Workforce scheduling and planning : a combinatorial approach

Workforce Scheduling and Planning: A Combinatorial Approach

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op dinsdag 24 april 2012 om 16.00 uur

door

Murat Fırat

geboren te Istanbul, Turkije

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ing. G.J. Woeginger

Copromotor:
dr.ir. C.A.J. Hurkens

A catalogue record is available from the
Eindhoven University of Technology Library

ISBN: 978-90-386-3126-4

Acknowledgements

The history of this work goes back to 2007. It was a sunny day of June in Istanbul. I saw an open position at the Eindhoven University of Technology and I contacted Cor Hurkens, since I wanted to work on scheduling and the project title was "scheduling tasks with skill requirements". I would like to thank Cor Hurkens for accepting my application for this position and for letting me join to the Combinatorial Optimization group at the TU Eindhoven. During my Ph.D. study, Cor was always like a friend with me rather than a supervisor. Sometimes we enjoyed solving our problems and sometimes we had long and non-converging discussions. They all will stay as nice memories in my mind. Moreover, all our travels for conferences and workshops were amazing. Thanks to Cor Hurkens for everything.

My special thanks go to Gerhard Woeginger. Besides his broad academic knowledge, he has a great personality. I always enjoyed getting a piece of advice in each of our conversations. I will remember all of them throughout my life. I also enjoyed very much talking about diverse subjects during the lunch breaks with Gerhard. He helped me a lot in building up my vision on academic life.

It was a pleasure for me to assist Jan Karel Lenstra in one of his courses. My special thanks go to him for participating my graduation committee. I would like to thank Rudi Pendavingh and Judith Keijsper for proofreading of my papers and thesis. They are very good colleagues and it was nice to hear their helpful ideas and comments after my talks. I thank Johann Hurink, my graduation committee member, for carefully reading my manuscript and I appreciate his suggestions for the improvement of this thesis.

Someone to whom I owe many thanks: Alexandre Laugier. His guidance and his patience made my working environment more productive. I had the opportunity to attend France Telecom seminars several times. These seminars took place in "Alexandre's village" where we all enjoyed the nature and the history of more than one thousand years. I always had fun during our conversations and I learned a lot from Alexandre. I also thank Anne Marie Bustos for being hospitable during my visits to France.

Bana her an ve her yerde sevgi ve desteklerini esirgemenyen ve benim ilk öğretmenlerim olan babam Sadettin Fırat'a, annem Beşire Fırat'a, dünyada en çok güvendiğim doktor olan ablam Nil Fırat'a, her zaman yanımda olan ve beni destekleyen güzeller güzeli biricik eşim Pınar Çelebi Fırat'a teşekkür ediyorum.

I would like thank my parents Sadettin Fırat and Beşire Fırat for being my first teachers in my life and for supporting me always, my sister Nil Fırat who is the most dependable doctor in the world for me, my wife Pınar Çelebi Fırat for supporting me

against all difficulties of life and for being always with me.

I had great colleagues in Eindhoven: John van den Broek, Nikhil Bansal, Leen Stougie, Jan Draisma, Christian Eggermont, Stefan van Zwam, Maciej Modelski, Hein van der Holst, Peter Korteweg, Rene Sitters, and many others. Special thanks to Kora Lemmens and Harma Koops for their precious helps. I had enjoyable coffee breaks with John van den Broek and he was a very good roommate. I really enjoyed our conversations with Christian almost on everything.

I would like to thank Yorgo Istefanopulos for encouraging me in starting my master program and Ümit Bilge for supervising my master thesis and for her support during my assistance in the BUFAIM Laboratory.

Murat Fırat,
Eindhoven
April 2012

Contents

Chapter 1

Introduction

This thesis investigates solution methodologies for concrete combinatorial problems in *scheduling* and *planning*. In all considered problems, it is assumed that the available information does not change over time; hence these problems have a deterministic structure.

The problems studied in this thesis are divided into two groups; "workforce scheduling" and "planning". In workforce scheduling, the center problem is to build a schedule of *tasks* and *technicians*. It is assumed that the time line is split into workdays. In every workday, tasks must be grouped as *sequences*, each being performed by a *team* of technicians. *Skill requirements* of every task in a sequence must be met by the assigned team. This scheduling problem with some other aspects is difficult to solve quickly and efficiently.

A workday schedule of the aforementioned problem includes many-to-one type workforce assignments. As the second problem in workforce scheduling, *stability* of these workforce assignments is investigated. The stability definition of *Gale-Shapley* on the marriage model is extended to the setting of multi-skill workforce assignments.

Generally, the scheduling problems and the assignment problems are highly involved in the area of project management. For example, in a telecommunication company there may be thousands of activities and thousands of technicians to schedule. In this scheduling task, the problem may be so complicated that maintaining efficiency in these schedules is almost impossible without dividing it into subproblems. Then combinatorial optimization comes into the game and provides useful tools to maintain the control of quality in such a project management and to achieve improvements.

In the second problem group, two problems related to vehicle planning are studied; the "dial a ride problem" and the "vehicle refueling problem". In the former, the goal is to check whether a list of pick-up and delivery tasks can be achieved under several timing constraints. In the latter, the goal is to make refueling decisions to reach a destination such that the cost of the travel is minimized.

1.1 Preliminaries

This section explains some basic topics in combinatorial optimization such as algorithms, computational complexity, graphs, and networks.

Combinatorial Optimization. Combinatorial optimization is a branch of mathematics that analyses countable discrete structures and finds the ones meeting some criteria. In combinatorial optimization, a problem asks for "an optimum object in a finite collection of objects"[1] . Every problem in combinatorial optimization has a set of *instances*. Every instance of a problem has a set of *feasible solutions*, each having an *objective value*. The goal or the *objective* of a problem is to find a solution that is feasible and has either *maximum* or *minimum* objective value. Such solutions are called *optimum* solutions. Maximization (minimization) problems look for feasible solutions with maximum (minimum) objective values.

1.1.1 Algorithms

Algorithms. An algorithm is a sequence of well-defined computational steps. It takes an input and transforms it into a desired output. An algorithm is said to *solve* a given combinatorial optimization problem if, for each instance of the problem, it will output an optimal solution or report that no solution exists in finitely many steps. The time complexity of an algorithm is the number of steps needed to solve the problem instance as a function of the instance size.

Computational complexity. In computer science, problem instances are represented as strings over an alphabet. For example, the set $\{0, 1\}$ is a binary alphabet and the problems are encoded by using this basic binary alphabet. The input size of an instance is generally denoted by n. An algorithm may perform different number of operations for different instances of a problem. The *running time* of an algorithm is the maximum number of operations to find a solution over the whole set of instances. This is called the *worst-case* performance and algorithms are classified by their worst case performances.

Asymptotic notation is used to express the running time of algorithms. It is useful for two reasons. First, it simplifies the expression of running times. Second, it is useful to show how an algorithm responds to changes in the input size. Below, two function sets of the asymptotic notation are defined.

$$O(g(n)) = \{f(n) : \exists c > 0, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$$

$$\Omega(g(n)) = \{f(n) : \exists c > 0, n_0 > 0 \text{ such that } 0 \leq cg(n) \leq f(n), \forall n \geq n_0\}$$

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

If $f(n) \in O(g(n))$, it is written $f(n) = O(g(n))$. An algorithm is said to be *efficient* or *polynomial-time*, if it solves a problem with a running time that is bounded by a polynomial function.

1.1.2 Computational complexity theory

Although the researchers have made a huge effort for many decades, no efficient algorithm could be found for many computational problems. Several researchers like

[1] This is the first sentence of Alexander Schrijver in his book "Combinatorial Optimization".

Edmonds (1962) studied how to relate these hard problems in terms of their difficulty. Following these attempts, "computational complexity theory" was built up by Cook (1971) and Karp (1972).

In computational complexity theory, every optimization problem has a corresponding *decision problem* whose answer is either YES or NO. A decision problem points a particular solution or a particular set of solutions and asks whether this particular solution (set) exists or not. In fact, a decision problem is merely a rephrasing of the corresponding optimization problem. A formal definition of a decision problem includes two parts. Firstly, the sets, parameters, and related information is given. Secondly, the existence of the particular solutions is asked. Below, a specific problem in single-machine scheduling as an example.

> Problem: Weighted sequencing on single-machine (WSSM)
>
> Instance: Given a number k and a set J of jobs. Processing a job $j \in J$ takes $p_j \in \mathbb{R}_+$ time units on a single machine, and it has a weight $w_j \in \mathbb{R}_+$.
>
> Question: Does there exist a permutation π of the jobs in J such that $\sum_j w_j C_j$ is no more than k where $C_j = \sum_{\pi_{j'} \leq \pi_j} p_{j'}$?

Every decision problem that is a rephrasing of an optimization problem has a numerical bound, and in the above example, this is k. A decision version of an optimization problem with maximization (minimization) objective asks whether a solution value exists with "at least" ("no more than") the numerical bound. In the above example, the objective of the corresponding optimization problem is minimizing $\sum_j w_j C_j$. If a decision problem is solved by a polynomial-time algorithm, so is the corresponding optimization problem: A binary search would be enough to find the optimal solution with the answers of the decision problem.

Decision problems that can be solved by efficient (polynomial-time) algorithms form the complexity class P. For example, the WSSM is in P, since the algorithm called *Smith's rule* proposed by Smith (1956), solves it in polynomial time to optimality.

In complexity theory, a *certificate* of a decision problem refers to any information that enables us to construct a solution. The fundamental complexity class NP includes those decision problems whose YES instances and NO instances can be distinguished with a given certificate in *polynomial time*. Instances of decision problems in P can be identified, as YES or NO instance, by solving them with polynomial-time algorithms. Therefore, we do not even need a certificate for these problems. By this argument, it is easy to see that P is a subset of NP.

The class NP also includes many problems for which no polynomial-time algorithms are known. This point raises the well-known open question "Is P equal to NP?". There is a common belief among the researchers that it is unlikely that the answer to this question is yes.

A major breakthrough in complexity theory is achieved by Stephen A. Cook. Cook (1971) proved that every problem in NP can be reduced to one particular problem called the *satisfiability problem*, shortly the SAT problem. It is defined as

> Problem: Satisfiability (SAT)

INSTANCE: Given a set U of boolean variables and a collection C of clauses over U. A literal is either a boolean variable or a negation of a boolean variable and a clause is a disjunction of literals.

QUESTION: Is there a satisfying truth assignment for C?

A decision problem is *NP-complete* if and only if it is in NP and it is as difficult as any problem in NP. Cook's result enabled researchers to prove NP-completeness of many problems by reducing the SAT to these problems instead of showing that each of them is as difficult as any problem in NP. In the last decades, many problems have been shown to be NP-complete, hence proving NP-completeness of a problem has turned out to find a particular NP-complete problem with a similar structure and reducing the latter to the former. A comprehensive analysis of NP-complete problems is given by Garey and Johnson (1979). *NP-hardness* generalizes NP-completeness in the sense that if an NP-complete problem can be reduced to a problem, that problem is said to be NP-hard without the requirement that it is in NP.

For example, let problem A and problem B be two problems in NP. A reduction from A to B maps all instances of A to some instances of B in "polynomial time" such that any YES instance of A corresponds to a YES instance of B and any NO instance of A corresponds to a NO instance of B. Then we say that "A is reduced (or transforms) to B", and we write "A $\propto$ B". If A $\propto$ B, then B is at least as difficult as A. This leads to the following cases; (1) if B is in P, so is A, and (2) if A is NP-complete, so is B.

Next, we define a specific scheduling problem as follows:

PROBLEM: SEQUENCING ON SINGLE-MACHINE (SSM)

INSTANCE: Given a number k and a set J of jobs. Processing a job $j \in J$ takes $p_j \in \mathbb{R}_+$ time on a single machine.

QUESTION: Does there exist a permutation π of the jobs in J such that $\sum_j C_j$ is no more than k where $C_j = \sum_{\pi_{j'} \leq \pi_j} p_{j'}$?

Note that we can transform any SSM instance to a WSSM instance by defining $w_j = 1$ for every $j \in J$ in polynomial time. Hence, SMM $\propto$ WSSM and SSM is in P as well.

Algorithms revisited. The algorithms that guarantee optimality of solutions are called *exact* algorithms. Exact algorithms for NP-complete problems do not run in polynomial time, otherwise it would result in P=NP. The survey of Woeginger (2001) summarizes basic techniques used in exact algorithms for NP-complete problems. *Approximation* algorithms that do not ensure optimality but in general they find near optimal solutions for NP-complete problems. Approximation algorithms have provable solution quality that tells us a priori how the solutions may deviate at most relative to the optimal solution.

Algorithms based on educated techniques like intuition, experience, and rule of thumb are called *heuristics*. They have conceptual simplicity. However, heuristic algorithms cannot guarantee solution quality in general. Some easy problems in P

can be solved by priority rule-based heuristics to optimality. As the difficulty of problems increases, heuristics may even fail to find any feasible solution if the only solutions are in the states they chose not to try.

1.1.3 Graphs and networks

A graph is a mathematical abstraction that is determined by a set of points and a set of edges, each joining a pair of points. Many real world problems can be described by means of graphs. For example, the points (also called vertices) may represent warehouses and customers, and a line joining a warehouse-customer pair may represent a transportation with a certain capacity. The graph $G = (V, E)$ has two sets; the set V of vertices and the set E of edges. An edge $\{u, v\} \in E$ has the endpoints, vertices, u and v. We say that vertex u, also v, is *incident* on edge $\{u, v\}$ and the edge $\{u, v\}$ incident on its endpoints, u and v. The vertices are *adjacent* to each other, if they are joined by edge, and two edges are adjacent if they have a common endpoint. Two edges with no common endpoint are said to be *disjoint*. In a complete graph every vertex pair is joined by an edge.

In graph theory context, a *matching* is a set of edges, each pair in which is disjoint. The problem of finding a maximum-size matching in general graphs can be solved by the well-known *blossom algorithm* of Edmonds (1965) in polynomial time, hence maximum matching problem is in P. A graph $G = (V, E)$ is a *bipartite graph* if its vertices can be divided into two disjoint sets so-called *partitions*. So the graph G with partitions $X \subset V$ and $Y \subset V$ is denoted by $G = (X, Y, E)$. The bipartiteness of a graph can be tested by the *breadth first search* (BFS) algorithm in polynomial time and the *augmenting path* algorithm finds a matching with minimum size on a bipartite graph. Many real-world problems can be solved by creating a convenient bipartite graph and finding a matching with minimum size on it. The marriage problem and the assignment problem are two of these problems.

A graph is *simple* if and only if, any vertex is not joined by an edge to itself and any vertex pair is joined by at most one edge. A *path* is an alternating sequence of vertices and edges in which no edge and no vertex is repeated. Usually, every edge has a weight in path problems and the "shortest path problem" asks for a path with minimum weight from one vertex to another. It is formally defined as

> Problem: Single pair shortest path problem (SPSP)
>
> Instance: Given a number k and a graph $G = (V, E)$ with a length function $w : E \to \mathbb{R}$. We distinguish two vertices $v_0, v_k \in V$ in order to find a path from the former to the latter.
>
> Question: Does there exist a path $p = \{v_0, \ldots, v_k\}$ such that $\sum_{e \in p} w(e)$ is no more than k?

Bellman-Ford algorithm solves the SPSP in polynomial time. Moreover, if all edges have nonnegative weights, *Dijkstra's* algorithm also solves the SPSP in polynomial time (see Chapter 24 of Cormen et al. (2009)).

In *interval* graphs, every vertex corresponds to an interval on the real line and any two vertices are joined by an edge if the corresponding intervals intersect and vice versa.

Graphs in which the edges are ordered pairs of vertices are called *digraphs*. The edges in digraphs are directed and they are called *arcs*. The *tail* of an arcs is its starting vertex, and the *head* is its ending vertex. A digraph is usually denoted by $D = (V, A)$ and an arc $a \in A$ is expressed by $[u, v]$ where u and v are the tail and the head. A digraph with weighted edges and/or vertices in the context of graph theory is called a *network*. The weights may represent capacity, cost, and so on.

Transportation networks are used to model the *flow* of commodity, information, or traffic from a special vertices called *sources* to a special vertices called *sinks*. A sink is a vertex at which there is some demand, and demands of sinks in the network are satisfied by the sources. The vertices between the sources and the sinks are called *intermediate vertices*. At an intermediate vertex, the flow is said to be *conserved* if and only if the outgoing flow is equal to the incoming flow. The flow through an arc is usually limited by a capacity and it may have a cost. We say that a flow is *feasible* if and only if it is conserved at intermediate vertices, it respect the capacities on arcs, and the total demand is satisfied. In flow problems, the objective is usually maximizing the flow amount and/or minimizing the flow cost. The book of Ahuja et al. (1993) gives an extensive analysis of network flow problems.

1.2 Algorithmic approaches

1.2.1 Greedy algorithms

Greedy algorithms are heuristic methods. Although it is *not* true in general, some problems can be solved optimally by greedy algorithms. These problems exhibit the following property.

Greedy choice property. A solution is built up by a greedy algorithm step by step. In each step, a locally optimal choice is made with the hope of finding a globally optimal solution. The current choice is the best one at the moment and it depends on the previous choices but not on any future choice. Once a choice is made, it is never reconsidered later. The problem is reduced to a smaller one by iteratively making greedy choices.

If it can be shown that a greedy algorithm gives the optimal solution of a given problem, it is usually preferred due to shorter running time compared to other solution methods. For example, the shortest path problem and the minimum spanning tree problem can be solved in polynomial time by greedy approach. Cormen et al. (2009) provide an extensive list of problems solved by greedy algorithms and a comprehensive analysis.

Caro et al. (1996) study the recognition of greedy structures of certain combinatorial problems. These are mainly graph related problems. The authors describe a special case of the SAT problem that can be solved greedily.

1.2.2 Linear programming

An optimization problem is called a *Linear Programming* (LP) if the objective and the constraints are expressed with linear functions. In an LP model, strict inequalities are not allowed in the formulation. An example of a linear program in standard form is given by

$$\text{maximize } c^T x \tag{1.1}$$
$$\text{subject to } Ax \leq b \tag{1.2}$$
$$x \geq 0 \tag{1.3}$$

where the vector $x \in \mathbb{R}^n$ represents the variables. The objective in (1.1) is maximization and the vector $c \in \mathbb{R}^n$ is the coefficient vector. The linear constraints are expressed in (1.2) and (1.3). The constraints in (1.2) are the so-called *main constraints*. The matrix $A \in \mathbb{R}^{m \times n}$ specifies the coefficients and the vector $b \in \mathbb{R}^m$ is called the *right-hand side* that represents the bound for restrictions. The special constraints in (1.3) are called *nonnegativity constraints*. Linear programs can be solved in polynomial time (Khachiyan (1979)), and hence belong to the complexity class P.

Due to their discrete nature, many combinatorial optimization problems can only be formulated with integrally valued decision variables. A formulation with linear objective as in (1.1), linear constraints as in (1.2), and integrally valued variables, $x \in Z^n$, is called an *integer programming* (IP). If a formulation consists of both continuous and integral variables, it is called a *mixed integer programming* (MIP).

Theorem 1.2.1. *Integer programming is NP-hard.*

Proof. We give a polynomial time reduction from the SAT in conjunctive normal form to 0-1 Integer Programming that is a special case of Integer Programming. Given a SAT instance, for every variable in set U, we create a binary variable. For every clause in set C, we define a constraint in the following way: Let $c \in C$ be a clause with a set L_c of literals and let the set $L'_c \subseteq L_c$ be the literals that are negations of boolean variables. The constraint that corresponds to the clause c turns to be $\sum_{i \in L_c \setminus L'_c} x_i + \sum_{i \in L'_c} (1 - x_i) \geq 1$. Note that a negated literal a boolean variable corresponds to the complement of the corresponding binary variable.

We see that a solution of the SAT problem satisfies every clause which implies that there is at least one true literal in every clause, hence every constraint in binary program is satisfied. This leads to a solution of the corresponding binary program. Conversely, by construction, any binary program solution gives a solution to the SAT problem as well. Consequently, Integer Programming is NP-hard, since finding its solution is as difficult as finding a SAT solution. □

The details of the proof showing that the integer programming is in NP can be found in the book of Papadimitriou and Steiglitz (1982).

Corollary 1.2.2. *Mixed integer programming is NP-hard.*

Proof. Integer Programming is a special case of Mixed Integer Programming. □

Duality. Every LP problem is associated with a particular LP problem that is called its *dual*. Then the original LP is called the *primal*. An LP and its dual can be thought of as the complements of each other for two reasons. First, they have opposite objectives (maximizing vs. minimizing or vice versa). Second, objective values of all dual solutions are either greater than equal to (in case primal has maximization objective) or less than or equal to (in case primal has minimization objective) the objective values of all primal solutions. This is called *weak duality*. Optimal solutions of both problems have the same objective values and this is called *strong duality*. We refer to the Chapter 4 of Bertsimas and Tsitsiklis (1997) for more information about the duality theory for LP models, and to the Chapter 5 of Bertsekas (2003) for the duality theory in convex programming.

The dual of the LP problem in (1.1)-(1.3) is given by

$$\begin{aligned} &\text{minimize} && b^T y && (1.4)\\ &\text{subject to} && A^T y \geq c && (1.5)\\ & && y \geq 0 && (1.6) \end{aligned}$$

In fact, the dual problem may be helpful in solving and analyzing the primal problem and vice versa. For example, verifying that the solutions found by an algorithm are indeed optimal, can be done by showing that for any solution found by that algorithm there is a dual solution with the same objective value. The dual problem may also be used to develop algorithms. Depending on the nature of the problem under consideration, decisions while solving the primal problem can be guided by dual objective and/or dual feasibility. The *primal-dual* algorithms use the guidance of dual objective while taking care of primal objective to reach optimality. Primal-dual approach is also useful in developing approximation algorithms (see Chapter 1 of Williamson and Shmoys (2011)).

Complementary slackness. As mentioned before, the objective values of the primal solutions and the dual solutions are equal only at optimality. Complementary slackness condition states two relations between primal and dual: (1) if a constraint is *tightly* satisfied in an optimal solution of the primal problem, the corresponding dual variable can take positive value in the optimal solution of the dual problem, and (2) if a constraint is "not" tightly satisfied in an optimal solution of the primal problem, the corresponding dual variable must be zero in the optimal solution of dual problem.

Let a_{ij} be the entry of matrix A in ith row and in jth column, and let $\Upsilon \subseteq \mathbb{R}^n \times \mathbb{R}^m$ such that for every $\langle x^*, y^* \rangle \in \Upsilon$ we have

$$x^* = \text{Argmin}\{c^T x : Ax \leq b, x \geq 0\}, y^* = \text{Argmax}\{b^T y : A^T y \geq c, y \geq 0\} \qquad (1.7)$$

Then the complementary slackness condition is formally stated, without proof, by the following theorem

Theorem 1.2.3. *Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be feasible primal and dual solutions respectively. One has $\langle x, y \rangle \in \Upsilon$, if and only if the following conditions are satisfied:*

$$(b_i - \sum_{j=1}^{n} a_{ij}x_j)y_i = 0, \quad i = 1, \ldots, m. \tag{1.8}$$

$$(\sum_{i=1}^{m} a_{ij}y_i - c_j)x_j = 0, \quad j = 1, \ldots, n. \tag{1.9}$$

We note that the proof of the above theorem can be found in many optimization books; see for example Bertsimas and Tsitsiklis (1997).

1.3 Scheduling

The process of making decisions about how to perform certain tasks while respecting given constraints is called *scheduling*. In the scheduling context, a *sequence* denotes an ordering of tasks. Solutions of many scheduling problems involve tasks sequences. Tasks may be production processes of machines in manufacturing, take-offs and landings at airports, operations in construction projects, and executions of processing units in computer environments. The constraints to be respected may be precedence relations, duration bounds, earliest possible start times, due dates, and skill requirements. Scheduling is rooted in the work of Henry Gantt which goes back to the 1910s. He developed the well-know Gantt charts that are still used by modern scheduling softwares.

In scheduling, we see that first studies in the literature belong to Smith (1956), Johnson (1954) and Jackson (1955). After the development of computational complexity theory, many researchers worked on exploring the complexity hierarchy of scheduling problems. For example, Lawler (1978), Lenstra and Rinnooy Kan (1978), and Lenstra and Rinnooy Kan (1980) studied the complexity of basic deterministic scheduling problems. A comprehensive complexity hierarchy of scheduling problems is given by Brucker and Knust (2009).

It has been shown that many scheduling problems are NP-Hard. This directed researchers, like Hochbaum and Shmoys (1988), Lenstra et al. (1990), and Schuurman and Woeginger (2002), to develop approximation algorithms. For some scheduling problems, not only optimality, but also certain solution quality cannot be guaranteed a priori with polynomial time algorithms. This property is called *inapproximability*. Kellerer et al. (1996), Hoogeveen et al. (2001), and Woeginger (2004) are some of the researchers who studied inapproximability of scheduling problems.

1.3.1 Machine scheduling

In machine scheduling, jobs are processed by using available *resources* or so-called machines. In *single machine* scheduling problems, there is one resource with a capacity of performing one job at a time. These problems are important in the sense that they appear as sub-problems in complex scheduling problems. For example, the WSSM in Section 1.5.1 is a single machine scheduling problem. Many single machine scheduling

problems can be solved in polynomial time. As additional constraints are considered, they become NP-complete.

In *parallel machine* scheduling, there are several machines, each may have different processing speeds. Jobs consist of several parts so-called *operations* in many scheduling problems. *Flow shop scheduling* problems consist of several types of machines such that each job has exactly one operation for every machine and all jobs go through all the machines in the same order.

