1001–1018, 1992.

57. Pickard J. C. and M. Queyranne, "The Time-Dependent Traveling Salesman Prob-

lem and Its Application to the Tardiness Problem in One-Machine Scheduling", *Operations Research*, Vol. 26, pp. 86–110, 1978.

58. Fox, K. R., *Production Scheduling on Parallel Lines with Dependencies*, Ph.D. Dissertation, John Hopkins University, 1973.

59. Fox, K. R., B. Gavish and S. C. Graves, "An n-Constraint Formulation of the (Time-Dependent) Traveling Salesman Problem", *Operations Research*, Vol. 28, pp. 1018–1021, 1980.

60. Finke, G., A. Claus and E. Gunn, "A Two Commodity Network Flow Approach to the Traveling Salesman Problem", *Congressus Numerantum*, Vol. 1, pp. 167–178, 1984.

61. Reinelt, G., "TSPLIB- A Travelling Salesman Problem Library", *ORSA Journal on Computing*, Vol. 3, pp. 376–384, 1991, (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, last access March, 2003).

62. Fischetti, M. and P. Toth, "A Polyhedral Approach to the Asymmetric Travelling Salesman Problem", *Management Science*, Vol. 43, pp. 1520–1536, 1997.

63. Toth, P. and D. Vigo, "Exact Solution of the Vehicle Routing Problem", in T. Crainic and G. Laporte (eds.), *Fleet Management and Logistics*, pp. 1–31, Kluwer, Boston, 1998.

64. Laporte, G., M. Gendreau, J. Potvin and F. Semet, "Classical and Modern Heuristics for the Vehicle Routing Problem", *International Transactions in Operations Research*, Vol. 7, pp. 285–300, 2000.

65. Laporte, G. and F. Semet, "Classical Heuristics for the Vehicle Routing Problem", in P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pp. 109–128, SIAM, Philadelphia, PA, 2001.

66. Van Laarhoven, P. J. M. and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht, 1987.

67. Van Breedam, A., "Improvement Heuristics for the Vehicle Routing Problem Based on Simulated Annealing", *European Journal of Operational Research*, Vol. 86, pp. 480–490, 1995.

68. Osman, I., "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Annals of Operations Research*, Vol. 41, pp. 421–451, 1993.

69. Glover, F. and M. Laguna, *Tabu Search*, Kluwer, Boston, MA, 1997.

70. Rochat, Y. and E. Taillard, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics*, Vol. 1, pp. 147–167, 1995.

71. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley, Reading, MA, 1989.

72. Schmitt II, L. and M. Amini, "Parametric Experimentation with a Genetic Algorithmic Configuration for Solving the Vehicle Routing Problem", in P. Toth and D. Vigo (eds.), *Proceedings of the DSI National Meeting*, Orlando, FL, 1996.

73. Dorigo, M., V. Maniezzo and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man and Cybernetics B*, Vol. 26, pp. 29–41, 1996.

74. Bullnheimer, B., R. F. Hartl and J. Strauss, "Applying the Ant System to the Vehicle Routing Problem", in S. Voss, S. Martello, I. H. Osman and C. Roucairol (eds.), *Meta–Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pp. 109–120, Kluwer, Boston, MA, 1998.

75. Hertz, J., A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Com-*

putation, Lecture Notes Series I, Santa Fe Institute Studies in the Sciences of Complexity, Addison – Wesley: Redwood City CA, 1991.

76. Kohonen, T., *Self-organizing Maps*, Springer Series in Information Sciences, Springer-Verlag, Berlin, 1995.

77. Ghaziri, H., "Supervision in the Self-Organizing Feature Map: Application to Vehicle Routing Problem", in I. Osman and J. Kelly (eds.), *Metaheuristics: Theory and Applications*, Kluwer, Boston, MA, 1996.

78. Gendreau, M., G. Laporte and J. Potvin, "Metaheuristics for the Capacitated VRP", in P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, pp. 129–154, SIAM Publishing, Philadelphia, PA, 2001.

79. Cordeau, J., M. Gendreau, G. Laporte, J. Potvin and F. Semet, "A Guide to Vehicle Routing Heuristics", *Journal of the Operational Research Society*, Vol. 53, pp. 512–522, 2002.

80. Vigo, D., "A Heuristic Algorithm for the Asymmetric Capacitated Vehicle Routing Problem", *European Journal of Operational Research*, Vol. 89, pp. 108–126, 1996.

81. Salhi, S. and G. Nagy, "A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauls", *Journal of the Operational Research Society*, Vol. 50, pp. 1034–1042, 1999.

82. Wade, A. and S. Salhi, "An Investigation into a New Class of Vehicle Routing Problem with Backhauls", *OMEGA*, Vol. 30, pp. 479–487, 2002.

83. Christofides, N., A. Mingozzi and P. Toth, "The Vehicle Routing Problem", in N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), *Combinatorial Optimization*, pp. 315–338, Wiley, Chichester, 1979.

84. Gaskell, T., "Bases for the Vehicle Fleet Scheduling", *Operational Research Quarterly*, Vol. 18, pp. 281–295, 1967.

85. Yellow, P., "A Computational Modification to the Savings Method of Vehicle Scheduling", *Operational Research Quarterly*, Vol. 21, pp. 281–283, 1970.

86. Paessens, H., "The Savings Algorithm for the Vehicle Routing Problem", *European Journal of Operational Research*, Vol. 34, pp. 336–344, 1988.

87. Golden, B. L., T. L. Magnanti and H. Q. Nguyen, "Implementing Vehicle Routing Algorithms", *Networks*, Vol. 7 , pp. 340–349, 1977.

88. Nelson, M. D., K. E. Nygard, J. H. Griffin and W. E. Shreve, "Implementation Techniques for the Vehicle Routing Problem", *Computers and Operations Research*, Vol. 12, 273–283, 1988.