If every job has a fixed order of operations and if this order may vary from one job to another, the scheduling model is called a *job shop*. In job shops, the jobs may visit the machines more than once and the number of operations may vary from one job to another. In an *open shop*, the jobs do not have a fixed order for operations.

Graham et al. (1979) introduced the notation $\alpha|\beta|\gamma$ for the classification of models in machine scheduling. In this notation, the fields α, β, and γ represent the machine environment, the job characteristics, and the criterion to optimize respectively. For example, the WSSM and the SSM are denoted by $1||\sum w_j C_j$ and $1||\sum C_j$ in this notation. Brucker (2007) provides an extensive collection of numerous machine scheduling problems. The theory and practice of machine scheduling is covered by Pinedo (2008).

1.3.2 The resource-constrained project scheduling

In the resource-constrained project scheduling, shortly the RCPSP, a set of activities is involved in a project. There are several types of resources and in order to finish the project, available resources must be allocated to activities in required amounts. The RCPSP generalizes machine scheduling models in the sense that the resources can be used partially to perform activities. The resources are renewable which means that they are available for every time unit in constant amount. The *makespan* refers to the latest completion time of all activities in the project and usually the objective is minimizing the makespan. The RCPSP has become a central problem in project scheduling literature and many real-world complex scheduling problems are defined using RCPSP's setting. Brucker et al. (1999) and Hartmann and Briskorn (2010) give extensive literature surveys on the RCPSP.

1.3.3 Workforce scheduling

Multi-skill technician-task scheduling (MTTSP). Similar to the RCPSP, a set of tasks is given in the MTTSP. As a generalization of resource constraints, there are two types of resources; renewable resources and non-renewable resources. Renewable resources are technicians and they have skills in several specialization fields. The degree of expertise in a specialization field is expressed by *hierarchical* levels. There is a project budget that can be used to hire external companies for outsourcing some tasks. This budget may be considered as non-renewable resource. Performing a task requires the availability of a certain skill combination and usually this requirement is met by a team of technicians. In the problem, capacities of renewable resources vary with time, since the availability of technicians is considered.

Both the RCPSP and the MTTSP in the most general form are NP-hard. In the literature, many heuristic algorithms are proposed for these problems and the growth of computational power in the latest years made it possible to obtain good quality solutions.

Scheduling problem of France Telecom. One of the topics studied in this thesis is a MTTSP with some restrictions. In order to make this scheduling problem known among researchers, France Telecom introduced it as the subject of the challenge organized by the French Operational Research and Decision Support Society (ROADEF) in 2007 (Dutot et al. (2006)).

In the problem, tasks are grouped into *priority classes*, each representing an urgency level. It is assumed that the time line is split into periods as workdays. If the processing of a task starts in a workday, it must be completed in the same workday. The technicians working in a team must stay together throughout the workday; hence a team can perform a number of tasks as long as (1) the total length does not exceed a workday, and (2) skills in the team meet the requirements of each task. The objective is minimizing the weighted sum of the latest completion times of the priority classes.

Outsourcing some tasks is possible by using a fixed budget. Since the budget amount does not appear in the objective of the problem, it does not concern us that any amount of money is saved from this budget. Therefore, in all solutions of the problem, this budget is used as much as possible. Here the challenge is to figure out which subset of the tasks should be outsourced such that the remaining tasks can be performed in shortest time by the available technician group.

1.4 The stable assignment problem

1.4.1 The stable marriage problem

The concept of *stability* was introduced by David Gale in the beginning of the 1960s. Gale and Shapley (1962) defined stability in the setting of the marriage problem. In this setting, there are two sets of players with equal sizes; *men* and *women*. Each player orders strictly *all* players in opposite sex in a so-called *complete* preference list. A marriage is *stable* if and only if it *does not* contain such a man-woman pair that they are not partners and prefer each other to their current partners. It is said that such a pair *blocks* the marriage and makes it *unstable*.

If we consider a complete bipartite graph whose partitions are the player sets, then the *marriage* problem is equivalent to the *perfect matching* problem subject to the *stability* condition. In marriage problems, it is common to use an objective of maximizing the satisfaction of one particular player set. In a *man-optimal* stable marriage, every man is as happy as in any stable marriage. Woman-optimal is defined similarly. Man-optimal and woman-optimal stable marriages can be constructed in polynomial time by the *proposal-disposal* algorithm of Gale and Shapley (1962).

In the following theorem, we give one of the fundamental results of Gale and Shapley (1962).

Theorem 1.4.1. *(Gale and Shapley (1962)) There always exists a stable set of marriages.*

Theorem 1.4.1 tells us that for any preference of players, stable marriages can be constructed.

1.4.2 The university admissions problem

Our stability analysis in Chapter 3 includes a reduction of a workforce assignment problem to the university admissions problem. In the *University Admissions* problem, players, that are *universities* and *students*, have usually *incomplete* preference lists over the opposite player set. A university can admit a number of students at most its *quota*. A *stable admission* does not contain a university-student pair such that the following cases are satisfied simultaneously:

- they are not assigned to each other,
- the student is not admitted or he/she prefers that university to his/her current university,
- the university has not filled its quota and it can immediately admit the student, or its quota is filled and there exists a student who is admitted but is preferred less than that student.

A fundamental result in the university admissions problem is obtained by Gale and Sotomayor (1985) and it is given in the following theorem.

Theorem 1.4.2. *(Gale and Sotomayor (1985)) In all stable admissions; the same subset of students is admitted, and every university admits the same number of students.*

Gale and Sotomayor (1985) also showed that stable admissions, if one exists, can be constructed in polynomial time. Baïou and Balinski (2000a) provide us an iterative approach in which a particular case causing instability is detected and prevented in every iteration. The contribution of the authors is important in the sense that this approach provides an implicit description of stable admissions by pointing out what to avoid to satisfy the stability.

Marriage problem revisited. It does not remain true that stable marriages always exist, if the preference lists are *incomplete*. If the players like some players in the opposite set *equally* or they have *ties* in their preference lists, stable marriages can be constructed in polynomial time (Irving (1994)). However, Iwama et al. (1999) proved that the stable marriage problem is NP-complete if the preference lists are incomplete and the preferences have ties. Then Iwama et al. (2004) and Iwama et al. (2008) developed approximation algorithms for the marriage problem with incomplete lists and ties.

1.4.3 The stable allocation problem

In stability terminology, marriages are one-to-one type assignments and admissions are many-to-one type assignments. Baïou and Balinski (2000b) developed a graph-theoretic approach in which the properties of many-to-many type assignments, or *allocations*, are analyzed. The authors proved the following result

Theorem 1.4.3. *(Baïou and Balinski (2000b)) There always exist many-to-many type stable assignments.*

Baïou and Balinski (2002) extended many-to-many type stable assignment problems to an allocation problem in which players in one set offer some amount of work and players in the opposite set seek a certain amount of work. The authors proved that stable allocations can be constructed in polynomial time. This generalization is indeed important for practical cases such as finding allocations for agents who can flexibly work for those companies that can outsource a certain amount of work to these agents.

Stability concepts are used in many real-world assignment applications. Some of these applications are assigning medical students to hospitals in the U.S.A., Canada, Scotland, and Japan; assigning the students to universities in Turkey, and assigning academicians to universities in France.

1.5 Planning problems

We can define *planning* as a process of setting certain goals, systematically making decisions on how to allocate the resources, and determining the tasks or actions to pursue the stated goals. The range of a plan depends on the involved actions and it may be long, intermediate, or short. Then the decisions to be made are *operational*, *tactical*, and *strategic* respectively. Proper planning is important, if it is expected to minimize the timing and the use of resources. We can give diverse examples for planning like finding the routes for car navigation, deciding which train must stay on which track of the shunting yard during the night, and rescheduling the flights of airplanes and passengers in commercial aviation when disruptions occur. In the following sections we explain two planning problems that are studied in this thesis.

1.5.1 The dial-a-ride problem

Hunsaker and Savelsbergh (2002) discussed feasibility testing for a dial-a-ride problem under maximum wait time and maximum ride time constraints. Dial-a-ride problems concern the dispatching of a vehicle to satisfy requests where an item (or a person) has to be picked up from a specific location and has to be delivered to some other specific location. In this pick up and delivery service, items are carried at the same time as long as the vehicle capacity allows. Dial-a-ride problems arise in many practical application areas, as for instance shared taxi services, courier services, and transportation of elderly and disabled persons. Next, we give a formal definition of the problem.

Problem: Dial-a-ride problem

Instance: A sequence of $2n + 1$ events has to be served in the given order by a single vehicle. The first event is the dispatch of the vehicle from a central facility. The remaining $2n$ events are grouped into a set $\mathcal{P}$ of pairs $\langle i, j\rangle$ with $i < j$ where i and j are not necessarily consecutive. In every pair $\langle i, j\rangle$, the earlier event i is the pickup and the later event j is the delivery of some fixed item. The vehicle can wait at most ω_i at the location where event i occurs. Event i must occur between α_i and β_i. Travel time between location i and $i + 1$ is $\gamma_{i,i+1}$, and for pair $\langle i, j\rangle$, the riding time between events i and j can be at most δ_{ij}.

Question: Do there exist $2n + 1$ time points for these $2n + 1$ events satisfying the aforementioned constraints?

1.5.2 The vehicle refueling problem

The vehicle refueling problem involves refueling decisions during a travel on a route with a fixed order of cities. In the travel, fuel prices vary from one city to another and the available fuel amount is limited in every city. The maximum fuel amount that the vehicle can carry while departing from a city depends on the city of departure. Therefore, tank capacity of the vehicle is not fixed throughout the travel. The distance between two consecutive cities is specified as the necessary fuel amount to travel it.

Refueling decisions should be made in a way that the vehicle should have enough fuel in every city to reach the next one and finally the destination. The goal is to reach the destination with minimum refueling cost. Vehicle refueling problem is encountered in several applications like in planning inventory of a single item and refueling the airplanes once their flight schedules are fixed. Below we give a formal definition of the problem.

Problem: Vehicle refueling problem

Instance: Given a constant k, the set $S = \{1, \ldots, n\}$ of cities, the set T of tank capacities, the set P of fuel prices, set D of distances, and set U of available fuel amounts. The city n is the destination. For any city $w \in S$, total refueling amount must be enough to reach city w, refueling amount in city w cannot be more than U_w, and the total fuel amount in tank, after refueling is done, cannot be more than T_w. x_w is the refueling amount in city w and it costs $p_w x_w$ where p_w is the fuel price in city w.

Question: Does there exist a refueling policy such that $\sum_{w\in S} p_w x_w$ is no more than k?

Theorem 1.5.1. *(Lin et al. (2007)) A special case of the vehicle refueling problem with $T_w = T$ and $U_w \equiv \infty$ for all $w \in S$ can be solved in linear time.*

The authors propose an algorithm in which refueling decisions are made in a complicated way. Every city distinguishes two cities among the successive ones: the *first cheaper* city and the *last reachable* city with full tank. In a city, if the first

cheaper city is later than the last reachable city, the tank is filled to the full capacity and the vehicle goes to next city, otherwise the refueling is done, if necessary, in an amount to reach the first cheaper city and the vehicle directly goes to the first cheaper city.

An equivalent lot sizing problem. The vehicle refueling problem is equivalent to a specific single-item lot sizing problem. The equivalence can be easily seen if the *fuel* is considered as the *item* to produce and/or to keep in the inventory such that every *city* corresponds to a *time interval* and the *distance* to travel from a certain city to the next one becomes the *demand* in the corresponding time interval. Then every *refueling* can be seen as a *production* with zero setup cost and with the linear cost function. Both production and inventory are capacitated and the capacities vary over time intervals.

The equivalent lot sizing problem has been studied by Sedeo-Noda et al. (2003) and the authors propose an $O(n \log n)$ time greedy algorithm. For an overview of lot sizing problems in general we refer to Jans and Degraeve (2008).

Vehicle refueling via network flow. The vehicle refueling problem can expressed as a flow problem on a specific network that is given by Sedeo-Noda et al. (2003). In general, network flow problems are solvable in polynomial time. Ahuja and Hochbaum (2008) developed an $O(n \log n)$ time algorithm to solve the flow problem on the specific network for the vehicle refueling problem.

1.6 Results and overview of the thesis

This section summarizes the main results obtained on the problems that are studied in this thesis.

1.6.1 Multi-skill workforce scheduling

In Chapter 2, we worked on the multi-skill workforce scheduling problem of France Telecom. In this section, we give an overview of the objectives, the contributions, and the results. The work in this chapter is published by Firat and Hurkens (2011a) in a refereed journal.

Objectives. The goal in the scheduling problem of France Telecom (defined in Section 1.3.3) is to build time-efficient schedules. Here, the challenge is to determine task sequences such that the following points should be considered simultaneously: (1) hard tasks should be packed efficiently into sequences in order not to waste the working time of experts, (2) technicians should be chosen for teams for the efficient use of skills. These two points are closely related to each other and both should be taken care of simultaneously during schedule construction. Moreover, any method with a global view of the problem is time consuming due to high complexity. Therefore, we aimed to have some *flexibility* in our solution methodology in order to repair the damage of choices that are locally advantageous but not globally.

Contributions. Our contributions in the multi-skill workforce scheduling problem

of France Telecom are listed below.

- We formulated several key properties of tasks (Section 2.5.2).
- We designed a MIP model to calculate lower bounds for the objective of the problem (Section 2.5.3).
- We developed a MIP model, called the Flexible Matching Model (FMM), to construct day schedules by extending the workloads of teams iteratively. In fact, the FMM is a MIP formulation of a many-to-one assignment problem on a bipartite graph with some additional constraints. It builds teams in a flexible way by taking care of efficient packing of hard tasks as well as maximizing the utilization of experts (Section 2.5.4).

Our solution methodology. In the schedules of the France Telecom problem, workdays are independent from each other, since no overflow of task processing is allowed and teams are formed again and again in every workday. Therefore, a schedule is formed by consecutive day schedules. The main idea of our solution methodology is to start constructing a day schedule with teams, each performing a single task. Then we extend workloads of the teams by adding more tasks and at the same time we reallocate the technicians. A day schedule is completed whenever no task can be inserted anymore.

Once a task is inserted into the day schedule, it is never removed later. Moreover, a day schedule is not changed once it is completed. Considering the mentioned points, our solution methodology can be classified as a *constructive heuristic* with greedy property.

Problem instances. In the 2007 ROADEF Challenge, 30 problem instances are provided by the France Telecom as a test bed. The instances are grouped into three sets with same size; A, B, and X. The instance set A includes *small-size* instances and the skill distribution in the technician groups can be considered as *homogenous*. In this thesis, *rare expertise* refers to heterogenous skill distribution within the technician group. The instance sets B and X include relatively *large-size* instances with rare expertise.

Performances in the ROADEF Challenge 2007. We consider three best solution approaches of the ROADEF Challenge 2007: Hurkens (2009), Cordeau et al. (2010), and Estellon et al. (2009). The percentage distance between a solution value and the lower bound value is used for comparisons. We see that in small-size instances (i.e. instances of set A) all approaches have high quality solutions. However, as the instance size grows and rare expertise is introduced, the solution qualities of Cordeau et al. (2010) and Estellon et al. (2009) decrease.

Our results. We believe that the instance sets B and X are better representatives for the real life. Therefore, the conclusions are drawn on our computational results for these instances (see Table 2.7 for more details). We use the percentage distance between a solution value and the lower bound value for comparisons.

- In *all* instances of X, our solutions values are better (here lower) than all other solution approaches.

- In instance sets B and X, our solution values are around 5% than the solution values of the winner, Hurkens (2009), and around 10% lower than the solution values of Cordeau et al. (2010), and Estellon et al. (2009) on the average.

- We have best solution values in 17 instances out of 20 instances of sets B and X and in the remaining 3 instances we are not the worst. Therefore, we have a stable solution quality.

1.6.2 Stabile multi-skill workforce assignments

In Chapter 3, we worked on stable multi-skill workforce assignments. This section gives a summary of this study. The work in this chapter is submitted by Firat et al. (2011c) and revised for publication in a refereed journal.

Motivation. Every workday schedule in the solutions of the France Telecom problem is a workforce assignment between technicians and sequences of tasks. In this chapter, we perceive a *job* as a *sequence of tasks*. Then a workday schedule turns out to be a many-to-one type assignment between technicians and jobs under skill requirement constraints. *Since the solution value of the schedule remains the same for every feasible workforce assignment in a workday, we can consider preferences of technicians and jobs*. From the technicians' point of view, some jobs may be more preferred due to several reasons such as skill match, job location, job provider and so on. On the other hand, from the job providers' point of view, there are several criteria to distinguish technicians from each other like experience, age, adaptability for teamwork and so on.

Now the following question comes to mind: "Which of the feasible assignments are better if the preferences of technicians and jobs are considered?".

Objectives. The above question addresses *Gale-Shapley stability*. To the best of our knowledge, Gale-Shapley stability has not been considered within the multi-skill workforce scheduling in the literature. In our analysis, we extend the notion of blocking pairs to multi-skill workforce assignment, we express the stability condition with linear inequalities of binary variables, and finally we propose MIP-models to find optimal workforce assignments.

Players. In this workforce assignment problem, players are *technicians* and *jobs*. The skill requirements of a job is a combined requirement covering all skill requirements of tasks in the corresponding sequence. We assume that preferences are *complete* and *without ties*.

Blocking case. In our assignment game, feasibility must be taken into account. Given an assignment, we say that job j *likes* a technician t if and only if t is not in the team of j and there exists a technician t' in the team of j such that j prefers t to t' and t can replace t' by preserving the team feasibility if j. Technician t *likes* job j if and only if t is not in the team of j and t prefers j to his current job. Note that technicians like jobs based only on preferences, whereas this is not the case for jobs.

Definition 1.6.1. *(Stability of a workforce assignment)*
A workforce assignment is "stable" if and only if it does not contain such a technician-job pair that they like each other.

Next, we give a formal definition of stable workforce assignment problem.

> Problem: Stable technician-job assignment problem
>
> Instance: Given the set $\mathbb{S}$ of specialization fields, the set $\mathbb{L}$ of hierarchical skill levels , the set T of technicians, and the set J of jobs. For a technician $t \in T$, the matrix $S_t \in \{0,1\}^{\mathbb{L}\times\mathbb{S}}$ specifies skills and P_t specifies the preferences. For a job $j \in J$, the matrix $RQ_j \in \mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$ specifies the skill requirements and P_j specifies the preferences.
>
> Question: Does there exist a technician-job assignment such that skill requirements of every job is satisfied and it is *stable* as in the Definition 1.6.1?

The above problem is denoted by $STJAP(n,n,n)$ where the three n's stand for $|\mathbb{L}|$, $|\mathbb{S}|$, and the upper bound on the size of teams in feasible assignments respectively. The following theorems summarize our results.

Theorem 1.6.1. *(Section 3.4.2) STJAP(n,1,n) is in P.*

In the proof of the above theorem, we reduce STJAP(n,1,n) to a version of the university admission problem.

Theorem 1.6.2. *(Section 3.4.3) STJAP(n,n,n) is NP-complete.*

We note that the proof of the above theorem is by reduction from "three dimensional matching problem" (3-DM).

Theorem 1.6.3. *(Section 3.5) The set of stable technician-job assignments can be characterized by a set of linear inequalities of binary variables.*

Computational results. Our computational results show that stable assignments of the instances fewer than 20 technicians, around 10 jobs with at most 5 skill domains and 5 skill levels can be found in less than one second by solving our MIP model (see Section 3.5.2).

1.6.3 The dial-a-ride problem

In Chapter 4, we focus on the feasibility testing of a dial-a-ride problem, that is introduced by Hunsaker and Savelsbergh (2002). The work in this chapter is published by Firat and Woeginger (2011b) in a refereed journal.

Our main result in this chapter shows that this feasibility testing can be done in linear time by expressing it as a shortest path problem in vertex-weighted interval graphs. This result is obtained in several steps given below.

System of linear inequalities. We formulate the dial-a-ride feasibility test of Hunsaker and Savelsbergh (2002) as a system of linear inequalities (Section 4.3).

Difference constraint system. The obtained system of linear inequalities is rewritten by standard methods into a system of difference constraints. At this step, we create a directed graph in which every variable is represented by a vertex and every difference constraint is represented by an arc. The underlying difference constraint

system has a feasible solution if and only if the corresponding directed graph does not contain any negative-weight cycles (see Theorem 24.9 in Cormen et al. (2009),).

Shortest path problem. A careful examination of forward and backward arcs in the directed graph results in realizing that the graph has a negative weight cycle if and only if the there exists a shortest path between two specific vertices. This brings us to the shortest path problem in directed graphs which can be solved in $O(n \log n)$ time by the algorithm of Fredman and Tarjan (1987). We note that every forward arc on this directed graph can be perceived as a time interval (Section 4.5.1).

Vertex-weighted interval graph. Our last step in getting a linear time algorithm is realizing that two arcs on a shortest path have intersecting time intervals. Then we define a vertex-weighted interval graph in which every arc of the directed graph in our previous stage is represented by an interval with the same weight as of the arc. Finally, we conclude that the feasibility test for the dial-a-ride problem of Hunsaker and Savelsbergh (2002) can be done in linear time (Section 4.5.1).

1.6.4 The vehicle refueling problem

In Chapter 5, we worked on a vehicle refueling problem. We present a greedy algorithm for the problem in Section 5.4. We give three proofs for the correctness of the greedy algorithm. In the literature, equivalent problems are studied under lot sizing and network flows. For those problems, researchers proposed greedy algorithms that are the same as the one we give in Section 5.4.

Analysis of solutions. In the solutions of the greedy algorithm, we distinguish two cases; (1) an *empty-tank* event occurs in a city, if the vehicle departs from that city with a fuel amount in tank just enough to reach the next city, and (2) a *full-tank* event occurs in a city, if the vehicle departs with a fuel tank. The *fuel type* refers to the city index of the fuel amount. The most expensive fuel type in the tank, when an event occurs, is said to be *dominant* and a dominant fuel type is associated with a number of consecutive cities that form its *region*. The price of the dominant fuel type is called *regional price* of its region.

Regional prices. Empty-tank event results in a regional price decrease. Full-tank event results in a regional price increase.

Dual variables. There are three types of dual variables. The first type dual variable corresponds to the regional price increase and it takes positive value in a city if and only if the regional price increases. The second type dual variable corresponds to the regional price decrease and it takes positive value in a city if and only if the regional price decreases. The third type dual variable corresponds to the difference between the regional price in a city and the fuel price in that city. It takes positive value if this difference is positive.

In Section 5.4.3, we prove the correctness of the greedy algorithm using network flow. On a specific network, we express solutions of the greedy algorithm by a flow. Then we show that there is no negative cost directed cycle in the residual network of this flow, hence the flow is optimal or has minimum cost (see Chapter 9 of Ahuja et al. (1993)).

In Section 5.4.4, we prove the correctness of the greedy algorithm using convexity.

We explore the neighborhood of a solution of the greedy algorithm and we show that all neighbor solutions have higher cost than the solution of our greedy algorithm. This implies that solutions of our greedy algorithm have locally minimum cost. By considering that the problem has a linear, hence convex, objective function, they have globally minimum cost.

Chapter 2

A MIP-based approach to a multi-skill workforce scheduling problem

This chapter deals with scheduling complex tasks with an inhomogeneous set of resources. The problem is to assign technicians to tasks with multi-level skill requirements. Here, the requirements are merely the presence of a set of technicians that possess the necessary capabilities. An additional complication is that a set of combined technicians stays together for the duration of a work day. This typically applies to scheduling of maintenance and installation operations. We build schedules by repeated application of a flexible matching model that selects tasks to be processed and forms groups of technicians assigned to combinations of tasks. The underlying mixed integer programming (MIP) model is capable of revising technician-task allocations and performs very well, especially in the case of rare skills.

2.1 Introduction

As specialization in production and maintenance increases, the importance of skill management in employee scheduling grows significantly. Especially when activities require skills from several specialization fields at different levels, skill management becomes more challenging. Multi-skilled employee scheduling can be encountered, for example in companies having operations like maintenance, construction and installation in which the work is carried out at different physical locations. Then it makes sense to keep a combined group of workers together for a workday.

The scheduling problem under consideration is the ROADEF Challenge 2007 problem. It falls in the class of "resource-constrained project scheduling problem" (the RCPSP) and has some additional aspects that increase the complexity and make it impossible to use known approaches for the RCPSP in the literature. We develop an approach to this problem based on a hybrid combination of MIP models and apply it on maintenance instances provided by France Telecom in the ROADEF Challenge 2007. Since the problem instances of France Telecom have been used as a test bed

for a computational challenge held in 2007, we can compare the performance of our method to other approaches tackling the same problem.

In the problem, we are given a set of tasks and a set of technicians. The capabilities or skills required by tasks are described in *domains* (specialization fields), at several levels of expertise so-called *hierarchical levels*. A task can be processed by a group of technicians in a fixed time provided that the collective capabilities of this group are above a certain threshold that is called its *skill requirements*. Among tasks, there are *precedence relations* requiring that a task must be completed before another task can be processed. Then the former is said to a *predecessor* of the latter. The set of tasks is partitioned into several priority classes depending on their urgency levels, customer properties, and so on. The latest completion time of tasks under a priority class is called the *priority span*.

The time line is split into equal-size periods of hours that are called the *workdays*. Start time and completion time of a task must be within the same workday. In each workday, teams of technicians are supposed to perform a number of tasks without overlapping, one at a time, without interruption. In deciding which tasks are to be processed by which teams, an important constraint is that tasks should be *sequenced* properly not to violate any precedence relation. Any travel or setup time between tasks in a sequence is not taken into account. A team formed on a certain workday to carry out a task must stay together for the duration of that workday.

Outsourcing some tasks is possible by using a fixed budget. Every task has a fixed outsourcing cost and since the transportation times and the lead times are not known, we use the convention of "outsourcing the successors of outsourced tasks" which means if a task is outsourced, so are all its successors. Clearly, completion times of the outsourced tasks are not taken into account.

The availability of technicians is considered in the problem. Completion times of hard tasks are highly effected if some experts are not available on some workdays. Therefore, it is important to outsource the right combination of tasks when there is a heterogenous skill distribution within the technician group. We call this *rare expertise*. Moreover, efficient packing of the expert-requiring tasks on the days when experts are available is also crucial.

The objective of the problem is *minimizing* the weighted sum of priority spans. Therefore, it is clear that the outsourcing budget is to be used as much as possible to achieve short completion times.

Results of this chapter. We propose a MIP-based combinatorial approach to the workforce scheduling problem of France Telecom. In our approach, we formulate several key task properties and calculate lower bounds for the problem. Our approach constructs good quality solutions compared to the other known approaches.

The proposed combinatorial approach is composed of two phases: *preprocessing* and *schedule construction*. In the preprocessing phase, we calculate several key properties of tasks and calculate lower bounds. In order to calculate the lower bounds, we solve a simplified problem in which skill requirements and precedence relations of tasks are relaxed, preemption is allowed, unavailability of technicians is taken into account, and the option of outsourcing tasks is preserved. The problem is formulated as a MIP model which assumes that priority classes are completed in a pre-determined

order. This order is given by the specified permutation of priority classes (for brevity: "priority permutation").

In the schedule construction phase, a number of alternative schedules are built by varying several parameters and strategies. One important parameter for a schedule is the priority permutation, since priority classes are handled sequentially while constructing a schedule. It is assumed that priority permutations with smaller lower bound values promise low-cost schedules. Therefore, firstly, the priority permutation with smallest lower bound value is used.

From our point of view, a schedule is composed of workday schedules due to the following aspects of the problem: a task cannot be performed in more than one workday, and teams of technicians are formed daily. Therefore, our algorithm finds a packing of tasks (workloads of teams) greedily in the form of sequences, each being not longer than a workday. The task sequences are initialized by single tasks and their lengths are increased by adding more tasks iteratively. In every iteration, the algorithm simultaneously finds enough skilled technicians for every sequence. Constructing a day schedule in this way was firstly introduced by Hurkens (2009) in the ROADEF Challenge 2007 and this approach was ranked first in the final stage among 11 qualified participants.

In this study, our main contribution is introducing the flexibility in extending the task sequences. The flexibility is maintained by three aspects of our approach (1) the sequences are allowed to get *merged* if necessary (2) the technicians can be *exchanged* easily among the groups (3) ordering of tasks in a sequence is determined *dynamically* by considering their precedence relations. The goal of this flexibility is being able to schedule more tasks within a workday, especially the ones requiring experts. If the expertise in the technician group is rare, as it is usually the case in the companies, then the schedule cost is sensitive to the utilization of the experts. Hence, it is expected, and also supported by our computational results, that the flexibility of our approach leads to better packing of hard tasks and lower schedule costs.

The extension of task sequences is carried out by finding simultaneous technician-tasks assignments that correspond to many-to-one type matchings on the constructed bipartite graph. Matchings are restricted by skill requirements, precedence relations and total durations. The problem of finding matchings is formulated as a MIP model with small number of variables.

The chapter is organized as follows. In Section 2, we give the problem description. Problem complexity is analyzed in Section 3 and a literature review is presented in Section 4. Our solution approach is explained in Section 5. The computational results of our algorithm are comparatively reported in Section 6 and finally we discuss the applicability of our solution methodology to other multi-skill workforce scheduling problems in Section 7.

2.2 Problem description and notation

The problem we consider in this chapter was described by Dutot et al. (2006) as the contest problem in the ROADEF Challenge 2007. In the following sections we

describe the problem and introduce the necessary notation.

2.2.1 Skills

The tasks in our scheduling problem require specializations in several fields. We use the term *skill domain* for a specialization field and *skill level* to interpret the degree of expertise *hierarchically*. The set of skill domains (levels) is denoted by $\mathbb{S}$ ($\mathbb{L}$). A skill at level $l \in \mathbb{L}$ in domain $s \in \mathbb{S}$ is denoted by $\langle l, s \rangle \in \mathbb{L} \times \mathbb{S}$. Skill requirements of tasks are specified by matrices in $\mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$. Technician skills can be expressed as skill vectors in $\{0, 1, \dots |\mathbb{L}|\}^{\mathbb{S}}$, however using the skill matrices in $\{0, 1\}^{\mathbb{L}\times\mathbb{S}}$ makes it easier to formulate the problem.

Expertise in the literature of multi-skill workforce project scheduling

To the best of our knowledge, only Bellenguez and Neron (2004)consider hierarchical skill levels other than the participants of the ROADEF Challenge 2007. The majority of studies treat human resources as skilled or unskilled in domains (for example Cai and Li (2000), Bellenguez and Neron (2007), Li and Womer (2009), Valls et al. (2009) and Avramidis et al. (2010)). Gutjahr et al. (2008), Yoshimura et al. (2006), and Heimerl and Kolisch (2010) use competence score as an interpretation of skill level. The drawback of scoring employee expertise rather than leveling has the difficulty in expressing skill requirements of tasks clearly. If a task requires expertise, then its skill requirement is merely a high value in a corresponding domain. This high demand may be satisfied either by assigning an expert or by collecting many non-expert employees. In case of skill leveling, the second case is not an option. Gutjahr et al. (2008) limit the number of employees that can be assigned to a task and define special variables for experts to handle this issue in their nonlinear MIP model. Yoshimura et al. (2006) use a specific parameter for experts or so-called "project leaders" in order to select an expert for each project.

Use of skills: We assume that technicians contribute simultaneously in all possible domains while processing a task and this is called *simultaneous skill use*. In the literature of multi-skill workforce project scheduling, Valls et al. (2009), Heimerl and Kolisch (2010), Drezet and Billaut (2008) and Ballou and Tayi (1996) also make the assumption of *simultaneous skill use*.

2.2.2 Technicians

We are given a set T of technicians to perform the tasks. The unavailability periods of technicians are considered within a finite scheduling horizon. $A(t, h)$ denotes the unavailability of technician t. It is equal to 1 if t is available on day h and zero otherwise.

Skills of a technician, say $t \in T$, are expressed by a matrix $S_t \in \{0, 1\}^{\mathbb{L}\times\mathbb{S}}$. Although a more compact way of expressing the skills of technicians is using skill vectors in $\{1, \dots, |\mathbb{L}|\}^{\mathbb{S}}$, skill matrices are convenient to formulate the problem and to develop the necessary notation. If technician t is proficient in skill $\langle l, s \rangle$, then

$S_t^{\langle l,s\rangle} = 1$. Clearly, if a technician is qualified in a skill, then he is also qualified at lower levels in the domain of this skill. Hence $S_t^{\langle l,s\rangle} = 1 \Rightarrow S_t^{\langle l',s\rangle} = 1, \forall l' \leq l$. Once we are given S_t, the proficiency of technician t in skill domain s is found by $\max\{\{0\}, \{l \in \mathbb{L} | S_t^{\langle l,s\rangle} = 1\}\}$.

A skill matrix example of a technician $t \in T$, in a problem instance with $|\mathbb{L}| = 3$, $|\mathbb{S}| = 4$ may be

$$S_t = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

We see that technician t is expert with proficiency of level 3 in domain 1. He qualifies to skill level 2 in domain 3, but he has no skill in domains 2 and 4. In the above example, the skill vector of technician t is $SV_t = (3, 0, 2, 0)$.

Skills of teams: Let $\tau \subset T$ denote a team of technicians. We note that the skill matrix of a team may not have binary entries anymore. We find skills of the team τ by summing up the skills of technicians in τ, so we have $S_\tau = \sum_{t \in \tau} S_t$.

2.2.3 Tasks

In our scheduling problem, a set J of tasks is given. In this section we explain the aspects of our scheduling problem related to tasks.

Skill requirements: Tasks require skill qualifications. The skill requirements of a task $j \in J$ are expressed by a matrix $RQ_j \in \mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$ which provides the information of the desired skill quantity (number of technicians) and skill quality (expertise). The requirements in RQ_j are cumulative in the sense that any requirement at a level is carried to lower ones in the same domain. Therefore, for a task j and a skill $\langle l, s\rangle$ we have $RQ_j^{\langle l',s\rangle} \geq RQ_j^{\langle l,s\rangle}$ for all $l' \leq l$.

An example of skill requirement matrix for task j in an instance for which $|\mathbb{L}| = 3$ and $|\mathbb{S}| = 4$ may be given by

$$RQ_j = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

According to the given skill requirement example, in a team processing task j, there must be at least two technicians qualified in domain 2, one being proficient at least at level 2 and one at least at level 1. Let $T(j) \subset T$ be such a team. Consequently, the team $T(j)$ must satisfy $S_{T(j)}^{\langle l,s\rangle} \geq RQ_j^{\langle l,s\rangle}$ for all $\langle l, s\rangle \in \mathbb{L} \times \mathbb{S}$.

Durations: The time needed to perform task j is called its duration and denoted by d_j. The duration of each task is fixed and does not vary with the number and expertise of technicians assigned. Processing of tasks cannot be interrupted, and if a team started performing a task, that team must finish the task within the same workday. This also implies that $d_j \in \{1, 2, \ldots, H\}$ where H is the workday length.

Precedence relations: Precedence relations of task j enforce that all tasks in $Pred(j)$ must be completed before the task j starts. A task $k \in Pred(j)$ is said

to be a *predecessor* of task j and their relation is denoted by $k \to j$. Moreover, task j is also said to be a *successor* of task k. Let CT_j be the completion time of task j in a schedule and it is the sum of the hours in previous workdays and the portion of the workday in which task j is completed. Precedence relations of task j enforce $CT_k \leq CT_j - d_j$, for all $k \in Pred(j)$.

Outsourcing: External companies may be hired to outsource a task by paying its outsourcing cost c_j. Outsourced tasks are discarded and need not be scheduled. The total cost of outsourced tasks must not exceed the outsourcing budget B. In the problem definition, no related information to outsourcing of tasks is given like delivery times and transportation times, so we have the convention of outsourcing the successors of outsourced tasks. Let $\Omega \subseteq J$ denote the set of outsourced tasks. The convention of outsourcing the successors results in the following property of Ω: $\{k \in J \mid \Omega \cap Pred(k) \neq \emptyset\} \subseteq \Omega$.

Priority classes: Tasks are partitioned into several priority classes. We denote the set of priority classes by P and $P(j) \in P$ is the priority class of task j. The latest completion time of tasks under a priority class is called the priority span. It is denoted by C_p for a priority class $p \in P$ and found by $C_p = \max\{CT_j \mid P(j) = p, j \notin \Omega\}$. The overall make span is the length of a schedule and denoted by $C_0 = \max\{CT_j \mid j \in J \setminus \Omega\}$. Note that outsourced tasks do not contribute to the schedule cost. Priority class 0 is an artificial priority class that is used to include the overall makespan in the quality evaluation of schedules and of course every task belongs to priority class 0.

2.2.4 Schedules

In this section we explain some more notation about schedules and we state the feasibility conditions. Finally, the objective of the scheduling problem is discussed.

Workday Concept: The time axis is partitioned into intervals of length H. These successive intervals represent workdays. Within a certain time interval (workday), the technicians performing a certain task must work together during this interval and they form a team. Another important restriction is that the processing of each task must stay within a time interval. The set D is assumed to have enough number of workdays to complete the processing of all tasks.

Teams: In each workday of schedules, teams of technicians are formed to process the assigned tasks. Let Ξ be the set of teams specified by a schedule and $\tau \in \Xi$ be a team in that schedule. $T(\tau) \subseteq T$ denotes its technicians, $J(\tau) \subseteq J$ denotes the tasks in the workload of τ, and $\delta(\tau) \in \{1, 2, \ldots, |D|\}$ denotes the workday in which τ is formed. If tasks j' and j are in $J(\tau)$, then $j' <_\tau j$ denotes that j' is processed before j. In the schedules constructed by our combinatorial algorithm, we assume that there is no *idle time* between the tasks in the workloads of teams. Therefore the completion time of task j is determined as below:

$$CT_j = (\delta(\tau) - 1)H + \sum_{j' : j' <_\tau j} d_{j'} + d_j \tag{2.1}$$

Note that the team information consisting of sequences of tasks and groups of technicians is enough to define a solution. Workday schedules constructed by our

combinatorial algorithm together with the corresponding outsourcing decision form a well-defined solution. Therefore we do not have decision variables for start time of tasks in our matching models.

Feasibility of schedules

A schedule is feasible if and only if the following constraints are satisfied:

- *Outsourcing:*

$$\sum_{j \in \Omega} c_j \leq B \tag{2.2}$$

The total cost of outsourced tasks must not exceed the outsourcing budget.

$$\{k \in J \mid \Omega \cap Pred(k) \neq \emptyset\} \subseteq \Omega \tag{2.3}$$

The successors of the outsourced tasks are outsourced as well.

- *Task and technician assignments:*

$$\mid \{\tau \in \Xi | j \in J(\tau)\} \mid = 1, \quad \forall j \in J \setminus \Omega \tag{2.4}$$

Each non-outsourced task is processed by exactly one team.

$$\mid \{\tau \in \Xi | t \in T(\tau), \delta(\tau) = h\} \mid \leq A(t,h), \quad \forall h, \forall t \in T \tag{2.5}$$

Technicians can be in at most one team on the days they are available.

- *Completion times*

$$CT_k \leq CT_j - d_j, \quad \forall j \in J \setminus \Omega, \forall k \in Pred(j), \tag{2.6}$$

A successor must not start before the completion of all of its predecessors.

$$\max\{(\delta(\tau)-1)H, \max\{CT_{j'} | j' <_\tau j\}\} \leq CT_j - d_j, \quad \forall \tau \in \Xi, j \in J(\tau) \tag{2.7}$$

Processing of tasks in the team workload must not overlap. Note that we do not allow "*idle times*" between tasks in a team workload. Therefore, the inequality sign can be replaced by an equality sign for the solutions of our combinatorial algorithm.

$$CT_j \leq \delta(\tau)H, \quad \forall \tau \in \Xi, \forall j \in J(\tau) \tag{2.8}$$

Workloads of teams must not exceed the workday length.

$$C_p = \max\{CT_j \mid P(j) = p, j \notin \Omega\} \tag{2.9}$$

Priority span is the latest completion time of the tasks belonging to that priority.

- *Skill requirements:*

$$RQ_j^{\langle l,s\rangle} \leq S_{T(\tau)}^{\langle l,s\rangle}, \quad \forall \tau \in \Xi, \forall j \in J(\tau), \forall \langle l,s\rangle \in \mathbb{L} \times \mathbb{S} \tag{2.10}$$

The technicians in every team must be enough skilled to process the assigned tasks in the workload.

Objective

The objective value of a solution in our scheduling problem is calculated by $\sum_p w(p)C_p$ where C_p denotes the priority span as expressed in (2.9). In the benchmark instances of France Telecom, the weights were given as $\{1, 28, 14, 4, 0\}$ for priorities $\{0, 1, 2, 3, 4\}$ respectively. The objective is to *minimizing* this weighted sum of priority spans. Note that the outsourcing cost is not included in the objective, therefore, in the combinatorial algorithm, we aim to use the outsourcing budget as much as possible to decrease the total work load in the schedule.

A MIP model of the problem is given by Cordeau et al. (2010). The authors report that after 24-hour run of the MIP solver, an optimal schedule of even small instances with 7 technicians and 20 tasks could not be found.

2.3 Problem complexity

Let us call the decision version of our scheduling problem the "Scheduling Problem of France Telecom" (SPFT). Firstly, we give a formal definition of the SPFT and then we prove that it is NP-complete.

> Problem: Scheduling Problem of France Telecom (SPFT)
>
> Instance: Given a numerical bound k, integers H, B, denoting workday length, outsourcing budget of the project respectively. The set T of technicians, the set J of tasks, the set P of priority classes, the set $\mathbb{L} \times \mathbb{S}$ of skills, and the set D of workdays. For every $t \in T$; there are skills $S_t \in \{0,1\}^{\mathbb{L}\times\mathbb{S}}$ and availability $A(t,h) \in \{0,1\}$, for all $h \in D$. For every $j \in J$; there are skill requirements $RQ_j \in \mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$, duration $d_j \in \{1, \ldots .H\}$, outsourcing cost $c_j \geq 0$, predecessors $Pred(j) \subseteq J$, and priority class $P(j) \in P$. The last completion of tasks in priority class $p \in P$ is denoted by C_p and $w(p)$ denotes the weight of the priority class p.
>
> Question: Does there exist a schedule satisfying the feasibility conditions mentioned in Section 2.2.4 with cost $\sum_{p\in P} w(p)C_p$ no more than k?

Theorem 2.3.1. *SPFT is NP-complete.*

Proof. First of all, SPFT is in NP, since in polynomial time we can compute the cost of a given schedule and we can check whether it is feasible. We give a reduction from the subset sum problem to the SPFT; see Garey and Johnson (1979) for the NP-completeness of the subset sum problem.

> Problem: Subset Sum
>
> Instance: An integer Σ and a set $A = \{a_1, \ldots, a_n\}$ in which each element a_i has size $s(a_i)$ and $\Pi = \sum_{a_i \in A} s(a_i)$.
>
> Question: Does there exist a subset $A' \subseteq A$, such that the total size of elements in A' is equal to Σ?

Let us construct a special case of the SPFT from an instance of the subset sum problem as follows:

- $H = \Pi - \Sigma, B = \Sigma, |D| = 1$.
- $L = \{1\}, S = \{1\}$, and $P = \{1\}$ with $w(1) = 1$.
- $T = \{1\}$ with $S_1 = \{1\}, A(1,1) = 1$.
- for every item $a_i \in A$ we create a task $j(a_i)$ with $d_{j(a_i)} = c_{j(a_i)} = s(a_i)$, $R_{j(a_i)} = \{1\}$, and $Pred(j(a_i)) = \emptyset$.

Question: Does there exist a schedule with objective value H ?

Note that the minimum value of the schedule cost is attained when outsourcing budget is completely used: $\sum_{j(a_i) \in \Omega} c_{j(a_i)} = B = \Sigma$ making the schedules length as well as the schedule cost $\Pi - \Sigma = H$. So a YES instance of the special case of the SPFT corresponds to a YES instance of the subset sum problem. It is also clear that a NO instance of the special case of the SPFT corresponds to a NO instance of the subset sum problem. This shows that the subset sum problem is reduced to the SPFT, hence the SPFT is NP-complete. □

2.4 Literature review

2.4.1 The resource-constrained project scheduling (RCPSP)

The problem considered in this chapter is a generalization of the RCPSP that is extensively reviewed by Brucker et al. (1999) and Hartmann and Briskorn (2010).

"Multi-mode resource-constrained project scheduling problem" (MM-RCPSP) is a generalization of the RCPSP in which activities may require renewable, non-renewable, and doubly constrained resources. (see for example De Reyck and Herroelen (1999)). In the "multi-skill project scheduling problem" (MSPSP) the resources are renewable human resources or staff members. Every staff member can have several skills among the needed ones by the activities. As Bellenguez and Neron (2007), and Li and Womer (2009) mentioned, the MM-RCPSP formulation can be used to describe MSPSP, however the number of combinations becomes very large even for

the moderate size of employee groups, thus making it impossible to use the exact methods proposed for the MM-RCPSP to solve the MSPSP.

In the literature of project scheduling with multi-skilled human resources, several objectives are considered. For example, Avramidis et al. (2010), and Li and Womer (2009) consider minimizing staffing cost; Bellenguez and Neron (2004), Bellenguez (2006) consider minimizing the makespan, and Wu and Sun (2006) consider minimizing the outsourcing cost. Gutjahr et al. (2008) use a hybrid objective of maximizing economic gains and personal improvement, Heimerl and Kolisch (2010) minimize both makespan and outsourcing cost.

We encounter different solution methodologies in the literature of project scheduling with multi-skilled human resources. Heimerl and Kolisch (2010) formulates an elegant MIP model to solve project staffing and project scheduling simultaneously. Li and Womer (2009) proposes a hybrid algorithm based on MIP modeling and constraint programming. The authors argue that Bellenguez and Neron (2004) do not consider personnel capacity that seems to be a result of unavailability, expertise, and other personal attributes. In another recent study, Gutjahr et al. (2008) proposes a greedy heuristic as well as a hybrid solution methodology using priority-based rules, ant colony optimization and genetic algorithm to solve the so-called "project selection, scheduling and staffing with learning problem".

The project scheduling problem considered by Bellenguez and Neron (2004) shows significant similarity to our problem. In their scheduling problem, skills are expressed in domains by hierarchical levels in the same way as our problem. Skills of workers and skill requirements of jobs are specified by the same matrices as in Section 2.2.2 and in Section 2.2.3. The authors assume that technicians can only work in one skill domain while performing a task contrary to our assumption "*simultaneous skill use*". This assumption seems more reasonable in cases if tasks take short time and require skills in many domains. On the other hand, if the tasks in a project require skills in several but not many domains, assigning one person to each piece of work may lead to underutilization. In the authors' problem, activities can be processed continuously after each other, so there is no workday concept. Outsourcing of tasks is not an option in the problem and the objective is minimizing the makespan whereas we minimize a weighted sum of priority spans. Consequently both problems fall in the class of the RCPSP, but differ from each other in the mentioned points.

Bellenguez and Neron (2004) worked on finding lower bounds for the makespan. Firstly, the authors use the precedence graph to find a lower bound value that is the length of the longest path (critical path). For every activity, a time window with an earliest start time and a deadline is determined. Then the authors improve the lower bounds value using two methods: compatibility graph and energetic reasoning. In the compatibility graph, a node represents an activity with a weight that is equal to its duration and two activities are joined by an edge if their time windows intersect and if both can be processed by workers at the same time. In this graph a maximum-weight independent set is found by using a heuristic algorithm and by solving a MIP model with CPLEX. The weight of the independent set provides an improved lower bound value. In energetic reasoning, certain time windows are calculated and in each of them it is checked if the mandatory parts of the activities can be processed by workers. In

a negative result, the current lower bound is increased by one time unit.

2.4.2 Solution approaches in the ROADEF Challenge 2007

In the ROADEF Challenge 2007, the solution approach of Hurkens (2009) was ranked first. Cordeau et al. (2010) and Estellon et al. (2009) tied for second place. Cordeau et al. (2010) describes a MIP model for our scheduling problem and the authors mention that it can not be solved for large instances optimally in a reasonable time. They develop a meta-heuristic method that consists of a construction heuristic and an adaptive large neighborhood search with several destroy and repair methods. The solution strategy is viewed as a standard simulated annealing algorithm with a complex neighborhood search due to the acceptance criterion of the solutions.

Estellon et al. (2009) designed a local search scheme in which a greedy algorithm is employed to obtain a feasible solution and this solution is improved by a local search strategy. The authors use a methodology including three key points, *search strategy*, *moves* and *evaluation of moves*. They also point out that a careful implementation increases the convergence speed of local-search heuristics and stochastic elements are useful to improve the diversification.

As mentioned in the introduction, Hurkens (2009) considers the same problem and proposes a two-phase MIP-based solution methodology. In the first phase, a MIP model computes lower bound values and determines the tasks to be outsourced. Our MIP model is based on the author's one, but it is a slightly improved version in a way that tighter lower bound values for instances with heterogeneous skill distribution are found. In the second phase, two matching models are used to find technician-task assignments having limited flexibility compared to our matching models.

2.5 Scheduling with flexible matching model

2.5.1 An overview of the combinatorial algorithm

The scheduling problem under consideration is a complex problem. The main goal of the problem turns out to determine the workloads of teams, such that not only the number of processed tasks on each workday should be maximized, but also the tasks in a workload of a team should have similar skill requirements. Especially, when the *skill quality* or the expertise, within the technician group is limited, finding a match between workloads and the team skills becomes harder due to the heterogeneity in skill distribution. In such cases, the main goal turns out to be maximizing the number *as well as* the *hardness* of selected tasks in the workloads, since the priority spans are sensitive to the utilization of experts.

Hurkens (2009) introduced the idea of greedily constructing the workloads, while satisfying skill requirements. The author developed a solution methodology using *matching models* which find technician-task assignments simultaneously. In this thesis, we improved Hurkens' solution approach by obtaining tighter lower bound values and by adding flexibility to the matching models to have a better packing of hard tasks in schedules.