89. Daskin, M., *Private Communication.*

90. Martello, S. and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, New York, 1990.

91. Altınel, İ. K. and T. Öncan. *A New Enhancement of the Clarke and Wright Savings Heuristic for the Capacitated Vehicle Routing Problem.* Research Paper Series No.FBE-IE-02/2003-02. Department of Industrial Engineering, Boğaziçi University, İstanbul, Türkiye, 2003.

92. Christofides, N. and S. Eilon, "An Algorithm for the Vehicle Dispatching Problem", *Operational Research Quarterly*, Vol. 20, pp. 309–318, 1969.

93. Augerat, P., J. Belenguer, E. Benevent, A. Corberan, D. Naddef and G. Rinaldi, 1995, *Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem*, Technical Report RR 949-M, Universite Joseph Fourier, Grenoble, 1995.

94. http://www.branchandcut.org, 2004.

95. http://neo.lcc.uma.es/radi-aeb/WebVRP, 2004.

96. Camerini, P. M., G. Galbiati and F. Maffioli, "Complexity of Spanning Trees Problems: Part I", *European Journal of Operational Research*, Vol. 5, pp. 346-352, 1980.

97. Camerini, P. M., G. Galbiati and F. Maffioli, "On the Complexity of Finding Multi–Constrained Spanning Trees", *Discrete Applied Mathematics*, Vol. 5, pp. 39–50, 1983.

98. Hall, L., "Experience with a Cutting Plane Algorithm for the Capacitated Spanning Tree Problem", *INFORMS Journal on Computing*, Vol. 8, pp. 219–234, 1996.

99. Amberg, A., W. Domschke and S. Voβ, "Capacitated Minimum Spanning Trees: Algorithms Using Intelligent Search", *Combinatorial Optimization: Theory and Practice*, Vol. 1, pp. 9–39, 1996.

100. Sharaiha, Y., M. Gendreau, G. Laporte and I. Osman, "A Tabu Search Algorithm for the Capacitated Shortest Spanning Tree Problem", *Networks*, Vol. 29, pp. 161–171, 1997.

101. Ahuja, R. K., J. B. Orlin and D. Sharma, "Multi-exchange Neighborhood Structures for the Capacitated Minimum Spanning Tree Problem", *Mathematical Programming Series A*, Vol. 91, pp. 71–97, 2001.

102. Ahuja, R. K., J. B. Orlin and D. Sharma, "A Composite Very Large-scale Neighborhood Structure for the Capacitated Minimum Spanning Tree Problem", *Operations Research Letters*, Vol. 31, 185–194, 2003.

103. Patterson, R., E. Rolland and H. Pirkul, "A Memory Adaptive Reasoning Technique for Solving the Capacitated Minimum Spanning Tree Problem", *Journal of Heuristics*, Vol. 5, pp. 159–180, 1999.

104. Patterson, R. and H. Pirkul, "Heuristic Procedure Neural Networks for the CMST problem", *Computers and Operations Research*, Vol. 27, pp. 1171–1200, 2000.

105. Rolland, E., R. Patterson and H. Pirkul, "Memory Adaptive Reasoning & Greedy

Assignment Techniques for the Capacitated Minimum Spanning Tree Problem", in S. Voss, S. Martello, I. Osman and C. Roucairol (eds.) *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pp. 487–498, Kluwer Academic Publishers, Norwell, MA, 1999.

106. Craig, G., M. Krishnamoorthy and M. Palaniswami, "Comparison of Heuristic Algorithms for the Degree Constrained Minimum Spanning Tree", in I. H. Osman and J. P. Kelly (eds.), *Metaheuristics: Theory and Applications*, pp. 83–96, Kluwer, Boston, 1996.

107. Feo, T. and M. Resende, "Greedy Randomized Adaptive Search Procedures", *Journal of Global Optimization*, Vol. 6, pp. 109–133, 1995.

108. Souza, M. C., C. Duhamel and C. S. Ribeiro, "A GRASP Heuristic for the Capacitated Minimum Spanning Tree Problem Using a Memory–Based Local Search Strategy", in M. G. C. Resende and J. Souza, (eds.), *Metaheuristics: Computer Decision-Making*, Kluwer, 2002.

109. Jothi, R. and B. Raghavachari, "Design of Local Access Networks", *Proc. 15th IASTED International Conference on Parallel and Distributed Computing Systems (PDCS)*, pp. 883–888, 2003.

110. Jothi, R. and B. Raghavachari, "Revisiting Esau-Williams' Algorithm: On the Design of Local Access Networks", submitted to Telecommunication Systems, 2004.

111. Karnaugh, M., "A New Class of Algorithms for Multipoint Network Optimization", *IEEE Transactions on Communications*, Vol. 24, pp. 500–505, 1976.

112. Kershenbaum, A., R. Boorstyn and R. Oppenheim," Second-Order Greedy Algorithms for Centralized Teleprocessing Network Design", *IEEE Transactions on Communications*, Vol. 28, pp. 1835–1838, 1980.

113. Kershenbaum, A. and W. Chou, "A Unified Algorithm for Designing Multidrop

Teleprocessing Networks", *IEEE Transactions on Communications*, Vol. 22, pp. 1762–1772, 1974.

114. Dai, H. and S. Fujino, "On Designing Constrained Local Access Networks", *Proceedings of the Fourth International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 167–176, IEEE Computer Society, 2000.

115. Bruno, G. and G. Laporte, "A Simple Enhancement of the Esau-Williams Heuristic for the Capacitated Minimum Spanning Tree Problem", *Journal of the Operational Research Society*, Vol. 53, pp. 583–586, 2002.

116. http://www.ms.ic.ac.uk/info.html, 2004.