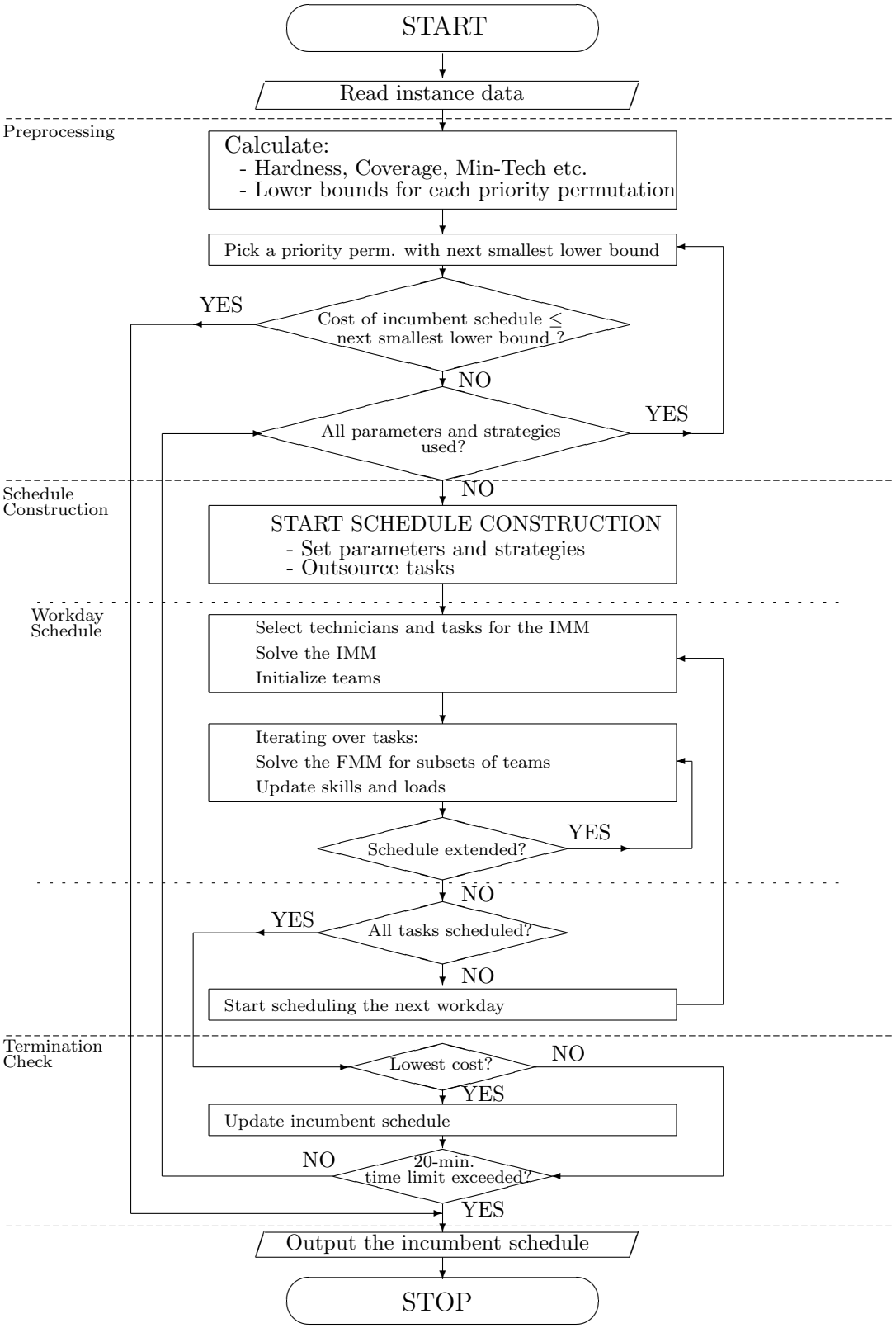

Figure 2.1: Flowchart of the Combinatorial Algorithm

Figure 2.1 shows the flowchart of our algorithm. The algorithm consists of two main phases; preprocessing and schedule construction. The preprocessing phase includes calculation of necessary parameters for tasks and computation of lower bounds. Computing lower bounds is merely solving a simplified problem in which skill requirements and precedence relations are relaxed, pre-emption is allowed, and only technician availabilities are taken into account. The simplified problem boils down into finding minimum time needed to satisfy the cumulative man-hour demand of tasks with the option of outsourcing tasks. It is assumed that the priority classes are processed sequentially and lower bounds are computed for all priority permutations in practice, since depending on the number of tasks, any priority sequence may result in lower schedule cost than the others.

In the schedule construction phase, alternative schedules are constructed by sequentially assigning priority classes. Hence, for every schedule, a certain priority permutation is fixed in advance. Priority permutations are considered according to their lower bound values: the one with smallest lower bound value is considered first, and next the second smallest and so on. Moreover, we construct more than one schedule for the same priority permutation by using several parameters and strategies for some decisions.

Having constructed a complete schedule, its cost is compared to the cost of incumbent schedule. If the cost is smaller, then the incumbent schedule is updated with the latest constructed one. Whenever the cost of the incumbent schedule is smaller than or equal to the lower bound value of the next considered priority permutation, then this priority permutation is neglected. In such a case, it is clear that no schedule under that priority permutation can improve the schedule cost ever found. Time limit was specified as 20 minutes in the ROADEF Challenge 2007, so the algorithm stops constructing schedules unless it is terminated before.

The combinatorial algorithm builds workday schedules successively. Constructing a workday schedule starts with the "*initial matching*". Initial matching is performed by solving a MIP model called the 'initial matching model" (IMM). The model finds a many-to-one type matching on a bipartite graph in which one partition includes technicians and one partition includes tasks. Initial matching results in a partially constructed day schedule in which teams have a single task in their workloads. Next, the number of scheduled tasks in the initialized workday schedule is increased by adding more tasks greedily. We call the process of increasing the number of scheduled tasks "extending workday schedule" and this process is performed by solving a MIP model called the "flexible matching model" (FMM). Tasks can be inserted into a partial workday schedule as long as there are some teams with workload length less than a workday, skills requirements are met, and sequencing of tasks is possible respecting the precedence relations. The algorithm starts scheduling the next workday, if no more tasks can be inserted into the current workday schedule. Constructing of workday schedules continues until all non-outsourced tasks are scheduled.

Cordeau et al. (2010) and Estellon et al. (2009) argue that the combinatorial algorithm proposed by Hurkens (2009) is an application of local search with large neighborhood exploration. However both our algorithm and Hurkens' algorithm, are constructive heuristics. Alternative schedules are constructed and once a complete

schedule is constructed, then it is not modified. The strategy of both algorithms is to obtain good quality solutions with the simultaneous technician-task assignments. Therefore they cannot be classified as local search algorithms.

2.5.2 Calculating key properties of tasks

In the preprocessing phase, several key properties of tasks are calculated by aggregating their multi-dimensional properties. They are *min-tech*, *hardness*, *coverage*, and *matching weight*. In this section, all properties of the tasks are calculated by considering the skills that are possessed by the technician group.

Min-tech

Min-tech, denoted by MT_j, of a task j is the minimum number of technicians who can process it. This simple concept is important for two reasons: (1) man-hour demand of task j is calculated by $MH_j = MT_j d_j$ and this is used in lower bound calculations (2) in the FMM where the efficiency of an assignment is controlled by punishing the (positive) deviation from MT_j.

A simple integer programming (IP) model in (2.11) is used to calculate Min-Tech for each task at a time and this IP model is solved by CPLEX. The binary decision variable x_t indicates that technician t is assigned to task j. The constraints of the model enforce that skill requirements of task j are met and the objective minimizes the size of the selected team. Note that the index j is not used, since we run the following IP model for each task at a time.

$$MT_j = \min \left\{ \sum_{t \in T} x_t : \sum_{t \in T} S_t^{\langle l,s \rangle} x_t \geq RQ_j^{\langle l,s \rangle}, \forall \langle l,s \rangle \in \mathbb{L} \times \mathbb{S};\ x_t \in \{0,1\} \right\} \tag{2.11}$$

Hardness

A task is said to be *hard* if it requires skills that are not common among technicians. Hence hardness of a task is a relative concept depending on the skill distribution of a technician group. For example, a task requiring moderate skill levels may be "relatively" hard for a technician group, if there are few technicians specialized in the demanded fields. Before defining the hardness, we define the "value" of a skill, denoted by $v(l,s)$ for skill $\langle l,s \rangle$. The value of a skill is the ratio of the number of tasks that demand it to the number of technicians who possess it. It is calculated by

$$v(l,s) = \frac{|\{j \in J | RQ_j^{\langle l,s \rangle} > 0\}|}{S_T^{\langle l,s \rangle}}, \quad S_T^{\langle l,s \rangle} = \sum_{t \in T} S_t^{\langle l,s \rangle}. \tag{2.12}$$

Note that the value of a skill is high if the skill is rare among the technicians and also if there is some demand from tasks. Having introduced skill value, we can define "hardness", denoted by h_j for task j, by

$$h_j = \sum_{\langle l,s\rangle \in \mathbb{L}\times\mathbb{S}} \upsilon(l,s) RQ_j^{\langle l,s\rangle} \tag{2.13}$$

If the skill requirement of task j is tightly satisfied by the whole technician group, then the processing of that task is impossible when some of the necessary technicians are unavailable. It may also be very difficult to build a team for such task on a certain day, if necessary technicians have already been allocated to several teams.

The tightness of skill satisfaction of a task can be expressed by

$$\chi_j = \max_{\langle l,s\rangle \in \mathbb{L}\times\mathbb{S}} \left\{ \frac{RQ_j^{\langle l,s\rangle}}{S_T^{\langle l,s\rangle}} \right\} \tag{2.14}$$

Definition 2.5.1. *(Special task)*
A task j with $\chi_j = 1$ is called "special".

Note that $0 \leq \chi_j \leq 1$ for feasible problem instances. The ratio $RQ_j^{\langle l,s\rangle}/S_T^{\langle l,s\rangle}$ has a small value for common skill levels among technicians. The value of χ_j gives a sign for the relative expertise requirement and the extreme case is a special task with $\chi_j = 1$.

Coverage

Task j is said to *likely cover* task k if the skills that are required by task k, but not by task j, are common within the technician group. Then there is a high possibility that the team performing task j can also perform task k. We define $\alpha_{jk}^{\langle l,s\rangle}$ as the pairwise comparison of skill requirements of tasks j and k for skill level $\langle l,s\rangle$. Two tasks with total duration not longer than a workday can be compared for coverage, otherwise they cannot be in the workload of the same team. If $RO_j^{\langle l,s\rangle}$ is positive and not less than $RO_k^{\langle l,s\rangle}$, a team processing task j can process task k as well, so $\alpha_{jk}^{\langle l,s\rangle} = 1$. If $RO_j^{\langle l,s\rangle}$ is strictly less than $RO_k^{\langle l,s\rangle}$, then $\alpha_{jk}^{\langle l,s\rangle}$ is found by calculating how common the skill $\langle l,s\rangle$ among technicians. Pairwise comparison for each skill level between task j and k is given as follows:

$$\alpha_{jk}^{\langle l,s\rangle} := \begin{cases} 0 & \text{if } RQ_j^{\langle l,s\rangle} = RQ_k^{\langle l,s\rangle} = 0 \\ 1 & \text{if } RQ_j^{\langle l,s\rangle} \geq RQ_k^{\langle l,s\rangle} \text{ and } RQ_j^{\langle l,s\rangle} \neq 0 \\ \frac{S_T^{\langle l,s\rangle}}{(S_T)_{\max}} & \text{if } RQ_j^{\langle l,s\rangle} < RQ_k^{\langle l,s\rangle} \end{cases}$$

where $(S_T)_{\max} = \max_{\langle l,s\rangle \in \mathbb{L}\times\mathbb{S}} \left\{ S_T^{\langle l,s\rangle} \right\}$. The coverage of task j over task k is determined according to the expression below:

$$\gamma_{jk} := \begin{cases} 1 & \text{if } \frac{\sum_{(l,s)} \alpha_{jk}^{\langle l,s\rangle}}{\sum_{\langle l,s\rangle} \operatorname{sign}(\alpha_{jk}^{\langle l,s\rangle})} > \varsigma \\ 0 & \text{otherwise} \end{cases}$$

After some experimentation, we decided to use the tuned value $\varsigma = 0.9$. The coverage value of task j is given below:

$$cov_j = \frac{\sum_{k \in J'} \gamma_{jk}}{\max\{|J'|, 1\}}, \quad J' = \{j' \in J | j' \neq j, d_{j'} + d_j \leq H\}. \tag{2.15}$$

Matching weight

For the definition of matching weights to be used in the IMM and in the FMM, we consider the domination properties of tasks within the workload. For example, if a task has long duration and large min-tech, then it can be counted as a dominant task. We call this "quantitative" dominance. Moreover, if a task requires expertise such that other tasks in the workload can also be processed by this expertise, then this is "qualitative" dominance. Quantitative dominance is expressed by $MT_j d_j$ and qualitative dominance by $h_j cov_j$. Our combinatorial algorithm treats tasks in non-increasing order of their weights. The matching weight of a task j is a combined measure of the following criteria: hardness h_j, coverage cov_j, min-tech MT_j, duration d_j, precedence relations, quantitative and qualitative dominance. The weight function in a general form we use is as follows

$$w_j = \varrho_1(\overline{h_j} + \overline{cov_j} + \overline{MT_j} + \overline{d_j}) + \varrho_2 \overline{MT_j}\,\overline{d_j} + \varrho_3 \sum_{k:j \to k} w(P(k)) \overline{MT_k}\,\overline{d_k} + \varrho_4 \overline{h_j}\,\overline{cov_j} \tag{2.16}$$

where "$-$" is used to interpret that all criteria are normalized. The values of coefficients ϱ_i, $i = 1, .., 4$, are determined in such a way that contributions of all expressions have the same order of magnitude. In the combinatorial algorithm, a task of high weight is selected with high probability.

2.5.3 Lower bounds

The second part of the preprocessing phase consists of computing lower bounds. We find lower bounds of our problem by solving a simplified problem which is formulated as a MIP model called the "lower bound model" (LBM). This method was first proposed by Hurkens (2009) and we slightly improve it to obtain tighter lower bounds values for instances with heterogeneous skill distribution. In the literature, several methodologies are encountered to find lower bounds for the RCPSP. Some of them are: linear programming based lower bounds (Mingozzi et al. (1998)), destructive approach (Heilmann and Schwindt (1997)), Lagrangian relaxation of the corresponding integer programming formulation (Möhring et al. (2003)) and destructive approach combined with constraint programming techniques (Brucker and Knust (2000)). Unfortunately, these techniques are not directly applicable to our problem because of its different aspects.

In the simplified problem of lower bound computing phase, precedence relations are relaxed, preemption is allowed, and outsourcing option of tasks is preserved. Skill

requirements are relaxed by Hurkens (2009), however in the LBM we consider skill requirements of tasks. This results in much tighter lower bounds in rare expertise instances. The relaxed problem amounts to finding the minimum time needed to meet the cumulative man-hour demand of tasks in every skill by considering technician availabilities. Table 2.1 shows the sets, the indices, the parameters, and the decision variables of the LBM.

Table 2.1: Sets, indices, parameters and variables of the LBM

Sets	
J	Set of tasks,
D	Set of days,
P	Set of priority classes,
$J(p)$	Subset of tasks in priority class p, $J(p) \in J, \forall p \in P$,
Indices	
p, p'	Priority class index, $p, p' \in P$
ν	Day index, $\nu \in D$
j, m	Task indices, $j, m \in J$
Parameters	
$w(p)$	Weight of priority class p, $w(p) \geq 0, \forall p \in P$
H	Length of one workday,
$Cmd_\nu^{\langle l,s \rangle}$	Cumulative man-hour availability in skill $\langle l, s \rangle$ until day ν,
$Amd_\nu^{\langle l,s \rangle}$	Man-hour availability in skill $\langle l, s \rangle$ on day ν
A_j	Outsourcing cost of task j
d_j	Duration of task j
π_p	Place of priority class p in priority permutation π,
B	Outsourcing budget
Variables	
C_p	Priority span of priority class p, $C_p \in Z_+$,
x_j	Binary variable indicating whether task j is outsourced
$z_{p\nu}$	Binary variable indicating whether C_p occurs on day ν
$\mu_{p\nu}$	Portion of day ν before C_p occurs

The LBM is used to compute the minimum aggregated times by selecting the best choice of tasks to outsource. Note that outsourcing decreases man-hour demand of tasks. The model assumes that priority classes are completed in a pre-specified order. The priority weights $\{28, 14, 4, 0\}$ suggest the ordering $\{1, 2, 3, 4\}$, however, we consider all priority permutations in practice: $\{1, 2, 3, 4\}$, $\{1, 3, 2, 4\}$, $\{2, 1, 3, 4\}$, $\{2, 3, 1, 4\}$, $\{3, 1, 2, 4\}$, $\{3, 2, 1, 4\}$.

The last completion time in a priority class is called "priority span" and it is denoted by C_p for priority class p. Priority spans are assumed to be in the order specified by the priority permutation. Let π_p denote the place of the priority class p in the permutation. So in the LBM, it is assumed that $C_p \leq C_{p'}$ for all $\langle p, p' \rangle$ such that $\pi_p < \pi'_p$. The objective of the LBM is the same as our scheduling problem. In the LBM, the time of the day when priority span of priority class p occurs is captured by two decision variables: a binary variable that indicates the day of the priority span and a continuous variable whose value tells us the portion of the span day that stays within the priority span.

$$\text{minimize} \quad \sum_{p \in P} w(p) C_p$$

subject to:

$$C_p \leq C_{p'} \qquad ,\forall \langle p, p' \rangle \in P : \pi_p + 1 = \pi_{p'} \tag{2.17}$$

$$z_{p\nu} \geq \mu_{p\nu} \qquad ,\forall p \in P, \forall \nu \in D \tag{2.18}$$

$$\sum_{\nu \in D} z_{p\nu} = 1 \qquad ,\forall p \in P \tag{2.19}$$

$$\sum_{\nu \in D} \{H\nu z_{p\nu} + H\mu_{p\nu}\} \leq C_p \qquad ,\forall p \in P \tag{2.20}$$

$$\sum_{\nu \in D} \{Cmd_{\nu}^{\langle l,s \rangle} z_{p\nu} + Amd_{\nu}^{\langle l,s \rangle} \mu_{p\nu}\} \geq \sum_{j \in J(p)} RQ_j^{\langle l,s \rangle} d_j (1 - x_j), \forall \langle l, s \rangle \in \mathbb{L} \times \mathbb{S}, \forall p \in P \tag{2.21}$$

$$x_m \leq x_j \qquad ,\forall m \to j, \forall j \in J \tag{2.22}$$

$$\sum_{j \in J} A_j x_j \leq B \tag{2.23}$$

$$x_j, z_{p\nu} \in \{0, 1\},\ \mu_{p\nu} \geq 0, C_p \in Z_+ \qquad ,\forall p \in P, \forall \nu \in D \tag{2.24}$$

The priority span ordering of successive priority classes in priority permutation is ensured by constraint (2.17). The relations of the priority span day variables are given in constraints (2.18) and (2.19). In constraint (2.20), it is ensured that the priority spans cover the time that is pointed by two dedicated decision variables. By constraint (2.21), the time pointed for priority span is enough to meet the manhour requirements of tasks. Our convention of "outsourcing the successors of outsourced tasks" is embedded into the LBM by constraint (2.22). Finally, the outsourcing budget constraint is given by (2.23).

Due to the fact that any of them may turn out to include the optimum schedule depending on the number of tasks in each priority class, lower bound values of all priority permutations are computed. In schedule construction, priority classes are handled sequentially. The tasks are considered in the order specified by the priority permutation. Priority permutation with smallest lower bound value is used first to construct schedules, and then the one with second smallest lower bound value and so on. Priority permutations with lower bound values greater than or equal to the lowest cost of constructed schedules are neglected.

2.5.4 Constructing alternative schedules

First of all we give the following definition:

Definition 2.5.2. *(Candidate task)*
If immediately scheduling of a task does not violate the schedule feasibility, that task is called a "candidate task".

Initial matching

In the initial matching, we create a complete bipartite graph $G_{IMM} = (T', J', E)$ where $T' \subseteq T$, $J' \subseteq J$. An illustration of the IMM model is shown in Figure (2.2). On the bipartite graph $G_{IMM} = (T', J', E)$, we want to find a many-to-one type assignment and here the term "matching" refers to this assignment. We denote this matching by $M \subseteq E$. Then the feasibility of technician allocation in the matching M is expressed by

$$|\{\{t, j\} \in M | j \in J'\}| \leq 1, \quad \forall t \in T' \tag{2.25}$$

and the skill requirements of the candidate tasks by

$$RQ_j \leq \sum_{t \in T' : \{t,j\} \in M} S_t, \quad \forall j \in J' \tag{2.26}$$

In fact, the matching M is an initialization of the workday schedule. The initialized teams are determined by

$$T(\tau_j) = \{t \in T' | \{t, j\} \in M\}, \quad \forall j \in J' \tag{2.27}$$

where $T(\tau_j)$ denotes the team of technicians performing task j.

The MIP model formulation. The problem of finding a matching, as explained above, is a hard problem, due to the skill requirements. Therefore, we formulate the problem of finding the aforementioned matching on the bipartite graph $G_{IMM} = (T', J', E)$ as a MIP model, the IMM. In the IMM, the binary decision variables x_{tj} for every $\langle t, j \rangle \in T' \times J'$ indicate whether the edge $\{t, j\}$ is in the matching or not. The binary decision variables y_j for every $j \in J'$ indicates whether an incident edge to job j is the matching. Every candidate task has a weight that is calculated as in (2.16). The objective of the IMM is maximizing the weighted team initializations.

Solutions of the IMM. In our scheduling context, the solution value $x_{tj} = 1$ corresponds to the case that t is *assigned* to job j, and $y_j = 1$ means that j *initializes a team.*

Handling large instances: In principle we aim to find the optimal initial matching on the complete set of available technicians and candidate tasks. This corresponds to $T' = T$ and $J' = J$. However, in large instances, one run of the IMM takes longer than the desired time if the number of the candidate tasks is high. Therefore, we settle for a heuristic solution by repeatedly applying the IMM to a subset of candidate tasks at a time.

Table 2.2 shows the notation, the parameters and the decision variables of the IMM. The mathematical formulation of the IMM follows Table 2.2.

Table 2.2: Sets, indices, parameters, and variables of the IMM

Sets	
J'	Set of tasks, $J' \subseteq J$
T'	Set of technicians, $T' \subseteq T$
Indices	
j	Task index, $j \in J'$
t	Technician index, $t \in T'$
Parameters	
w_j	Matching weight of task j
$S_t^{\langle l,s \rangle}$	Equal to 1 if technician t is skills in domain s at level l, otherwise 0
$RQ_j^{\langle l,s \rangle}$	Number of skilled technicians required by task j in domain s at level l
Variables	
x_{tj}	Binary variable indicating whether technician t is assigned to task j
y_j	Binary variable indicating whether task j initializes a team

$$\text{maximize} \quad \sum_{j \in J'} w_j y_j$$

subject to:

$$\sum_{j \in J'} x_{tj} \leq 1 \qquad \forall t \in T' \tag{2.28}$$

$$\sum_{t \in T'} S_t^{\langle l,s \rangle} x_{tj} \geq RQ_j^{\langle l,s \rangle} y_j \quad \forall j \in J', \forall \langle l,s \rangle \in \mathbb{L} \times \mathbb{S} \tag{2.29}$$

$$y_j, x_{tj} \in \{0,1\} \qquad \forall j \in J', \forall t \in T' \tag{2.30}$$

Constraints (2.28) enforce that a technician can be matched to at most one task. If a task initializes a team, then constraint (2.29) ensures that its skill requirements are met. Note that a task with a relatively high weight may get more technicians than necessary. Although it might seem as an inefficiency, in the schedule extension phase, the FMM dynamically reallocates the technicians to obtain an efficient packing of tasks. When no technician is left for assigning to tasks or all candidate tasks are assigned to technicians, the initial matching is completed. In the initialized workday schedule, all teams have one task in their workloads.

We solve the IMM models during schedule construction by using the solver CPLEX 12.0.

Figure 2.2 illustrates team initializations with an example of five technicians and three tasks. In the illustrated solution of the IMM, it turns out that two teams are initialized by j_1 and j_3. In the partial schedule, team τ_{j1} (τ_{j2}) has a load with duration d_1 (d_3). The initialized teams have the following technicians $T(\tau_{j1}) = \{t_4, t_2\}$ and $T(\tau_{j2}) = \{t_1\}$.

Next, we extend the workday schedule by adding more candidate tasks into it. Assigning technicians to candidate tasks via a matching has a more global view of all assignments than finding technicians for one task at a time. This advantage is more remarkable in the instances with heterogeneous skill distribution.

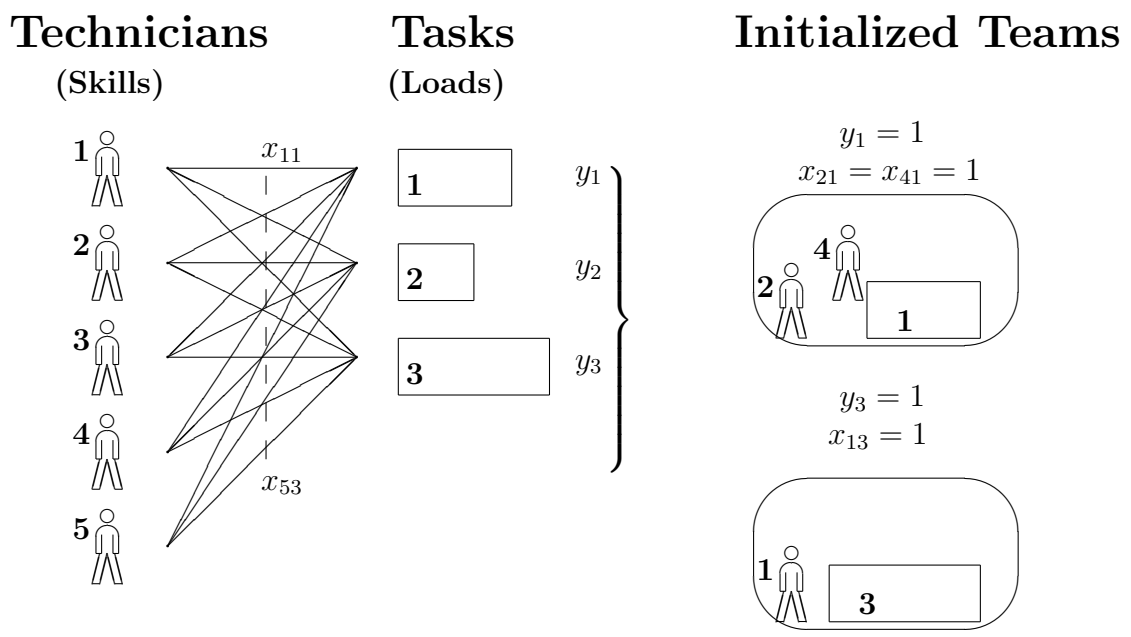

Figure 2.2: Initializations of teams in the IMM

Extending the partial day schedule

Once the teams are initialized in the initial matching, the partial workday schedule is extended by inserting more tasks. Adding more tasks to the schedule is not trivial, since some tasks may require slightly or even completely different skills than the ones already scheduled. Therefore, as technicians can be exchanged easily among the teams, the excess skills in teams that have filled their workloads can be taken out and given to the teams that have some space in their workloads to schedule more candidate tasks. The FMM exchanges technicians among the teams by making them *conditionally* available.

The need for flexibility in the FMM. The initial matching is of course a local decision if all candidate tasks are considered. Therefore, its solutions may have some drawbacks due to the local decisions. Below we give two cases:

- If two similar hard tasks initialize two different teams, but it seems better to have them in one team's workload. Merging these hard tasks as one workload of a team will lead the plenty technicians to contribute to other teams.

- If somehow an expert is assigned to a task, but he may use his skills more efficiently by processing another task. This accidently assigned technician can be taken out from his current team.

Unifying the concepts: *from "technicians and tasks" to "skills and loads".* In a partial workday schedule, a team has two features, its technicians and its workload. First, a team can be perceived as a combination of *skills* if technicians are considered. Second, a team can also be perceived as a *load* if its workload is considered. Therefore

from the first (second) point of view a team represents a *skill* (*load*). Moreover unassigned individual technicians (candidate tasks) can be perceived as *skills* (*loads*). These observations lead us to consider every item of the incomplete workday schedule either as "skill" or as "load". As a result of this unification of concepts, we have skills and loads in the FMM instead of technicians and tasks. The types of skills and loads depend on what they originate from, and their definition are given as

Definition 2.5.3. *(Active skill)*
The total skill of a team is called an "active skill" and $\mathbb{T}$ *denotes the set active skills.*

Definition 2.5.4. *(Active load)*
The workload of a team is called an "active load" and $\mathbb{W}$ *denotes the set active loads.*

Note that the active skill of team τ is given by $S_\tau = \sum_{t \in T(\tau)} S_t$ and the active load of τ is found by $RQ^{\langle l,s \rangle}_{J(\tau)} = \max_{j \in J(\tau)} RQ^{\langle l,s \rangle}_j$.

Definition 2.5.5. *(Latent skill)*
The skill of a technician $t \in T(\tau)$ *such that* $RQ_{J(\tau)} \in \mathbb{W}$ *is called "latent skill".*

Definition 2.5.6. *(Passive skill)*
The skill of a technician t *such that* $\{\tau \in \Xi | t \in T(\tau)\} = \emptyset$ *is called "passive skill".*

Definition 2.5.7. *(Passive load)*
The load of a candidate task is called a "passive load".

The underlying bipartite graph of the FMM. In a flexible matching, we firstly choose a subset of teams $\Xi' \subseteq \Xi$ from the initialized workday schedule. Here, we use the set Ξ to denote the teams of the initialized workday schedule, not of the completed workday schedule. Moreover, we choose again a subset of teams $\Xi'' \subseteq \Xi'$ whose loads will be active in the FMM. Lastly, we chose a set J' of candidate tasks. To sum up, in the FMM the teams in Ξ' have active skills, the teams in Ξ'' have active loads and latent skills besides their active skills, and the candidate tasks in J' have passive loads. Then, we create a complete bipartite graph $G_{FMM} = (\Delta, \Lambda, E)$ where the partition Δ includes the skills and the partition Λ includes the loads. Table 2.3 specifies the content of the partitions of G_{FMM} and it also lists the edges in E that are *not* included in the FMM model. Note that we have $|\Xi''| = |\mathbb{W}| \leq |\Xi'| = |\mathbb{T}|$.

On the bipartite graph $G_{FMM} = (\Delta, \Lambda, E)$, we want to find a many-to-one type assignment that we call a matching $M \subseteq E$. We note that having different types of edges in E in a solution results in different type of extensions in the partial workday schedule. These extensions are summarized as follows:

- *Edges of type* $\{S_\tau, RQ_{J(\tau')}\}$: The workloads of teams τ and τ' are *merged*. In this merging, tasks sequences are intertwined in such a way that the orders in previous sequences are preserved and precedence relations are respected. The combined workload is processed by the team $T(\tau)$. Such an extension is called *merging*.

Table 2.3: The properties of the bipartite graph G_{FMM}

Skills in partition Δ	
Active skills ($\mathbb{T}$)	$\{S_\tau \mid \tau \in \Xi'\}$
Passive skills	$\{S_t \mid t \in T \setminus \cup_{\tau \in \Xi'} T(\tau)\}$
Latent skills	$\{S_t \mid t \in \cup_{\tau \in \Xi''} T(\tau)\}$
Loads in partition Λ	
Active loads ($\mathbb{W}$)	$\{RQ_{J(\tau)} \mid \tau \in \Xi''\}$
Passive loads	$\{RQ_j \mid j \in J'\}$
Edges in E that are forbidden in the FMM	
Edges of the same teams	$\{\{S_\tau, RQ_{J(\tau')}\} \in \Delta \times \Lambda \mid \tau = \tau'\}\}$
Edges with total load greater than H	$\{\{S_\tau, RQ_{J(\tau')}\} \in \Delta \times \Lambda \mid \sum_{j \in J(\tau) \cup J(\tau')} d_j > H\}$
	$\{\{S_\tau, RQ_j\} \in \Delta \times \Lambda \mid \sum_{j' \in J(\tau)} d_{j'} + d_j > H\}$
Edges that result:	Violation of precedence relations

- *Edges of type* $\{S_\tau, RQ_j\}$: The workload of team τ is *extended* by adding candidate task j. The priority class and precedence relations of task j determine its place in the tasks sequence of the extended team τ. Such an extension is called *extending teamload*.

- *Edges of type* $\{S_t, RQ_{J(\tau')}\}$: The assigned skills form a *new* team to process the workload of team τ'. Clearly, the new skills meet the skill requirements $RQ_{J(\tau')}$. Such an extension is called *recombining technicians*.

- *Edges of type* $\{S_t, RQ_j\}$: If the *passive load* of candidate task j is matched with some *passive* and/or *latent skills*, a new team is formed that has the passive load j in its workload. Such an extension is called *initializing a team.*

Handling large instances: We construct alternative schedules by varying the size of active teams such that $|\Xi''| = |\mathbb{W}| \in \{3, 4, 5, 6, 7, 8\}$. In our computational results, for small-size and medium-size instances, corresponding to instances roughly with $|T| \leq 50$ and $|J| \leq 500$, we used all possible size of active teams. However, as the instance size grows, we can construct schedules only with $|\mathbb{W}| = 3$.

In a matching, if the active load of a team is matched with a new combination of skills, then team's latent skills, not included in the new skill combination, become free to be assigned to other loads. Thus latent skills are *conditionally* available for matchings and they play an important role in scheduling candidate tasks (passive loads) by determining skill combinations of teamloads flexibly.

Sequencing decisions

Case *merging*: Merging is the most complicated extension for sequencing decisions. Once an edge of type $\{S_\tau, RQ_{J(\tau')}\}$ is selected in a solution, the tasks of teams τ and τ' are interwind in a way that the precedence relations of all tasks remain satisfied and

the tasks in higher priority classes are completed as early as possible. The merging of workloads of teams τ and τ' starts with inserting of the first task in $J(\tau')$ into $J(\tau)$. Then second task of $J(\tau')$ is inserted and so on. In fact, the workload of τ is extended by inserting one task at a time during merging. Therefore sequencing in "merging" $J(\tau')$ and $J(\tau)$ uses the subroutine of "extending teamload" $|J(\tau')|$ times. The steps of sequencing decision in merging are given in Table 2.4.

Table 2.4: Sequencing decision in merging

Input: Tasks $J(\tau)$ and $J(\tau')$ with orderings π and π'
1: Initialize i=1;
2: **while** $i \leq |J(\tau')|$ **do**
3: Select $j \in J(\tau')$ with $\pi'_j = i$;
4: Insert$(j, J(\tau))$;
5: $i \leftarrow i + 1$;
6: **end**

Insert$(j, J(\tau))$

Initialize $k = |J(\tau)|, \iota = k$;
I1: **if** $Pred(j) = \emptyset$ and $\{j^\circ \in J | j \in Pred(j^\circ)\} = \emptyset$ **then**
I2: $k \leftarrow k - 1$;
I3: **end**
I4: **while** $\iota = k$ **do**
I5: Select $j^* \in J(\tau)$ with $\pi_{j^*} = k$
I6: **if** $\max_{j^\circ \in Pred(j)}\{CT_{j^\circ}\} > CT_{j^*} - d_{j^*}$ **then**
I7: $k \leftarrow k - 1$;
I8: **else if** $\min_{j^\circ \in \{j'' \in J | j^* \in Pred(j'')\}}\{CT_{j^\circ} - d_{j^\circ}\} < CT_{j^*} + d_j$ **then**
I9: $k \leftarrow k - 1$;
I10: **else if** $P(j) > P(j^*)$ **then**
I11: $k \leftarrow k - 1$;
I12: **else**
I13: $k \leftarrow k - 1$ and $\iota \leftarrow k$;
I14: **end**
I15: **end**
I16: $k \leftarrow |J(\tau)|$;
I17: **for** $k \geq \iota$ **do**
I18: Select $j^* \in J(\tau)$ with $\pi_{j^*} = k$;
I19: $\pi_{j^*} \leftarrow k + 1$;
I20: **end**
I21: $\pi_j \leftarrow \iota$;

The steps $1 - 6$ include the procedure selecting one task from $J(\tau')$ and inserting it into $J(\tau)$. In the **Insert** function, we decide where the task under consideration should be inserted in the workload of team τ. Let task j is to be inserted into $J(\tau)$.

If j has no precedence relations, then we make it the last task in $J(\tau)$. Otherwise, we make some comparisons of every task in $J(\tau)$, starts from the last one and proceeds to the first one, unless a decision is made. Let task j^* be the task being questioned for a late start time. In step I6, we check whether task j has a predecessor with start time greater than the start time of j^*. If this is the case, we place task j after task j^*. Then we check whether task j^* has a successor with start time earlier than $CT_{j^*} + d_j$ (Step I8). This implies that precedence relations of task j^* will be violated, if it starts after task j, so we place task j after task j^*. Finally, if letting task j start before task j^* will not violate any precedence relations, in step I10 we check urgency of both tasks. We let the task that is more urgent than the other one start earlier.

On the solutions of the FMM. In the FMM, a candidate task can be matched with "*any*" combination of skills provided that every teamload in the model either keeps its skills or finds a new skill combination to stay being processed. This is the key aspect of our matching model resulting in high flexibility. Each technician, no matter in a team or not, becomes a potential skill for candidate tasks. Candidate tasks may be added to the partial schedule by joining a teamload or by initializing a team. While joining to a teamload, a candidate task may bring some additional technicians to the team, if necessary.

Latent skills can contribute to other matchings if and only if active loads of their teams are matched with new skill combinations and remain processed. In other words, matching an active load with a new skill combination creates the opportunity of using some of its technicians, latent skills, in other matchings. The only reason why such new assignments are found is simply the desire to match as many as passive loads of candidate tasks in order to extend the partial workday schedule. The objective of the FMM is weighted sum of passive loads matchings, therefore a passive load with a high matching weight has the power to force the current partial schedule to make the needed latent skills available. In light of this fact we have the following observation:

Observation 2.5.1. *A candidate task with sufficiently high weight may force teams to merge or recombine their technicians, thereby making an expert technician available.*

Mathematical formulation of the FMM

The many-to-one type skill-load assignment problem is formulated as a MIP model, the FMM. In the FMM, we use binary decision variables $x_{\sigma\lambda}$ to indicate that the corresponding edge is chosen in the assignment. The index σ belongs to the set Δ of skills and the index λ belongs to the set Λ of loads. The sets, indices, parameters and variables in the formulation of the FMM are listed in Table 2.5.

Edges causing inefficiency: As mentioned before, a subset of teams ($\Xi' \subseteq \Xi$) in the partial day schedule is included in the FMM. An edge that may seem profitable on the bipartite graph for a specific FMM model, may not be so for the whole partial day schedule. An example is an edge between an active skill and a passive load such that the number technicians of the team of the active skill has more technicians than needed for the passive load. The FMM uses inefficiency decision variables to detect and prevent selecting such edges for assignments.

Table 2.5: Sets, indices, parameters and variables of the FMM

Sets	
Δ	Set of skills
Λ	Set of loads
$\mathbb{T}$	Set of active skills, $\mathbb{T} \subset \Delta$
$\mathbb{W}$	Set of team loads, $\mathbb{W} \subset \Lambda$
$\mathbb{F}$	Forbidden edges, $\mathbb{F} \subseteq \Delta \times \Lambda$
$\mathbb{M}$	Set of edges joining active skill and load of a team, $\mathbb{M} = \{\{S_\tau, RQ_{J(\tau')}\} \mid \tau = \tau'\}$
Indices	
σ, σ'	Skill indices, $\sigma, \sigma' \in \Delta$
λ, λ'	Load indices, $\lambda, \lambda' \in \Lambda$
Parameters	
$L(\sigma)$	Set of latent skills of active skill σ, $\sigma \in \Delta$
w_λ	Weight associated to load λ, $\lambda \in \Lambda$ (Note: If $\lambda \notin \mathbb{W}$, then $w_\lambda = w_j$ where j is candidate task, otherwise $w_\lambda = 0$)
mt_λ	Number of technicians used for performing all tasks in load λ, $\lambda \in \Lambda$
$S_\sigma^{\langle l,s \rangle}$	Skill value of σ in domain s at level l
$RQ_\lambda^{\langle l,s \rangle}$	Skill requirement of λ in domain s at level l
Variables	
$x_{\sigma\lambda}$	Binary variable indicating whether σ is assigned to λ, $\langle \sigma, \lambda \rangle \in \mathbb{M}$
y_λ	Binary variable indicating whether λ is assigned to a skill combination, $\lambda \in \Lambda$
ζ_λ	Inefficiency penalty of λ, $\lambda \in \Lambda$

$$\text{maximize} \quad \sum_{\lambda} w_\lambda y_\lambda - \zeta_\lambda$$

subject to:

$$\sum_{\lambda} x_{\sigma\lambda} \leq 1, \qquad \forall \sigma \in \Delta \tag{2.31}$$

$$\sum_{\sigma \in \mathbb{T}} x_{\sigma\lambda} \leq 1, \qquad \forall \lambda \in \Lambda \tag{2.32}$$

$$x_{\sigma\lambda} = 0, \qquad \forall \langle \sigma, \lambda \rangle \in \mathbb{F} \tag{2.33}$$

$$\sum_{\lambda'} x_{\sigma\lambda'} + y_\lambda \leq 1, \qquad \forall \langle \sigma, \lambda \rangle \in \mathbb{M} \tag{2.34}$$

$$\sum_{\sigma' \in L(\sigma)} \sum_{\lambda'} x_{\sigma'\lambda'} \leq |L(\sigma)| y_\lambda, \qquad \forall \langle \sigma, \lambda \rangle \in \mathbb{M} \tag{2.35}$$

$$\sum_{\sigma} S_\sigma^{\langle l,s \rangle} x_{\sigma\lambda} \geq RQ_\lambda^{\langle l,s \rangle} y_\lambda, \qquad \forall \langle l, s \rangle \in \mathbb{L} \times \mathbb{S}, \forall \lambda \in \Lambda \tag{2.36}$$

$$\sum_{\sigma \in \mathbb{T}} |L(\sigma)| x_{\sigma\lambda} + \sum_{\sigma \in \Delta \setminus \mathbb{T}} x_{\sigma\lambda} - mt_\lambda \leq \zeta_\lambda, \qquad \forall \lambda \in \Lambda \tag{2.37}$$

$$x_{\sigma\lambda}, y_\lambda \in \{0, 1\}, \zeta_\lambda \geq 0, \qquad \forall \langle \sigma, \lambda \rangle \in \mathbb{M}, \forall \lambda \in \Lambda \tag{2.38}$$

A skill can be assigned to at most one load (constraints (2.31)). A load may be matched to at most one active skill according to constraints (2.32). It can be matched to any number and any combination of passive and latent skills though. Having constructed the set $\mathbb{F}$ in advance, edges in $\mathbb{F}$ are forbidden by constraints (2.33).

In the FMM, a team has three contributions: an active skill $\sigma \in \mathbb{T}$, latent skills $L(\sigma) \in \Delta$, and a teamload $\lambda \in \mathbb{W}$. According to constraints (2.34), a team can contribute to extending the day schedule in one of the following ways: either its active skill is matched to a load or its active load is matched to skills. In the former case, the matched load is added to team's workload and technicians of the team stay together. Some additional technicians may join to team as well, if some other latent or passive skills are also assigned to matched load. In the latter case, a skill combination is assigned to active load (constraints (2.36)) and latent skills of the team may be used in other matchings (constraints (2.35)). Inefficiency variables z_λ are used to prevent assigning an unnecessarily high number of technicians to a skill (constraints (2.37)). Here $mt_\lambda = MT_j$ for a passive load λ with candidate task j and $mt_\lambda = |T(\sigma)|$ for an active load λ currently assigned to σ. For instance, if a load of a candidate task with $MT = 2$ is assigned to a team skill of 5 technicians, then this assignment is penalized by constraints (2.37) on the value of ζ_λ. Note that inefficiency is not forbidden in the FMM, but discouraged by penalizing.

The objective function is the sum of the weights of selected loads and the inefficiency drop. The candidate tasks contribute to the objective by their weights and influence the allocation of skills among teams. Note that a task with sufficiently high weight can even cause some inefficiency to get scheduled. Both the IMM and the FMM are implemented in Java and they are solved by using CPLEX 12.0.

Illustrative examples. Figure 2.3 and Figure 2.4 illustrate examples of extending the partial schedule of the example in Figure 2.2. In order to show all possible cases, we illustrated two different scenarios corresponding to two different solutions obtained with different parameter settings.

In Figure 2.3, and also in Figure 2.4, the FMM includes 2 teams (τ_1, τ_2) with technicians $T(\tau_1) = \{t_4, t_2\}$ and $T(\tau_2) = \{t_1\}$, 2 unassigned technicians (t_3, t_5) and one candidate task (j_2). In both figures, latent skills of the same team are encircled and connected to their current teams. The first two vertices of (left) right partition are active (skills) loads. There is no edge drawn between an active skill and an active load of the same team, since this matching does not make any sense and it is forbidden in the model.

In scenario 1 (Figure 2.3), the active load of τ_2 is matched to a new skill combination including the active skill of τ_1 and passive skill of t_5. This matching is an example of merging and the combined load is performed by the newly matched skill combination. Matching the active load of τ_2 allowed t_1 to be in the initialized team of j_2 together with technician t_3.

In scenario 2 (Figure 2.4), τ_1 recombined its technicians by being matched to passive skills of unassigned technicians t_3, t_5 and latent skill of technician t_2 who was in the previous combination as well. The number of technicians of τ_1 seems increased by one and this increase can be explained as an adjustment to have technician t_4 in

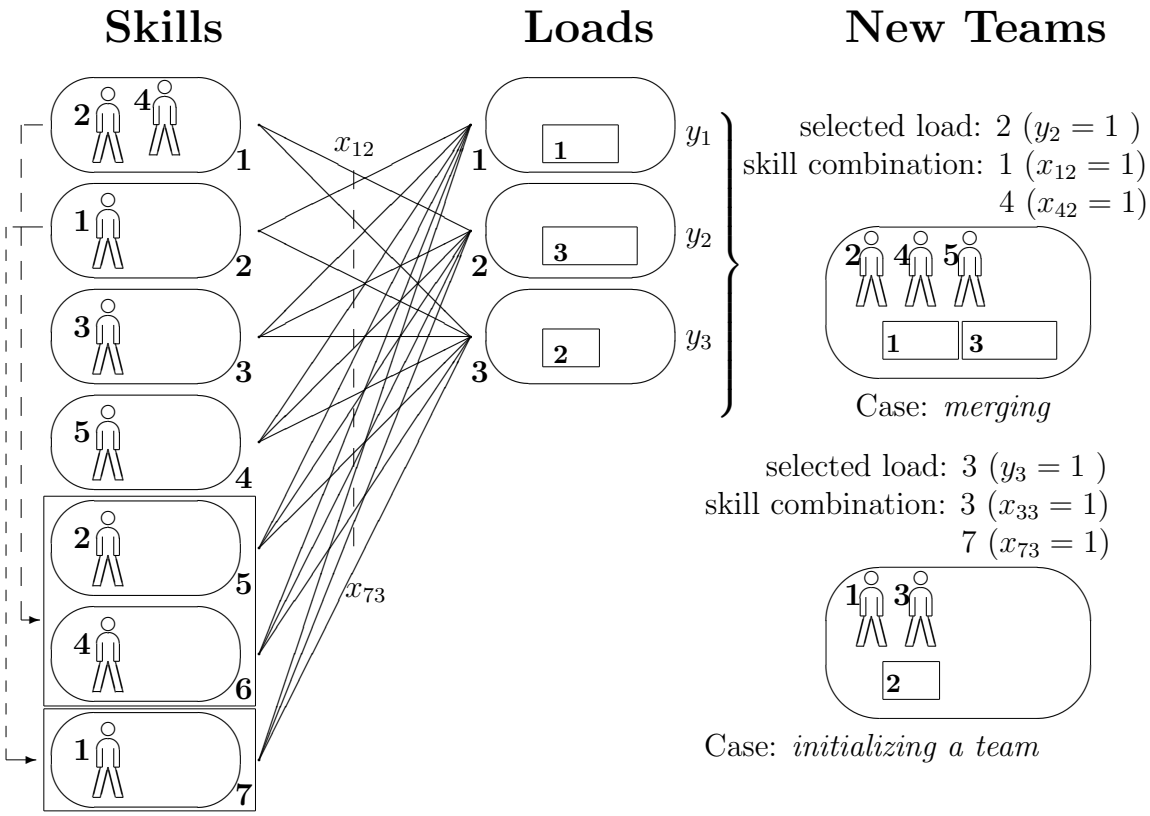

Figure 2.3: Extending partial schedule with the FMM (Scenario 1)

the extended team τ_2 as we now have j_2 in one load.

Due to its mentioned flexibility aspects, we call the bipartite matching model the *flexible matching model*, or the FMM. The main contribution of this study is the introduction of this model. Hurkens (2009) extends the partial day schedule using two different matching models iteratively, where the first one is used to assign multiple tasks simultaneously, and the second one is used to find efficient recombination of technicians. The FMM carries out these two steps simultaneously.

Parameters and strategies to construct different schedules

In this section we explain the strategies applied to find different schedules. We have observed in experiments that each of those strategies may lead to a best solution.

Efficiency in Initial Matching: When the formulation of the FMM is carefully examined, it is not difficult to see that if $\mathbb{T} = \emptyset$, the FMM boils down to the IMM plus efficiency constraints. So one of our strategies is adding efficiency constraints to the Initial Matching. This strategy is especially beneficial in instances where the average duration of tasks is close to workday length.

Fully loaded teams: While extending the partial day schedule some teams may reach a workload with total duration of a workday. As a strategy, we include a few of those fully loaded teams in the FMM with the hope to lower their skills excess by making some of their technicians conditionally available for candidate tasks. Fully loaded teams can contribute to extending partial schedule by recombining their technicians.

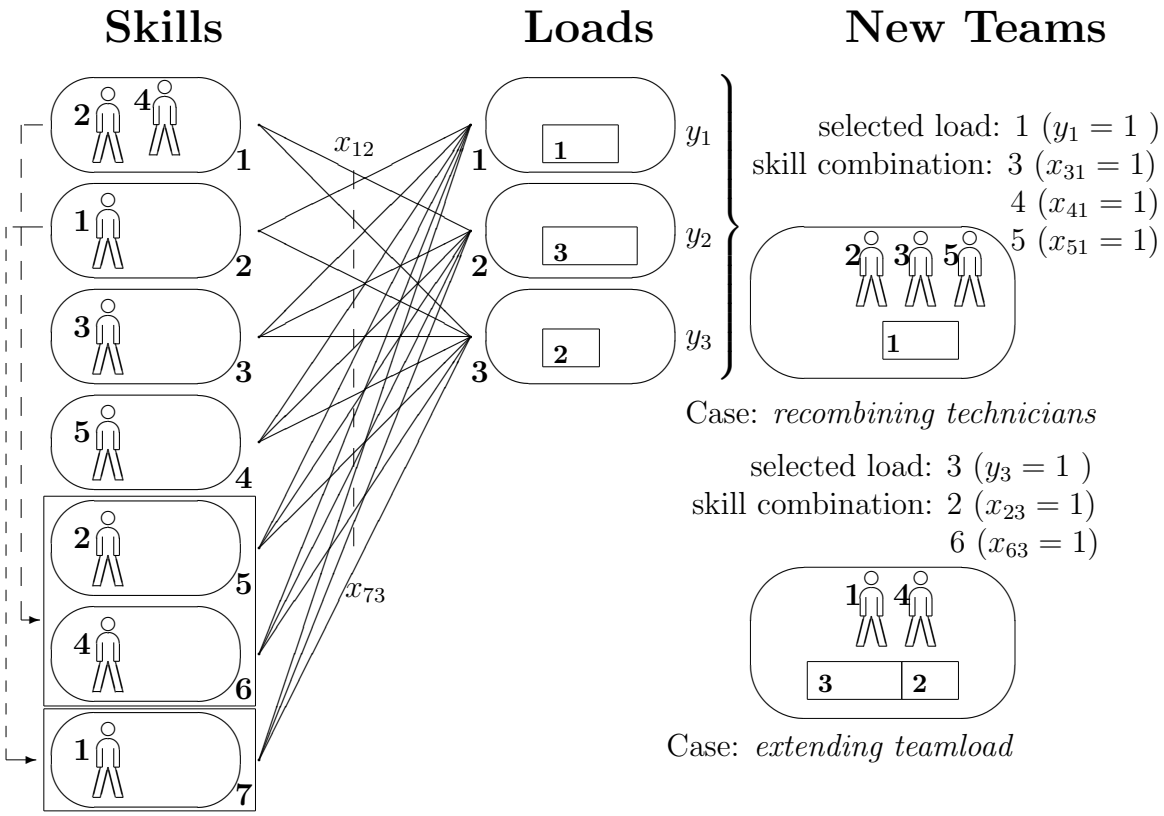

Figure 2.4: Extending partial schedule with the FMM (Scenario 2)

Number of teams in the FMM: In order to obtain the optimal solution of the FMM in a reasonably short time, a fixed number of teams is included. So we have $|\mathbb{T}| = |\mathbb{W}| \in \{3, 4, 5, 6, 7, 8\}$. As long as the time limit allows, we construct schedules for each fixed number. In large instances, solving the FMM models takes longer time, therefore a few numbers in $\{3, 4, 5, 6, 7, 8\}$ can be used to construct different schedules.

Selection of candidate tasks: Candidate tasks are included in the FMM in sequential order according to their priority classes. As a strategy we allow some of the special tasks in succeeding priority classes to be in the FMM. This way we aim to avoid the delays of the priority makespan due to rare expertise.

2.6 Computational results

2.6.1 On the rare expertise

Three sets of problem instances were provided by France Telecom in the ROADEF Challenge 2007. The descriptive statistics of the instances can be seen in Table 2.6. In the instance sets, the number of skill domains (levels) varies from 3 to 40 (2 to 7). Data set A was released in the first stage of the challenge for participants to start implementing their solution approaches. Data set B was released for fine tuning and the data set X was used for the final evaluation. Instances in the set A are smaller than the instances of B and X. There is no significant difference between data set B and X in terms of number of technicians and number of tasks. However the number of

special tasks is the point where they differ. (See section 2.5.1 for definition of special task.)

The costs of schedules constructed by the FMM and other heuristics are given in Table 2.7. We used the time limit of 20 minutes, as specified in the ROADEF Challenge 2007 and all results are obtained on a laptop with Intel Core 2 Duo 1.6 GHz Processor, 4GB RAM. The MIP models, the IMM and the FMM, are solved using the solver CPLEX 12.1.0. Table 2.8 shows the number of schedules constructed for each problem instance and the time needed to construct these schedules. The first column in Table 2.7 shows the problem instances and the next four columns report the results found by the FMM, Hurkens (2009), Cordeau et al. (2010) and Estellon et al. (2009). In each of these columns, the first entries are schedule costs and second entries are the relative difference in percentage that is defined as the difference of a schedule cost to the best schedule cost ever found (best schedule costs are listed in the column with title "BEST"). In the column "BEST", we report the lowest schedule costs by considering the results of the ROADEF Challenge 2007 as well. The last column, labeled "LB", lists lower bound values of instances.

It is seen in the last column of Table 2.6 that data set X includes instances with a higher number of special tasks. The average number of special tasks of the instance groups A, B and X are 1.3, 2.4 and 7.2 respectively. This hints that data set X instances have a heterogeneous skill distribution among technicians and therefore rare expertise is observed. The challenge in the instances of rare expertise can be realized by checking the gap between the best found schedules and lower bounds. In Table 2.7 we see that the gap between best schedules and lower bounds is smaller in data sets A and B compared to data set X. This may show either the weakness of lower bounds or the case that the approaches are not successful in handling rare expertise or both.

In the final evaluation of ROADEF Challenge 2007, 7 best schedule costs (out of 10 instances in set X) were due to Hurkens (2009). This shows that the solution approach of Hurkens (2009) was promising for cases of rare expertise. If the column BEST is examined in Table 2.7, it is seen that the FMM found 8 best schedules in data set X. In our opinion, this is the result of the flexibility in exchanging the technicians among teams and in diverse extension options while constructing day schedules. This flexibility leads to more efficient packing of special tasks and higher utilization of experts in instances with heterogenous skill distribution. In the instances X9 and X10, the gap between best schedule and lower bound has been decreased remarkably. As an improved version, the FMM found better schedules in all X instances compared to Hurkens' results.

If all data sets are considered, it seems reasonable to conclude that data sets B and X are better representatives of the real case instances due to the high number of technicians, tasks and skill domains. Moreover rare expertise is also a situation companies encounter in their operations. In instances with a small number of technicians, tasks and skill domains, schedule costs are sensitive to individual assignments, whereas in large instances the number of feasible schedules are high and the schedules are not sensitive to individual assignments. Therefore in our opinion, large instances are better to test the reliability of the algorithms.

Table 2.6: Problem instances A, B and X

	Data set A					Data set B					Data set X				
Instance	$\|T\|$	$\|J\|$	$\|S\|$	$\|L\|$	$\|SP\|$	$\|T\|$	$\|J\|$	$\|S\|$	$\|L\|$	$\|SP\|$	$\|T\|$	$\|J\|$	$\|S\|$	$\|L\|$	$\|SP\|$
1	5	5	3	2	0	20	200	4	4	0	60	600	15	4	27
2	5	5	3	2	0	30	300	5	3	0	100	800	6	6	0
3	7	20	3	2	1	40	400	4	4	0	50	300	20	3	0
4	7	20	4	3	0	30	400	40	3	15	70	800	15	7	0
5	10	50	3	2	1	50	500	7	4	9	60	600	15	4	13
6	10	50	5	4	5	30	500	8	3	0	20	200	6	6	3
7	20	100	5	4	1	100	500	10	5	0	50	300	20	3	0
8	20	100	5	4	0	150	800	10	4	0	30	100	15	7	5
9	20	100	5	4	3	60	120	5	5	0	50	500	15	4	10
10	15	100	5	4	2	40	120	5	5	0	40	500	15	4	14

SP: Special Tasks, $SP \subseteq J$

2.6.2 On the performances of heuristics

It is remarkable that the FMM has an average difference with the best available solution of 0.9% in data set X whereas it is 7.1%, 13.7% and 15.8% for Hurkens (2009), Cordeau et al. (2010) and Estellon et al. (2009) respectively. The results of Cordeau et al. (2010) are the best ones in data set A, however their average distance increases from A to B and from B to X. Hurkens (2009) has an increase from A to B and stays almost at the same level from B to X. The FMM starts with a high average distance in data set A and draws a slight increase from A to B. In sets B and X, it performs remarkably well. The superior performance in set X shows that in case of rare expertise, the FMM can find compact schedules and experiences less skill excess in assignments. Particularly, the instances X1, X5, X9 and X10 are sensitive to expert availabilities due to the high number of special tasks. Estellon et al. (2009) also underline those instances and emphasize the gap between the combinatorial approach by Hurkens (2009) and the other solution approaches.

2.7 Concluding remarks

In this chapter, we proposed a solution methodology that uses a flexible matching model as a core engine for a special multi-skill workforce scheduling problem. The scheduling problem was defined by France Telecom in the ROADEF Challenge 2007. The opportunity of outsourcing some tasks is one aspect of our problem that distinguishes it from the similar ones defined in the literature. Technicians must work in teams for a workday and it is assumed that they can simultaneously use their skills in all domains while performing tasks.

The main contribution of this study is introducing flexibility in the matching model that was firstly proposed by Hurkens (2009). Moreover we propose several key measurements for tasks. The flexibility of our matching model resulted in better packing of the expert-requiring tasks especially in instances where skill distribution among technicians is heterogenous. Besides rare expertise, our results are remarkably superior in large instances.

Table 2.7: Results of problem instances A, B and X

Instance	FMM	(%)	Hurkens	(%)	Cordeau	(%)	EsGaNo	(%)	BEST*	LB
A1	2340	0.0	2340	0.0	2340	0.0	2340	0.0	2340	2310
A2	4755	0.0	4755	0.0	4755	0.0	4755	0.0	4755	2100
A3	11880	0.0	11880	0.0	11880	0.0	11880	0.0	11880	11340
A4	13452	0.0	13620	1.2	13452	0.0	14040	4.4	13452	10680
A5	29355	1.8	29355	1.8	29355	1.8	29400	1.9	28845	26940
A6	20055	6.7	20280	7.9	18795	0.0	18795	0.0	18795	17640
A7	30960	1.4	32520	6.5	30540	0.0	30540	0.0	30540	28672
A8	17355	2.6	18960	12.1	17700	4.6	20100	18.8	16920	16216
A9	28280	3.4	28320	3.6	27692	1.3	27440	0.3	27348	25558
A10	39300	2.6	40650	6.1	38636	0.9	38460	0.4	38296	36992
Average		*1.8*		*3.9*		*0.9*		*2.6*		
B1	34575	2.0	35460	4.6	37200	9.7	33900	0.0	33900	31875
B2	16755	5.6	18300	15.3	17070	7.6	16260	2.5	15870	14280
B3	16275	1.7	16965	6.0	18015	12.6	16005	0.0	16005	13965
B4	23925	0.6	27015	13.6	23775	0.0	24330	2.3	23775	16800
B5	88920	0.3	94200	6.2	117540	32.5	88680	0.0	88680	79530
B6	28785	5.1	30510	11.4	27390	0.0	27675	1.0	26955	24180
B7	31620	0.0	33060	4.6	33900	7.2	36900	16.7	31620	25290
B8	35520	10.4	32160	0.0	33240	3.4	36840	14.6	32160	31890
B9	28080	0.0	28080	0.0	29760	6.0	32700	16.5	28080	25680
B10	35040	1.0	35040	1.0	35640	1.7	41280	19.0	34680	32370
Average		*2.7*		*6.2*		*8.1*		*7.3*		
X1	146220	0.0	151980	3.9	159300	8.9	180240	23.3	146220	136680
X2	7740	6.6	9090	25.2	8280	14.0	8370	15.3	7260	5700
X3	48720	0.0	50400	3.4	50400	3.4	50760	4.2	48720	36060
X4	64600	0.0	65640	1.6	66780	3.4	68960	6.7	64600	58230
X5	144750	0.0	147000	1.6	157800	9.0	178560	23.4	144750	130995
X6	9690	2.2	10440	10.1	9900	4.4	10440	10.1	9480	6150
X7	32040	0.0	33120	3.4	47760	49.1	38400	19.9	32040	25410
X8	23220	0.0	23580	1.6	24060	3.6	23800	2.5	23220	17600
X9	122700	0.0	136020	10.9	152400	24.2	154920	26.3	122700	98805
X10	120300	0.0	131700	9.5	140520	16.8	152280	26.6	120300	87210
Average		*0.9*		*7.1*		*13.7*		*15.8*		
Overall		***1.8***		***5.8***		***7.6***		***8.6***		

* Results of the ROADEF Challenge 2007 are also considered.

Table 2.8: Running Times and Number of Constructed Schedules

	Data set A		Data set B		Data set X	
Instance	*Time(sec.)*	*#Schedules*	*Time(sec.)*	*#Schedules*	*Time(sec.)*	*#Schedule*
1	9	745	1086	125	1110	9
2	10	830	1090	105	1084	168
3	1085	9108	1087	28	1091	136
4	1079	7563	1106	35	1096	9
5	1081	2209	1089	130	1173	8
6	1081	2617	1086	54	1081	560
7	1082	838	1109	45	1086	164
8	1085	693	1142	3	1087	157
9	1083	793	1090	328	1173	14
10	1085	910	1081	325	1110	14

Flexible matching models. As a topic of further research, it will be useful to adapt our matching model to other multi-skill workforce scheduling problems with necessary modifications. This way, a global view in the approach may be obtained.

Lower bound model. The LBM is designed to guide the schedule construction phase by finding out which priority permutations are promising for shorter project completion times. This model is convenient to use in tactical-level decision making with some modifications. As its constraints are considered, the shadow prices provide us some information about which skills should be trained for less outsourcing costs. If a longer time horizon is taken into account, it is possible to make a skill training plan in order to prevent future outsourcing costs.

Sequencing decisions. In our combinatorial algorithm, we make sequencing decisions by using some priority rules. These decision may also be made by solving a MIP model, if the number of tasks in the workload of a team is not high.

Chapter 3

Stable multi-skill workforce assignments

This chapter analyzes stability in multi-skill workforce assignments of technicians and jobs. In our stability analysis, we extend the notion of blocking pairs as stated in the marriage model of Gale-Shapley. It is shown that finding stable assignments is NP-hard. In some special cases stable assignments can be constructed in polynomial time. For the general case, we give a set of linear inequalities of binary variables characterizing the set of stable assignments. Then we address how to find optimal stable assignments for different objectives. In our computational results, we report the solution times to find stable assignments. Open questions and further directions are discussed in the conclusion section.

3.1 Introduction

Stable assignment problem is a well-studied problem in the literature. It was first introduced for the marriage problem by David Gale in the beginning of 1960's (Gale and Shapley (1962)). In the marriage problem, there are two sets of players with equal sizes: *men* and *women*. Each player in these sets has a complete preference ordering, without ties, over the players in the opposite set. In a marriage, every man or woman finds a partner from the opposite set and a marriage is said to be *stable*, if there is no man-woman pair such that they are not partners but they prefer each other to their current partners. Such pairs are called *blocking pairs*. Gale and Shapley (1962) showed that stable marriages always exist and described an algorithm to find stable marriages.

In real life, several assignment problems are solved by using stable marriage algorithms. For example, assigning medical students to hospitals in several countries such as the U.S.A., Canada, Scotland, and Japan (Iwama et al. (2008)). Other examples are assigning students to the universities in Turkey, and assigning academicians to departments of the universities in France (Baïou and Balinski (2000b)).

In this study, we are interested in an assignment problem with players *technicians* and *jobs*. In Chapter 2, we present a MIP-based solution approach for a multi-skill

workforce scheduling problem. In the problem, technicians are grouped to perform a sequence of tasks and every team performing a job must have a collective capability above a certain threshold. Technicians in a team work together during a work day and every task in the sequence assigned to a team must be performed without interruption, without overlapping, and start and completion must be on the same day. Therefore, the whole technician-task schedule is formed by successive day schedules. Once we define a job as the sequence of tasks, a day schedule turns out to be an assignment of technicians and jobs. In this study, we are interested in the stability of the assignments in day schedules. Note that a schedule constructed by the combinatorial algorithm in Chapter 2 provides us feasible instances, or assignments, of our problem as many days as it lasts.

We assume that every player, a technician or a job, has a strict preference order over the opposite set. For technicians, this is easily understood. However, for jobs it is not obvious to see how a technician is preferred to another one, since jobs are processed by teams of technicians in our problem. Hence we introduce a new terminology and we say that *a job "likes" a technician, say t, if and only if (1) in the current team of the job there is a technician, say t', to whom t is preferred, and (2) the replacement of t' by t results in another feasible team for the job.* Note that a job likes a technician based on an assignment. A job likes a technician as long as the feasibility of its team is preserved after the less preferred technician is replaced by the more preferred technician. This feasibility condition makes our assignment problem hard to solve. Therefore, *the results of Gale and Sotomayor (1985) for stable assignments do not immediately apply to our case*, since the number of technicians performing a job may vary in different assignments.

Results of this chapter. In this chapter, stability in multi-skill workforce assignments is analyzed. To do this, the necessary terminology is introduced. Complexity of two special cases has been studied. Our results show that the general problem is NP-complete whereas some special cases can be solved in polynomial time. Moreover, the set of stable assignments is described with a set of linear inequalities of binary decision variables. Additionally, we present the formulations to find optimal stable assignments.

This chapter is organized as follows. In section 3.2 we give the basic definitions in the multi-skill assignments between technicians and jobs. Section 3.3 is a literature review of Gale-Shapley stability and multi-skill workforce scheduling. Section 3.4 includes our results on special cases of our assignment problem. In section 3.5, the set of stable assignments is described with a set of linear inequalities including binary variables. Our computational results are presented in Section 3.6. Conclusions and future research directions are discussed in Section 3.7.

3.2 Problem description and notation

3.2.1 Skills

The skills of technicians and the skill requirements of jobs are defined in the same way as explained in Section 2.2.1.

3.2.2 Technicians

We are given a set T of technicians and all technicians are assumed to have skills as explained in Section 2.2.2. In this chapter, we keep the assumption of *simultaneous skill use*.

Skills of Teams: Let $T' \subset T$ denote a team of technicians. Skills of team T' is defined in the same way as in Section 2.2.2 and they are found by $S_{T'} = \sum_{t \in T'} S_t$.

3.2.3 Jobs

In our problem, a set J of jobs is given and every job requires skills. The skill requirements of a job $j \in J$ are described by a matrix $RQ_j \in \mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$ which provides the information of the desired skill quantity (number of technicians) and skill quality (expertise). By definition, the requirements in RQ_j are *cumulative* in the sense that any requirement at a level is carried to lower ones in the same domain. Therefore, for a job j and a skill $\langle l, s\rangle$ we have $RQ_j^{\langle l',s\rangle} \geq RQ_j^{\langle l,s\rangle}$ for all $l' \leq l$.

In the description of the set of stable assignments, in Section 3.5, we need to use non-cumulative or *explicit* skill requirements. Let RQ_j be the (cumulative) skill requirements of job j. The explicit skill requirements, denoted by $RQ_j^* \in \mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$ are obtained as follows:

$$RQ_j^{*\langle l,s\rangle} = \begin{cases} RQ_j^{\langle l,s\rangle} & \text{if } l = |\mathbb{L}|, \\ RQ_j^{\langle l,s\rangle} - RQ_j^{\langle l+1,s\rangle} & \text{if } 0 < l < |\mathbb{L}|. \end{cases} \Leftrightarrow RQ_j^{\langle l,s\rangle} = \sum_{\ell=l}^{|\mathbb{L}|} RQ_j^{*\langle \ell,s\rangle} \tag{3.1}$$

For example, let $|\mathbb{S}| = 4$, $|\mathbb{L}| = 3$, $T(j)$ be a team assigned to job j, and let the requirement matrix for job j be as follows:

$$RQ_j = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \Longrightarrow RQ_j^* = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

In the above example, $RQ_j^{\langle 1,2\rangle} = 2$ tells us that there must be at least two technicians contributing to the team skill at level 1 in domain 2, hence $S_{T(j)}^{\langle 1,2\rangle} \geq 2$. One of them must be qualified at least at level 2, due to $RQ_j^{\langle 2,2\rangle} = 1$, so one needs at least level 1 implying $RQ_j^{*\langle 1,2\rangle} = 1$.

In our stability analysis, distinguishing certain skills will play crucial role. Hence we give the corresponding definition below. Note that, the relations defined below are assignment-independent.

Definition 3.2.1. *(Missing skills)*
The set $M(j,t) = \{\langle l, s\rangle \in \mathbb{L} \times \mathbb{S} | RQ_j^{*\langle l,s\rangle} > 0, S_t^{\langle l,s\rangle} = 0\}$ *is called the set of "missing skills" of technician* $t \in T$ *for job* $j \in J$.

Definition 3.2.2. *(Contributing skills)*
The set $N(j,t) = \{\langle l, s\rangle \in \mathbb{L}\times\mathbb{S} | RQ_j^{*\langle l,s\rangle} > 0\} \setminus M(j,t)$ *is called the set of "contributing skills" of technician* $t \in T$ *for job* $j \in J$.

3.2.4 Preferences

In real life, a technician may have several criteria for preferring a job to another such as skill match, job location, job provider and so on. In this study, we do not distinguish between these criteria, but we simply assume that every technician in the technician group T has a complete and strict preference ordering over the jobs. Let $t \in T$ be a technician and $j, j' \in J$ be two jobs in the problem. In our notation $j' <_t j$ denotes that technician t prefers job j to job j'. The complete preference ordering of t over the jobs is denoted by P_t.

From the job provider's point of view, there may be several criteria to distinguish technicians from each other like experience, age, capability of working in teams and so on. We assume that every job in J has a complete and strict preference ordering over the technicians. Let $j \in J$ be a job and $t, t' \in T$ be two technicians in the problem. In our notation $t' <_j t$ denotes that job j prefers technician t to technician t'. The complete preference ordering of j over the technicians is denoted by P_j.

Definition 3.2.3. *(Rank function)*
Let $r : (T \times J) \cup (J \times T) \mapsto Z_+$ denote the rank function. For a given pair of players, r returns the place of the "second" player in the preference list of the "first" player.

Note that the preference $j' <_t j$ can alternatively be denoted by $r(t, j') > r(t, j)$ using the rank function. In order to keep the common notation with Gale-Shapley literature, we will use the former for interpreting preferences unless the numerical values of preferences are needed.

3.2.5 Assignments

As we will show in Section 3.4, feasibility makes our problem hard to solve in times bounded by a polynomial in the size of the input. Moreover, the varying team sizes between different solutions makes it impossible to attain stability by directly applying the algorithms that are proposed in the literature.

Throughout our analysis, μ denotes a feasible technician-job assignment, $J_\mu(t)$ denotes the job to which the technician $t \in T$ is assigned under μ, and if technician t is not assigned to any job, then $J_\mu(t) = null$.

Feasibility: An assignment is a many-to-one matching between technicians and jobs. Every job in set J must be processed by a team of technicians possessing at least the required skills. In a feasible assignment, teams are formed for every job, hence every team performs exactly one job. Let μ be a technician-job assignment and let $T_\mu(j)$ denote the team of technicians assigned to job j under μ. The feasibility condition concerning skill requirements is formally described as

$$T_\mu(j) \text{ is a } \textit{feasible} \text{ team for } j \in J \Leftrightarrow RQ_j^{(l,s)} \leq S_{T_\mu(j)}^{\langle l,s \rangle}, \forall \langle l, s \rangle \in \mathbb{L} \times \mathbb{S} \tag{3.2}$$

Clearly, in a feasible assignment, every technician can be assigned to at most one job:

$$| \{j \in J | t \in T_\mu(j)\} | \leq 1, \quad \forall t \in T \tag{3.3}$$

The skill requirements of jobs that are tightly met by the assigned team of technicians under an assignment play an important role in deciding which technicians in a team can be replaced by alternative technicians outside the team. So let us first define

Definition 3.2.4. *(Critical skills)*
Under an assignment, the skills that are tightly satisfied by eligible technicians in the team of a job are called critical skills of that job.

The set of critical skills of job j under assignment μ is denoted by $C(\mu, j)$, and it is given by

$$C(\mu, j) = \{\langle l, s\rangle \in \mathbb{L} \times \mathbb{S} | S^{\langle l,s\rangle}_{T_\mu(j)} = RQ^{\langle l,s\rangle}_j > 0\} \tag{3.4}$$

Definition 3.2.5. *(Idle technician)*
Under an assignment, a technician is called "idle" in the team of a job if and only if the job prefers him least among those not contributing to any of its critical skills.

For an idle technician t^* in the team of job j under μ, we have

$$r(t^*, j) = \max_{t \in T_\mu(j)} \{r(t, j) | C(\mu, j) \cap N(t, j) = \emptyset\} \tag{3.5}$$

where $C(\mu, j)$ and $N(t, j)$ denote the set of critical skills of job j, and the set of contributing skills of technician t for job j respectively. Note that a team stays feasible after removing an idle technician.

3.2.6 Stability

In assignment μ, let t be not assigned to job j. We say that technician t likes job j , if and only if t prefers j to his current job. Technicians prefer being assigned to being unassigned, hence an unassigned technician likes every job in J. Moreover, job j likes technician t , if and only if in its team there is a technician t' to whom j prefers t and replacing t with t' results in another feasible team for job j. Consequently, t likes j on the condition that

$$J_\mu(t) \neq j \text{ and } J_\mu(t) <_t j \tag{3.6}$$

Job j likes technician t if and only if

$$\exists t \notin T_\mu(j) \text{ and } \exists t' \in T_\mu(j) : (T_\mu(j) \setminus \{t'\}) \cup \{t\} \text{ is feasible} \tag{3.7}$$

Note that we use the word *like* instead of *prefer*. A player prefers one player to another according to the given preference ordering. However, in our terminology, players like each other under a given assignment.

Definition 3.2.6. *(Stability)*
A feasible assignment is stable if and only if it does not contain a technician-job pair such that the technician is not assigned to the job and they "like" each other.

Such pairs in the above definition are called blocking pairs. For a blocking pair $\langle t, j\rangle$ under μ, we have
1- Technician t prefers job j to the job he is currently assigned to,
2- There exists t' in the team of j such that
(a) job j prefers t to t',
(b) j can be performed if t and t' are replaced.

Note that the cases (1) and (2-a) depend only on the preferences and the case (2-b) depends on the skills of team performing job j under μ. Now let us examine the case (2-b). The following remark states the condition of the replacement of a technician by another one.

Remark 3.2.1. *Under a feasible assignment, let t' be in the team of job j but not t. Then t can replace t' if and only if t is qualified in those critical skills of j to which t' contributes.*

According to the remark above, t can replace t' if and only if

$$N(j,t') \cap C(\mu, j) \subseteq N(j,t) \tag{3.8}$$

Stability condition. We say that the assignment μ is stable *if and only if* there is no technician pair $\langle t, t'\rangle$ such that $t' \in T_\mu(j)$, $t \notin T_\mu(j)$, and the following relations hold

$$t' <_j t, \quad J_\mu(t) <_t j, \quad N(j,t') \cap C(\mu, j) \subseteq N(j,t) \tag{3.9}$$

Above we expressed the condition that leads to instability. The expression includes player preferences, contributing skills, and critical skills. In Section 3.5, the above stability condition is formulated as linear inequalities in which binary decision variables are used to detect critical skills in an assignment. Next, we describe how to eliminate idle technicians from an assignment which will be needed in our complexity analysis.

Elimination of idle technicians. Let technician t be idle in the team of job j under μ. Note that $T_\mu(j) \setminus \{t\}$ is feasible for job j, since

- $RQ_j^{\langle l,s\rangle} = S^{\langle l,s\rangle}_{T_\mu(j)} = S^{\langle l,s\rangle}_{T_\mu(j)} - S^{\langle l,s\rangle}_{\{t\}} = S^{\langle l,s\rangle}_{T_\mu(j)\setminus\{t\}}, \quad \forall \langle l,s\rangle \in C(t,j).$
- $RQ_j^{\langle l,s\rangle} \leq S^{\langle l,s\rangle}_{T_\mu(j)} - 1 \leq S^{\langle l,s\rangle}_{T_\mu(j)} - S^{\langle l,s\rangle}_{\{t\}} = S^{\langle l,s\rangle}_{T_\mu(j)\setminus\{t\}}, \forall \langle l,s\rangle \in \mathbb{L}\times\mathbb{S} \setminus C(\mu,j).$

The first of the above relations is due to (3.5), and the second one is true by definition of skill matrices. Now let $r(t,j') = r(t,j) + 1$. Then we add technician t to the team of job j', resulting in assignment μ'. We observe the following case

$$RQ_{j'}^{\langle l,s\rangle} + 1 \leq S^{\langle l,s\rangle}_{T_\mu(j')} + S^{\langle l,s\rangle}_{t} = S^{\langle l,s\rangle}_{T_{\mu'}(j')}, \quad \forall \langle l,s\rangle \in N(t,j'). \tag{3.10}$$

Note that (3.10) implies $N(t, j') \cap C(\mu', j') = \emptyset$ and adding one technician to the team $T_\mu(j')$ results in $C(\mu', j') \subseteq C(\mu, j')$. If there exists a technician t° in $T_\mu(j')$ such that $t^\circ <_{j'} t$ and $C(\mu', j') \cap N(t^\circ, j') = \emptyset$, technician t is not idle under μ', but technician t°. So in any case job j' has an idle technician under μ'. Next, we let that idle technician to join to the team of the job with one lower ranking than j' in his preference list. If j' is least preferred by that idle technician, then we leave that idle technician unassigned. This process progressively continues until teams have no idle technicians. Throughout the elimination no blocking pair is created and our procedure is monotonic in $\sum_{t \in T} r(t, J_\mu(t))$, hence will stop in polynomial time.

3.3 Literature review

Although Gale-Shapley stability and multi-skill workforce scheduling are well-known problems in the literature, Gale-Shapley stability in multi-skill workforce schedules has not been studied to the best of our knowledge.

3.3.1 Gale-Shapley stability

Having introduced the concept of stability in marriages, Gale and Shapley (1962) stated the following theorem

Theorem 3.3.1. *(Gale and Shapley (1962)) There always exists a set of stable marriages.*

The authors proposed a polynomial time algorithm so-called *proposal-disposal* algorithm to construct stable marriages. In fact, the proposal-disposal algorithm constructs *man-optimal* and *woman-optimal* stable marriages where a *man-optimal* stable marriage is the one in which every man is as happy as in any stable marriage. Woman-optimal stable marriage is defined similarly. Vande Vate (1989) showed that the relaxation of the integer programming (IP) model of the marriage problem has integral solutions. Hence a stable marriage can also be constructed by solving the corresponding LP model.

Gale and Sotomayor (1985) study a generalization of the marriage problem called "university admissions problem" in which the players are universities and applicants, and players may order a subset (not necessarily all) of the players in the opposite set in their preference lists. Such preference lists are said to be *incomplete*. Every university has a quota which is an upper bound for the number of applicants to be admitted. One of the authors' results is given in the following theorem.

Theorem 3.3.2. *(Gale and Sotomayor (1985)) If an applicant is admitted in a stable admission, then she/he is admitted in all stable assignments. Moreover, every university admits the same number of applicants in all stable admissions.*

Gale and Sotomayor (1985) show how to interpret a university admissions problem as a marriage problem. Moreover, Baïou and Balinski (2000a) give a separation algorithm to find stable admissions that uses a graph-theoretic approach to stability and runs in polynomial time. In fact, every admission is called a many-to-one

matching in graph theory and the authors showed that the results for one-to-one matchings (marriages) and many-to-one matchings (admissions) have equivalents in general many-to-many matchings (Baïou and Balinski (2000b)).

The version of the marriage problem in which incomplete preferences include ties is NP-hard (Iwama et al. (1999)), so is the university admissions problem. Several approximation algorithms have been proposed for this version of the marriage problem (Iwama et al. (2004), Iwama et al. (2008)). Recently, Gelain et al. (2010) propose a local search approach to the marriage problem with incomplete lists and ties. Moreover, Fleiner et al. (2007) study a generalization of classical stable marriage problem with partially ordered preference lists and forbidden pairs and show that the problem is NP-hard.

Baïou and Balinski (2000b) considered many-to-many assignments in a general manner where an applicant is not assigned to institutions. Instead he/she has a certain amount of time to work and an allocation of that time among institutions is found. So the time amount of applicants is allocated to institutions in real numbers. This generalization is indeed important for practical cases such as finding a schedule for employees who can flexibly work in several companies.

3.3.2 Multi-skill workforce scheduling

In this study, Gale-Shapley stability is analyzed in the multi-skill workforce scheduling context that is studied in Chapter 2. The multi-skill workforce schedules in Chapter 2 are composed of day schedules. In a day schedule, every team is assigned to a task sequence. Tasks in a sequence are processed without interruption, without overlapping within the same day. Every day schedule is an instance of our assignment problem. Jobs correspond to the task sequences and technicians correspond to the members of teams. Moreover, we assume that every job is day-long so that one team can perform only one job. For a detailed literature review of multi-skill workforce scheduling, we refer to Section 2.4.

3.4 Complexity analysis

3.4.1 Stable technician-job assignment problem

In this section we give a formal definition of our assignment problem and we denote it by $STJAP(a, b, c)$ where a, b, and c stand for $|\mathbb{L}|$, $|\mathbb{S}|$, and the upper bound on the size of teams in assignments respectively. For example, $STJAP(n, 1, n)$ denotes the special case of instances in which there is one skill domain, no restriction in the number of skill levels and no restriction on the size of teams performing jobs. Moreover, if the equality sign "=" is used in front of any term, that term is fixed to the corresponding parameter. For example, $STJAP(1, n, =2)$ denotes the special case in which assignments have teams of size two. In the following section, we analyze the complexity of solving the special cases $STJAP(n, 1, n)$ and $STJAP(1, n, =2)$. Now, we give a formal definition of the general problem.

STABLE TECHNICIAN-JOB ASSIGNMENT PROBLEM $STJAP(n, n, n)$

INSTANCE: The sets $\mathbb{S}$, $\mathbb{L}$, T, J denoting skill domains, skill levels, technicians, and jobs respectively. Skills $S_t \in \{0,1\}^{\mathbb{L}\times\mathbb{S}}$ for every t in T and skill requirements $RQ_j \in \mathbb{Z}^{\mathbb{L}\times\mathbb{S}}$ for every j in J.

Preferences P_t for every t in T and P_j for every j in J.

QUESTION: Does there exist a feasible assignment (satisfying (3.2) and (3.3)) that is *stable* according to the Definition 3.2.6?

3.4.2 Special case: $STJAP(n, 1, n)$

The special case $STJAP(n, 1, n)$ includes problem instances in which there is one skill domain, any number of skill levels, and teams may have any number of technicians. We drop the parameter "s" in the notation of skills and skill requirements due to unique skill domain. In an assignment of $STJAP(n, 1, n)$, technicians of a team, *without idle technician*, can be allocated using the "technician allocation rule" that is given in the proof of Theorem 3.4.3. Moreover, it is not difficult to see that this allocation is a partition of technicians, each subset of which is assigned to the part of the workload requiring skills at a specific level. So we distinguish parts of a job in the following definition.

Definition 3.4.1. *(Sub-job)*
In $STJAP(n, 1, n)$, the part of skill requirement of job j at level $l \in \mathbb{L}$ is called the sub-job*of job j at level l and denoted by $SJ(j, l)$.*

Note that the number of eligible technicians required by sub-job $SJ(j, l)$ is equal to $RQ_j^{*(l)}$ which is the explicit skill requirement at level l. Let μ be a feasible assignment of special case $STJAP(n, 1, n)$; satisfying (3.2) and (3.3). Clearly, $RQ_j^{(1)} = \sum_{l\in L} RQ_j^{*(l)}$ (due to (3.1)), and $S^{(1)}_{T_\mu(j)} = |T_\mu(j)|$. Then rewriting (3.2) gives

$$RQ_j^{(1)} \leq S^{(1)}_{T_\mu(j)} \Rightarrow \sum_{l\in\mathbb{L}} RQ_j^{*(l)} \leq |T_\mu(j)|, \quad \forall j \in J. \tag{3.11}$$

If team $T_\mu(j)$ satisfies (3.11) strictly, then $T_\mu(j)$ has an idle technician. Therefore, the assignments without idle technicians satisfy (3.11) with equality which implies that every sub-job is assigned to a number of technicians that is exactly equal to the corresponding (explicit) skill requirement. Moreover idle technicians from an assignment can be taken out by "idle technician elimination" procedure, explained in Section 3.2.6, without forming a blocking pair.

As discussed above, efficient assignments of $STJAP(n, 1, n)$ include sub-jobs assigned to a number of technicians exactly as many as required. From this point of view, such assignments resemble the university-applicant admissions in which all universities admit a number of applicants as many as their quotas. This analogy leads us to a complexity relation such that $STJAP(n, 1, n)$ reduces to a version of the university admissions problem. Next we give the formal definition of this problem.

University Admissions Problem with Filled Quotas (*UAFQ*)

Instance: The sets of players, U and A denoting the *universities* and the *applicants* respectively. A quota $q_u \in \mathbb{Z}_+$ for every u in U that is an upper bound on the number of applicants who may be admitted. Complete and strict preferences P_u for every u in U over applicants. Incomplete and strict preferences P_a for every a in A over universities.

Question: Does there exist a stable admission ν with *filled quotas*? In this context, ν is called stable if it satisfies (3.12) and it has filled quotas if it satisfies (3.13).

$$\{\langle a,u\rangle \in A\times U | \nu(a)\neq u, \nu(a) <_a u, \text{ and } \exists a' \in A : \nu(a')=u, a' <_u a\} = \emptyset \tag{3.12}$$

$$|\{a \in A | \nu(a) = u\}| = q_u, \quad \forall u \in U \tag{3.13}$$

Now we give one of the results obtained by Gale and Sotomayor (1985). We show that *UAFQ* is in P by using the authors' results in the following theorem and in the Theorem 3.3.2.

Theorem 3.4.1. *(Gale and Sotomayor (1985)) University admissions problem is solvable in polynomial time.*

Corollary 3.4.2. *UAFQ* $\propto$ *University admissions problem.*

Proof. By Theorem 3.4.1, a stable admission can be constructed in polynomial time and by Theorem 3.3.2 constructing one stable admission is enough to see if all stable admissions of the given instance are with filled quotas or not. □

Theorem 3.4.3. $STJAP(n,1,n) \propto UAFQ$.

Proof. Firstly, we will give a polynomial time transformation from $STJAP(n,1,n)$ to *UAFQ* as follows

- Create an applicant $a(t)$ for every element $t \in T$.
- Create a university $u(SJ(j,l))$ with $q_{u(SJ(j,l))} = RQ_j^{*(l)}$ for every sub-job $SJ(j,l)$.
- Preferences:

Compatibility and feasibility: In the context of university admissions, an applicant-university pair is said to be *compatible* if and only if both have each other in their preference lists. In our transformation, the pair $(a(t), u(SJ(j,l)))$ is compatible if and only if $S_t^{(l)} = 1$. The following preferences are defined for compatible applicant-university pairs.

$$u(SJ(j,l)) <_{a(t)} u(SJ(j',l')) \quad \text{if } j <_t j', \quad \forall j,j' \in J, l,l' \in \mathbb{L},. \tag{3.14}$$

$$u(SJ(j,l)) <_{a(t)} u(SJ(j,l')), \quad \forall l' \in \mathbb{L} : l < l'. \tag{3.15}$$

$$a(t') <_{u(SJ(j,l))} a(t) \quad \text{if } t' <_j t, \quad \forall j \in J. \tag{3.16}$$

Now, we show that a YES instance of the *UAFQ* corresponds to a YES instance of the $STJAP(n,1,n)$. Let ν be a stable admission with filled quotas of a YES instance of the *UAFQ* . We will show that the corresponding $STJAP(n,1,n)$ instance is a YES instance.

Next, we show how to obtain a technician-job assignment μ from the admission ν. Let $T_\mu(j)^{(l)}$ denote the applicants assigned to university $u(SJ(j,l))$ under ν which is given by $T_\mu(j)^{(l)} = \{t \in T | \nu(a(t)) = u(SJ(j,l))\}$. Then the team of a job j is determined by

$$T_\mu(j) = \bigcup_{l \in \mathbb{L}} T_\mu(j)^{(l)} \tag{3.17}$$

where we have $|T_\mu(j)^{(l)}| = q_{u(SJ(j,l))} = RQ_j^{*(l)}$ due to filled quotas in ν. Then, we show that teams under μ meet skill requirements at for every l in $\mathbb{L}$ as follows

$$S^{(l)}_{T_\mu(j)} = \sum_{t \in T_\mu(j)} S_t^{(l)} \geq \sum_{l' \geq l} \sum_{t \in T_\mu(j)^{(l')}} S_t^{(l)} = \sum_{l' \geq l} |T_\mu(j)^{(l')}| = \sum_{l' \geq l} RQ_j^{*(l')} = RQ_j^{(l)} \tag{3.18}$$

Now, in order to show the stability of μ we suppose, to the contrary, that μ is blocked by pair $\langle t^\circ, j^\circ \rangle$. So there must be a technician $t' \in T_\mu(j^\circ)$ such that $\nu(a(t')) = u(SJ(j^\circ, l'))$, and the following relations hold.

$$J_\mu(t^\circ) <_{t^\circ} j^\circ, t' <_{j^\circ} t^\circ, \text{ and } RQ_{j^\circ} \leq S_{(T_\mu(j^\circ) \setminus \{t'\}) \cup \{t^\circ\}}, \tag{3.19}$$

Note that there exists a level $l^\circ \leq l'$ such that $S_{t^\circ}^{(l^\circ)} = 1$ and $RQ_{j^\circ}^{*(l^\circ)} > 0$. To see this, let us suppose that $S_{t^\circ}^{(\ell)} = 0$, $\forall \ell \in \mathbb{L}$ which results in $RQ_{j^\circ}^{(\ell)} \leq S^{(\ell)}_{T_\mu(j^\circ)} - S_{t'}^{(\ell)}$ due to (3.19), thus making $T_\mu(j^\circ) \setminus \{t'\}$ feasible for job j°. Then rewriting (3.11) gives us a contradiction as below.

$$RQ_{j^\circ}^{(1)} \leq |T_\mu(j^\circ) \setminus \{t'\}| = \sum_{l \in \mathbb{L}} |T_\mu(j^\circ)^{(l)}| - 1 = \sum_{l \in \mathbb{L}} |RQ_{j^\circ}^{*(l)}| - 1 = RQ_{j^\circ}^{(1)} - 1 \tag{3.20}$$

Next, we state a claim which will help in detecting a blocking pair in ν, hence to a contradiction.

Claim: For any $l \leq l'$ with $S_{t^\circ}^{(l)} = 0$, we have the following relation.

$$\text{If } \exists t \in T_\mu(j^\circ)^{(l)}, \text{ then } \exists (t^*, l^*) \text{ s.t. } l^* < l, t^* \in T_\mu(j^\circ)^{(l^*)}, S_{t^*}^{(l)} = 1, \text{ and } t^* <_{j^\circ} t. \tag{3.21}$$

To prove the claim, let us plug the values $S^{(l)}_{t'} = 1$ and $S^{(l)}_{t^\circ} = 0$ into the rightmost inequality in (3.19). We get $RQ^{(l)}_{j^\circ} + 1 \leq S^{(l)}_{T_\mu(j^\circ)}$ where $S^{(l)}_{T_\mu(j^\circ)}$ is

$$S^{(l)}_{T_\mu(j^\circ)} = \sum_{\ell \geq l} S^{(l)}_{T_\mu(j^\circ)^{(\ell)}} + \sum_{\ell < l} S^{(l)}_{T_\mu(j^\circ)^{(\ell)}} = \sum_{\ell \geq l} RQ^{*(\ell)}_{j^\circ} + \sum_{\ell < l} S^{(l)}_{T_\mu(j^\circ)^{(\ell)}} = RQ^{(l)}_{j^\circ} + \sum_{\ell < l} S^{(l)}_{T_\mu(j^\circ)^{(\ell)}} \tag{3.22}$$

So $RQ^{(l)}_{j^\circ} + 1 \leq RQ^{(l)}_{j^\circ} + \sum_{\ell<l} S^{(l)}_{T_\mu(j^\circ)^{(\ell)}}$ becomes $1 \leq \sum_{\ell<l} S^{(l)}_{T_\mu(j^\circ)^{(\ell)}}$ which implies

$$\exists \langle t^*, l^* \rangle : l^* < l, t^* \in T_\mu(j^\circ)^{(l^*)}, \text{ and } S^{(l)}_{t^*} = 1. \tag{3.23}$$

To show that $t^* <_{j^\circ} t$, we suppose $t <_{j^\circ} t^*$ is true. Then $\langle u(SJ(j^\circ, l)), a(t^*) \langle$ would block ν due to fact that both $a(t^*)$ and $a(t)$ prefer $u(SJ(j^\circ, l))$ to $u(SJ(j^\circ, l^*))$. This completes our proof of the Claim.

Remembering that there exists a level $l^\circ \leq l'$ with $S^{(l^\circ)}_{t^\circ} = 1$ and $RQ^{*(l^\circ)}_{j^\circ} > 0$, and applying the above Claim repeatedly, we find a set of pairs $\{\langle t_0, l_0 \rangle, \langle t_1, l_1 \rangle, \ldots, \langle t_k, l_k \rangle\}$ such that $\langle t_0, l_0 \rangle = \langle t', l' \rangle$, that $l_k \leq l^\circ$, and that

$$S^{(l_{i-1})}_{t_i} = 1, \quad t_i <_{j^\circ} t_{i-1}, \quad i = 1, \ldots, k. \tag{3.24}$$

We see that the pair $\langle u(SJ(j^\circ, l_k)), a(t^\circ) \rangle$ blocks ν due to the preferences in (3.19) and (3.24). This contradicts the stability ν, hence, the stable assignment μ implies we have a YES $STJAP(n, 1, n)$ instance.

Next, we show that a YES $STJAP(n, 1, n)$ instance corresponds to a YES $UAFQ$ instance. Let μ be a "stable" assignment of the YES $STJAP(n, 1, n)$ instance. An admission ν from μ can be obtained in two steps: eliminating idle technicians, and allocating (or assigning) technicians to sub-jobs. Elimination of idle technicians can be done in polynomial time as mentioned in Section 3.2.6.

Technician allocation rule: Pick the sub-job that requires the highest skill-level among the ones to which no technician is allocated yet. Choose the technicians who are most preferred by the job among the eligible ones.

Note that technician allocation will always end up with an admission in which all quotas are filled. Plugging the equation $RQ^{(l)}_j = RQ^{*(l)}_j + \sum_{l<\ell} RQ^{*(\ell)}_j$ into the feasibility of skill requirements condition in (3.2) gives us

$$RQ^{*(l)}_j \leq S^{(l)}_{T_\mu(j)} - \sum_{l<\ell} RQ^{*(\ell)}_j, \quad \forall j \in J, 0 < l \leq |\mathbb{L}| \tag{3.25}$$

As seen above, for every sub-job there is enough number of technicians left after some technicians are allocated to the sub-jobs requiring higher skill levels. Therefore, the obtained admission ν has filled quotas. Next, to show that ν is stable, suppose that, to the contrary, $\langle u(SJ(j^\circ, l^\circ)), a(t^\circ) \rangle$ is a blocking pair. Firstly, we assume that $J_\mu(t^\circ) \neq null$ and let $\nu(a(t^\circ)) = u(SJ(J_\mu(t^\circ), l^*))$. So there must be an applicant $a(t')$ such that $\nu\big(a(t')\big) = u\big(SJ(j^\circ, l^\circ)\big)$ and we have the following preference.

$$a(t') <_{u(SJ(j^\circ, l^\circ))} a(t^\circ) \Rightarrow t' <_{j^\circ} t^\circ \tag{3.26}$$

Let us consider the case $J_\mu(t^\circ) = j^\circ$. Note that $l^* < l^\circ$, since $a(t^\circ)$ prefers $u\big(SJ(j^\circ, l^\circ)\big)$ to $u\big(SJ(j^\circ, l^*)\big)$. Technician t° with $S_{t^\circ}^{(l^\circ)} = 1$ is not assigned to $SJ(j^\circ, l^\circ)$, but to $SJ(j^\circ, l^*)$, although $t' <_{j^\circ} t^\circ$ which contradicts the technician allocation rule.

Next, consider the case $J_\mu(t^\circ) \neq j^\circ$. Then we have the following cases.

$$J_\mu(t^\circ) <_{t^\circ} j^\circ,\ t' <_{j^\circ} t^\circ, \text{ and } T_\mu(j^\circ) \setminus \{t'\} \cup \{t^\circ\} \text{ is feasible for } j^\circ. \tag{3.27}$$

Above, we see that $\langle t^\circ, j^\circ \rangle$ blocks μ which is a contradiction. Consequently, we showed that a YES $STJAP(n, 1, n)$ instance corresponds to a YES $UAFQ$ instance. □

3.4.3 Special case: $STJAP(1, n, =2)$

In the instances of the special case $STJAP(1, n, =2)$ every job is processed by two technicians, and technicians are either skilled or unskilled in a specialization field. We drop the parameter "l" in the notation of skills and skill requirements due to one skill level. There may be any number of specialization fields though. We note that the proof of Theorem 3.4.4 concerns the feasibility of skill requirements in $STJAP(1, n, =2)$.

Theorem 3.4.4. *$STJAP(1, n, =2)$ is NP-complete.*

Proof. Note that in polynomial time the feasibility and the stability of a given assignment can be checked. Now, we give a reduction from the "three-dimensional matching problem" (3-DM). Below we define the following variant of the NP-complete 3-DM problem (See Garey and Johnson (1979)).

> Problem: Three-Dimensional Matching (3-DM)
>
> Instance: An integer n.
>
> Three pairwise disjoint sets $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$.
>
> Set of triples $X \subseteq A \times B \times C$.
>
> Question: Does there exist a subset $X' \subseteq X$ of n triples, such that every element in $A \cup B \cup C$ occurs in exactly one triple in X'?

Let us consider an arbitrary instance of the 3-DM problem and translate it into a corresponding instance of our scheduling problem.

- For every element $a_i \in A$, create a corresponding job $j(a_i)$.
- For every element $b_j \in B$, create a corresponding technician $t(b_j)$.
- For every element $c_k \in C$, create a corresponding technician $t(c_k)$.
- Skill domains $\mathbb{D} = \{d(i,k) | i = 1, \ldots, n, k = 1, \ldots, n\} \cup \{\beta, \gamma\}$, skill levels $|\mathbb{L}| = 1$.

Skills and skill requirements are defined as follows:

$$RQ^{(d)}_{j(a_i)} = \begin{cases} 1 & \text{for } d \in \{d(i,1), \ldots, d(i,n), \beta, \gamma\} \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

$$S^{(d)}_{t(b_j)} = \begin{cases} 1 & \text{for } d \in \{d(i,k) | (a_i, b_j, c_k) \in X\} \cup \{\beta\} \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

$$S^{(d)}_{t(c_k)} = \begin{cases} 0 & \text{for } d \in \{d(1,k), \ldots, d(n,k), \beta\} \\ 1 & \text{otherwise} \end{cases} \tag{3.30}$$

We now show that a YES answer to our assignment problem corresponds with a YES answer to the 3-DM problem and vice-versa. Assume that we are given a feasible assignment μ with teams $|T_\mu(j(a_i))| = 2$, for all i. Considering the requirements in (3.28), we have:

$$1 = RQ^{(\beta)}_{j(a_i)} \leq S^{(\beta)}_{T_\mu(j(a_i))} \Rightarrow 1 \leq |T_\mu\big(j(a_i)\big) \cap \{t(b_1), \ldots, t(b_n)\}|, \quad \forall i \tag{3.31}$$

$$1 = RQ^{(\gamma)}_{j(a_i)} \leq S^{(\gamma)}_{T_\mu(j(a_i))} \Rightarrow 1 \leq |T_\mu\big(j(a_i)\big) \cap \{t(c_1), \ldots, t(c_n)\}|, \quad \forall i \tag{3.32}$$

Therefore we see that the team assigned to job $j(a_i)$ has one technician from set B, by (3.31), and one technician from set C, by (3.32), hence $T_\mu\big(j(a_i)\big) = \{t(b_j), t(c_k)\}$. Moreover, by (3.30) we see that $S^{(d(i,k))}_{t(c_k)} = 0$, but team feasibility enforces $S^{(d(i,k)))}_{T_\mu(j(a_i))} = 1$, by (3.28), implying that $S^{(d(i,k))}_{t(b_j)} = 1$. Therefore, by (3.29), $(a_i, b_j, c_k) \in X$. Feasibility of the assignment μ implies that every technician is in exactly one team and every job is performed, hence this coincides with a YES answer in the 3-DM problem. Then the desired subset is found by

$$X' = \{(a_i, b_j, c_k) \in X | \quad T_\mu\big(j(a_i)\big) = \{t(b_j), t(c_k)\}, \text{ and } \forall a_i \in A\}. \tag{3.33}$$

Let us consider the other direction. If a subset of the set X is given with the desired property in the 3-DM problem, then the triples in the subset can be directly translated to teams and a feasible assignment is obtained. This simple argument shows that a YES answer to the 3-DM problem leads to a YES answer to the assignment problem. □

Corollary 3.4.5. *$STJAP(n, n, n)$ is NP-complete.*

3.5 Stable assignments

In this section we present linear inequalities of binary variables and we will show that the solution set of these linear inequalities corresponds to the set of the stable assignments. In the following sections, we will address how to find the optimal stable assignments if the importance of assignments can be expressed with given weights of technicians and/or jobs.

Table 3.1: Indices,sets, parameter and variables in the linear inequalities

Indices		
	t, t'	Technician index, $t, t' \in T$
	j, j'	Job index, $j, j' \in J$
	l	Skill level index, $l \in \mathbb{L}$
	s	Skill domain index, $s \in \mathbb{S}$
Sets		
	$M(j,t)$	Missing skills of technician t for job j, $M(j,t) \subset \mathbb{L} \times \mathbb{S}$
Parameter		
	$\delta_{j,\langle t,t'\rangle}$	Equal to 1 if j does not prefer t to t', 0 otherwise
Variables		
	x_{tj}	Equal to 1, if t is assigned to j; otherwise 0
	y_{tj}	Equal to 1, if t likes j; otherwise 0
	$\beta_{j,\langle l,s\rangle}$	Equal to 1, if $\langle l,s\rangle$ is not *critical* for j; otherwise 0
	$\tau_{j,\langle t,t'\rangle}$	Equal to 1, if t cannot replace t' in team of j; otherwise 0

$$\sum_{j \in J} x_{tj} \leq 1, \qquad \forall t \in T \tag{3.34}$$

$$\sum_{t \in T} S_t x_{tj} \geq RQ_j, \qquad \forall j \in J \tag{3.35}$$

$$\sum_{j' : j' <_t j} x_{tj'} \leq y_{tj}, \qquad \forall t \in T, \forall j \in J \tag{3.36}$$

$$\sum_{j' : j <_t j'} x_{tj'} + y_{tj} \leq 1, \qquad \forall t \in T, \forall j \in J \tag{3.37}$$

$$\beta_{j,\langle l,s\rangle} \leq \sum_{t \in T} S_t^{\langle l,s\rangle} x_{tj} - RQ_j^{\langle l,s\rangle}, \qquad \forall j \in J, \forall \langle l,s\rangle \in \mathbb{L} \times \mathbb{S}, \tag{3.38}$$

$$\sum_{t \in T} S_t^{\langle l,s\rangle} x_{tj} - RQ_j^{\langle l,s\rangle} \leq |T| \beta_{j,\langle l,s\rangle}, \qquad \forall j \in J, \forall \langle l,s\rangle \in \mathbb{L} \times \mathbb{S}, \tag{3.39}$$

$$\tau_{j,\langle t,t'\rangle} \leq \sum_{\langle l,s\rangle \in M(j,t)} S_{t'}^{\langle l,s\rangle} (1 - \beta_{j,\langle l,s\rangle}), \qquad \forall j \in J, \forall \langle t,t'\rangle \in T \times T \tag{3.40}$$

$$\sum_{\langle l,s\rangle \in M(j,t)} S_{t'}^{\langle l,s\rangle} (1 - \beta_{j,\langle l,s\rangle}) \leq |\mathbb{L} \times \mathbb{S}| \tau_{j\langle t,t'\rangle}, \qquad \forall j \in J, \forall \langle t,t'\rangle \in T \times T \tag{3.41}$$

$$x_{t'j} \leq (1 - y_{tj}) + \tau_{j,\langle t,t'\rangle} + \delta_{j,\langle t,t'\rangle}, \qquad \forall j \in J, \forall \langle t,t'\rangle \in T \times T \tag{3.42}$$

$$x_{t'j}, y_{tj}, \tau_{j,\langle t,t'\rangle}, \beta_{j,\langle l,s\rangle} \in \{0,1\} \quad \forall j \in J, \forall \langle t,t'\rangle \in T \times T, \forall \langle l,s\rangle \in \mathbb{L} \times \mathbb{S} \tag{3.43}$$

3.5.1 The set of stable assignments

The main idea underlying the linear inequalities is to ensure that the *stability condition* is satisfied as given in Section 3.2.6. In this linear equalities, we guarantee that a technician t is assigned to a job j if and only if there is no technician t' such that t' is not assigned to job j, that job j prefers technician t' to technician t, and that technician t' can perform those critical skills to which technician t contributes. Therefore, forming a blocking pair is prevented. The notation used in the linear inequalities is given in Table 3.1.

Feasibility: Inequalities (3.34) and (3.35) ensure that feasibility conditions (3.2) and (3.3), explained in Section 3.2.5, are satisfied.

Recognition: By inequalities (3.36), a technician likes a job if he is assigned to another one that is less preferred by that technician. Inequalities (3.38) and (3.39) are used to figure out that a positive skill requirement is tightly satisfied in a team. In this case, the corresponding binary variable becomes zero. Similarly, inequalities (3.40) and (3.41) detect that a technician *cannot* replace another one in a team if the former one misses a skill that is critical in the team and the latter contributes at that skill.

Stability: The following lemma shows that all assignments constructed by solving the inequalities (3.34)-(3.43) are stable, i.e. no blocking pairs are formed by satisfying (3.42).

Lemma 3.5.1. *An assignment μ, with corresponding values of binary variables x, y, β, and τ, satisfying constraints (3.34)-(3.43) is* stable.

Proof. Suppose, to the contrary, that assignment μ is unstable, i.e. there is a technician-job pair $\langle t, j\rangle$ such that t is *not* in the team of j and they like each other. Since t likes j, we have $y_{tj} = 1$. Moreover, since j likes t, so there is a technician t' is in the team of j, so $x_{t'j} = 1$, such that $t' <_j t$, hence $\delta_{j,\langle t,t'\rangle} = 0$. To satisfy the inequality (3.42), we must have $\tau_{j,\langle t,t'\rangle} = 1$. The value $\tau_{j,\langle t,t'\rangle} = 1$ in the inequalities (3.40)-(3.41) tells us that there exists a missing skill of t in which t' is qualified. Then by inequalities (3.38)-(3.39), we see that this skill is critical. Finally, if t replaces t' in the team of j, the new team would not satisfy the requirement of the aforementioned critical skill, since t misses it. This contradicts the assumption that j likes t (or t can replace t'). □

Definition 3.5.1. *Let $\Pi = \{\mu | \mu$ satisfies $(3.34) - (3.43)\}$ and $\Sigma = \{\mu | \mu$ is stable $\}$.*

Note that Lemma 3.5.1 implies $\Pi \subseteq \Sigma$.

Lemma 3.5.2. *One has $\Sigma \subseteq \Pi$ implying that every stable assignment satisfies (3.34)-(3.43).*

Proof. Let μ be a stable assignment for which the values of x, y, β, nd τ are determined by checking teams, technicians skills, and skill requirements of jobs. By feasibility conditions in (3.2) and (3.3), inequalities (3.34) and (3.35) are satisfied and critical skills as in the Definition 3.2.4 satisfy inequalities (3.38).

Let us consider the inequality (3.42) whose satisfaction may not be seen immediately. Under μ, there are two possible cases for a technician-job pair (t', j). Firstly; for

the case $x_{t'j} = 0$, inequality (3.42) becomes redundant and the values $\tau_{j,(t,t')}$ for any pair (t, t') do not mean anything for us. Secondly; in the case $x_{t'j} = 1$, let us assume that, in the right side of inequality (3.42) $y_{tj} = 1$ and $\delta_{j,(t,t')} = 0$ for a technician t not in the team of j. The value $y_{tj} = 1$ implies that t prefers j to the job he is assigned under μ and the value $\delta_{j,(t,t')} = 0$ implies that j prefers t to t'. Let us consider the only critical case $\tau_{j,(t,t')} = 0$ that would lead to the violation of inequality (3.42). In this case, by inequalities (3.40)-(3.41) we see that all critical skills possessed by t' are also possessed by t which means that t can replace t' in team of j, hence (t, j) would be a blocking pair. This contradicts the stability of μ. □

Theorem 3.5.3. *The constraints (3.34)-(3.43) characterize the set of stable assignments.*

Proof. By Lemmas 3.5.1 and 3.5.2, we see $\Sigma = \Pi$. □

3.5.2 Optimality in stable workforce assignments

An optimal stable workforce assignment can be constructed by solving an integer programming (IP) model which involves the inequalities (3.34)-(3.43). The objective of this IP model may concern either the satisfaction of technicians and/or jobs or the cost of constructing the assignment. In the former, the goal is to maximize the weighted satisfaction of players, and in the latter the cost is minimized. In this section, we define the satisfaction of a player, technician or job, and we address how to find optimal stable assignments.

In some applications, a weight for every technician-job pair is specified, similar to the weights of applicant-university pairs used by Baïou and Balinski (2000a). Let w_{tj} denote the profit of assigning technician t to job j. Then an optimal stable assignment is found by "*maximizing* $\sum_{(t,j)\in T\times J} w_{tj}x_{tj}$ *subject to the constraints* (3.34)-(3.43)". Furthermore, if the number of available technicians is much more than the required number, the concern becomes minimizing the cost of constructing the assignment, i.e. minimizing the skill waste. Then w_{tj} denotes the cost of assigning technician t to job j and an optimal stable assignment is constructed by "*minimizing* $\sum_{(t,j)\in T\times J} w_{tj}x_{tj}$ *subject to the constraints* (3.34)-(3.43)".

Optimal assignments based on satisfaction of players

In some cases, it may be desired to find optimal assignments in which the satisfaction of a set of players is maximized. For example, a company may look for maximizing the satisfaction of its technicians. First of all, below we define the satisfaction of a technician and the satisfaction of a job is defined similarly.

Definition 3.5.2. *(Technician satisfaction)*
The satisfaction of technician t when he is assigned to job j is denoted by γ_{tj}. It is given by

$$\gamma_{tj} = \frac{|J| - r(t, j)}{|J|} \tag{3.44}$$

In the $STJAP(n, n, n)$, the set of stable assignments does not qualify to form a lattice due to the skill requirement aspect of the problem. In some instances there may be several stable assignments, each could be preferred by disjoint subsets of technicians to any other stable assignments. This shows that the set of stable assignments may contain incomparable stable assignments, hence there may not be unique *technician-optimal* assignment in which every technician is at least as happy as in any other stable assignment. In this section, we aim to find one of these assignments in which the weighted satisfaction of technicians is maximized.

Let w_t denote the weight of technician t. Then an optimal stable assignment based on technicians' satisfaction can be found by "*maximizing* $\sum_{t\in T} w_t \sum_{j\in J} \gamma_{tj} x_{tj}$ *subject to the constraints* (3.34)-(3.43)". If the instance includes a technician-optimal stable assignment, then the optimal stable assignment that is constructed by our procedure corresponds to that technician optimal assignment.

Satisfaction of jobs. Let w_j denote the weight of job j. We find an optimal stable assignment based on the satisfaction of jobs by "*maximizing* $\sum_{j\in J} w_j \sum_{t\in T} \gamma_{jt} x_{tj}$ *subject to the constraints* (3.34)-(3.43)" where satisfaction of job j is found by $\gamma_{jt} = (|T| - r(j,t))/|T|$.

3.6 Computational results

All results are obtained on a laptop with Intel Core 2 Duo 1.6 GHz Processor, 4 GB RAM. The MIP models are solved using the solver CPLEX 12.1.0.

Instance generation. The multi-skill workforce schedules that are constructed by the combinatorial algorithm in Chapter 2 include day schedules, each being a technician-job assignment. We generated our instances from these schedules. After some experimentation, we decided to generate instances in which number of jobs is in $\{5, 10, 15\}$, number of technicians varies from 5 to 45, number of domains from 5 to 15, and number of levels from 2 to 7. Skill requirements of jobs and skill distribution among technicians depend on the specific day of schedule, so we did not have any control on them.

Preferences. Preferences are determined with randomized scores between technicians and jobs. The score between a technician and a job is calculated by finding the overlap between offered and required skills, and a randomized noise is added to allow some diversity.

Table 3.2 shows both the instance properties and the solution times. It has three parts; small-size instances with repeated runs, instances without stable assignments with repeated runs, and single moderate-size instances with stable assignment solutions. The first column of the table shows the source schedules that are constructed with the multi-skill workforce problem instances provided by France Telecom for 2007 ROADEF Challenge. An instance of our assignment problem corresponds to a workday schedule of the mentioned schedules. The following four columns labeled with $|\mathbb{S}|$, $|\mathbb{L}|$,$|T|$, and $|J|$ show respectively number of domains, number of levels, number of technicians, and number of jobs of instances. Following three columns, labeled with "Avg.Time(sec.)", "#Instances", and "Empty (%)" show respectively average

Table 3.2: Computational results

Schedule*	$\lvert\mathbb{S}\rvert$	$\lvert\mathbb{L}\rvert$	$\lvert T\rvert$	$\lvert J\rvert$	Avg.Time(sec.)	#Instances	Empty (%)
A8	5	4	5-14	5	0.06	2757	78
X6	6	6	5-18	5	0.09	1710	96
A9	5	4	5-19	5	0.07	3799	74
A8	5	4	15-17	5	0.14	676	93
B1	4	4	9-20	5	0.24	1772	92
B2	5	3	5-22	5	0.26	462	97
B5	7	4	8-16	5	0.36	39	72
B6	8	3	10-22	5	0.37	596	96
A8	5	4	14-19	10	0.27	720	96
A9	5	4	10-19	10	0.21	826	88
A9	5	4	17	15	0.35	12	75
B7	10	5	12-23	5	-	22	100
B2	5	3	15-30	10	-	247	100
B2	5	3	22-30	15	-	22	100
X8	15	7	17-28	5	-	946	100
B5	7	4	27	10	4.65	1	0
X8	15	7	29	5	4.06	1	0
B5	7	4	36	15	35.51	1	0
B5	7	4	44	15	98.95	1	0
B2	5	3	23	5	4.66	1	0
B2	5	3	28	5	55.87	1	0
B6	8	3	23	5	1.51	1	0
B6	8	3	25	5	31.44	1	0
B6	8	3	26	5	25.24	1	0
B6	8	3	25	10	1.82	1	0
B6	8	3	28	10	6.29	1	0
B6	8	3	28	10	7.63	1	0
B7	10	5	29	5	91.43	1	0
B9	5	5	60	15	>3600	1	0

*source schedule of the 2007 ROADEF Challenge problem

time for finding the solution for instances with stable sets, number of total instances generated, and the ratio of the instances with empty stable assignment set. We did not record the times needed by CPLEX to figure out that stable assignment set of the current instance is empty. Therefore the average times in Table 3.2 indicate how long did it take to find stable assignments.

On the solution times. If all results are considered, we see that the ratio of instances having no stable assignment is quite high, more than 85% in general. According to the average running times in Table 3.2, it takes less than one second, to find stable assignments for instances with $|T|$ up to 20. In problem instances with high number of technicians, roughly more than 25, it takes longer time to find stable assignments. The solution times in the third part of Table 3.2 vary from one second to 100 seconds. This high variance may be due to the differences in other instance properties that are not shown in Table 3.2 like skill distribution, skill requirements, and preferences. An important result to mention is that if the number of technicians increases up to 60, CPLEX could not find a solution after one hour.

3.7 Concluding remarks

In this study we analyse the Gale-Shapley stability in multi-skill workforce scheduling framework. Our stability definition is based on excluding the blocking cases, as in the definition by Gale-Shapley. A polynomial-time algorithm for a special case is developed and NP-hardness of the general problem is proved. Moreover, we define the set of all stable assignments by a set of linear inequalities. In order to construct optimal stable assignments based on specified weights, we formulated three MIP models that include the aforementioned set of linear inequalities in their constraint set.

Our experimentation shows that stable assignments, if any exist, of the instances with less than 20 technicians, around 10 jobs with at most 5 skill domains and 5 skill levels can be found in short time, less than one second, by solving the MIP that is explained in Section 3.5.2.

In our workforce assignment problem, the set of stable assignments may be empty. In such cases, the instability in a given assignment can be repaired by decreasing the number of blocking pairs. This can be done with a local search method that could be implemented in the following way:

Consider a sequence of jobs, say $\{j_1, j_2, \ldots, j_m\}$ and assume that under the current assignment, the technicians $\{t_1, t_2, \ldots, t_m\}$ work in the teams such that $t_i \in T(j_i)$. Note that the chain of replacements in which t_i replaces t_{i+1} for $i = 1, \ldots, m-1$ and t_m replaces t_1 results in another feasible assignment. Moreover, if $\langle t_i, j_{i+1} \rangle$ is a blocking pair for some i in the initially given assignment and if the replacements do not create any new blocking pairs, then the replacements lead to a decrease in the number of blocking pairs. A chain of such replacements can be called a "cycle". Figuring out the cycles in a given assignment those of not increasing the number of blocking pairs is a possible direction to investigate further. Here, another point to mention is that a cycle results possibly in new *critical skills* of jobs in the next assignment.

Besides the attempts to decrease the number of blocking pairs, some more actions

may be taken to obtain some stable assignments like rejecting some jobs, or preparing the plan for increasing the skills of technicians by training them as much as needed. If all these modifications to the schedule are specified as profits and costs, developing optimal strategies may be a topic of further researches under the multi-skill workforce scheduling framework.

The definition of idle technicians, in Section 3.2.5, provides a proper way for jobs to reject the technicians. Using this basic idea, the Gale-Shapley algorithm may be adapted to the multi-skill workforce assignment problem. It may be possible that the existence of technician-optimal stable assignments can be checked in a faster way than we suggest in Section 3.5.2.

Incompleteness in preferences is a direction for further studies. It will be useful to investigate which preference structures and preferences sizes make the assignment problem easy to solve or result in an empty stable set of assignments. This direction of research may also lead us to find a certificate to check if the stable assignments set is empty.

Chapter 4

Analysis of a dial-a-ride problem

Hunsaker and Savelsbergh (2002) discussed feasibility testing for a dial-a-ride problem under maximum wait time and maximum ride time constraints. In this chapter, we show that this feasibility test can be expressed as a shortest path problem in vertex-weighted interval graphs, which leads to a simple linear time algorithm.

4.1 Introduction

Dial-a-ride problems concern the dispatching of a vehicle to satisfy requests where an item (or a person) has to be picked up from a specific location and has to be delivered to some other specific location. Dial-a-ride problems arise in many practical application areas, as for instance shared taxi services, courier services, and transportation of elderly and disabled persons.

4.2 Problem description and notation

Hunsaker and Savelsbergh (2002) analyzed the following feasibility question for a dial-a-ride problem arising in a taxi company: An instance specifies a sequence of $2n+1$ events that have to be served (one after the other and in the given order) by a single vehicle. The first event is the dispatch of the vehicle from a central facility. The remaining $2n$ events are grouped into a set $\mathcal{P}$ of pairs $\langle i, j\rangle$ with $i < j$. In every pair $\langle i, j\rangle$ the earlier event i is the pickup and the later event j is the delivery of some fixed item. Here, two events in such a pair are not necessarily consecutive in the event sequence. The goal of the problem is to decide whether there exist $2n+1$ time points for these $2n+1$ events subject to the following three groups of constraints.

Time windows: The ith event ($1 \leq i \leq 2n+1$) must occur during a pre-specified time window between time points α_i and β_i with $\alpha_i \leq \beta_i$.

Riding times: The riding time from the ith to the $(i+1)$th event ($1 \leq i \leq 2n$) is $\gamma_{i,i+1}$. For every pickup and delivery pair $\langle i, j\rangle \in \mathcal{P}$, the time from pickup to delivery can be at most $\delta_{i,j} > 0$ time units.

Waiting times: At the ith pickup or delivery location ($2 \leq i \leq 2n+1$), the vehicle can wait for at most ω_i times units before departing.

Hunsaker and Savelsbergh (2002) design a sophisticated linear time algorithm that tests for the existence of a schedule that satisfies the three constraint families listed above. Tang et al. (2010) identify a crucial gap in the algorithm of Hunsaker and Savelsbergh (2002), and they also provide a concrete counter-example where the algorithm declares a feasible instance to be infeasible. As a partial repair Tang et al. (2010) provides another algorithm for the problem with a weaker quadratic running time $O(n^2)$. We note that such a quadratic running time is too slow for practical applications: The feasibility test shows up as a subproblem in improvement-based algorithms, and has to be performed many times.

We note that there are three points in which our problem description deviates from the one by Hunsaker and Savelsbergh (2002). First, our riding time bounds $\delta_{i,j}$ can be arbitrary numbers, whereas the riding time bounds in Hunsaker and Savelsbergh (2002) are proportional to the distances between pickup location and delivery location. In this respect, our model is slightly more general and contains the model in Hunsaker and Savelsbergh (2002) as a special case. Secondly, our waiting time bounds ω_i depend on the event, whereas the waiting time bounds in Hunsaker and Savelsbergh (2002) all are identical. Again this is a slight extension of the model in Hunsaker and Savelsbergh (2002), which also is mentioned in the discussion section of Hunsaker and Savelsbergh (2002). Thirdly, the problem in Hunsaker and Savelsbergh (2002) also incorporates item sizes and a capacity bound for the vehicle. These capacity constraints are independent of the timing and riding constraints, and they can be checked separately in $O(n)$ overall time. This independent subproblem has been discussed and settled in Hunsaker and Savelsbergh (2002), and there is no reason for discussing it here again.

Results in this chapter. We formulate the dial-a-ride feasibility test of Hunsaker and Savelsbergh (2002) as a system of linear inequalities (Sections 4.3 and 4.3.1), which by standard methods can be rewritten into a system of difference constraints (Section 4.4). By carefully analyzing the structure of these difference constraints (Section 4.5), we then transform the problem into a shortest path problem in a vertex-weighted interval graph. All in all, this yields the desired linear time $O(n)$ algorithm for the feasibility test.

4.2.1 Preprocessing of time windows

Suppose that $\alpha_{i+1} < \alpha_i$ for some i with $1 \leq i \leq 2n$. Since the vehicle cannot serve the ith event before time α_i, it cannot arrive at the $(i+1)$th location before time α_i. Hence we may update the data as $\alpha_{i+1} := \alpha_i$. All updates (for all values of i) can be performed by a single $O(n)$ time pass over the locations in increasing order of i. A symmetric argument shows that whenever $\beta_{i+1} < \beta_i$ holds for some i with $1 \leq i \leq 2n$, then we may update $\beta_i := \beta_{i+1}$. Furthermore, all such updates can be done during a single $O(n)$ time pass over the locations in decreasing order of i.

Therefore we assume throughout that the left and right endpoints of the time

intervals form two non-decreasing sequences $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{2n+1}$ and $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_{2n+1}$.

4.3 Linear equations and inequalities

We formulate the dial-a-ride problem as a system of linear equations and inequalities. The ith event ($1 \leq i \leq 2n+1$) is described by three real variables: The arrival time A_i and the departure time D_i of the vehicle at the location of the ith event, and the time point E_i at which the actual pickup/delivery occurs. For the first event, we identify the variables E_1 and D_1 so that they coincide with the dispatch time of the vehicle from the central facility.

$$A_1 = \alpha_1 \quad \text{and} \quad E_1 = D_1 \tag{4.1}$$

Clearly the ith event must fit between the arrival and the departure time of the vehicle.

$$A_i \;\leq\; E_i \;\leq\; D_i \qquad\qquad i = 2, \ldots, 2n+1. \tag{4.2}$$

Now let us express the constraints of the problem. The ith event must occur during its time window $[\alpha_i, \beta_i]$.

$$\alpha_i \;\leq\; E_i \;\leq\; \beta_i \qquad\qquad i = 1, \ldots, 2n+1. \tag{4.3}$$

The riding time $\gamma_{i,i+1}$ from the ith to the $(i+1)$th event yields

$$A_{i+1} \;=\; D_i + \gamma_{i,i+1} \qquad\qquad i = 1, \ldots, 2n. \tag{4.4}$$

For every pickup and delivery pair $\langle i, j \rangle \in \mathcal{P}$, the time from pickup to delivery is constrained by

$$E_j \;\leq\; E_i + \delta_{i,j} \qquad\qquad \forall\, \langle i, j \rangle \in \mathcal{P}. \tag{4.5}$$

Finally, the waiting time constraints yield

$$D_i \;\leq\; A_i + \omega_i \qquad\qquad i = 2, \ldots, 2n+1. \tag{4.6}$$

The dial-a-ride instance has a feasible solution, if and only if the linear system (4.1)–(4.6) with $O(n)$ constraints has a feasible solution over the real numbers.

4.3.1 An equivalent linear inequality system

Our next step is to rewrite the system (4.1)–(4.6) into an equivalent but simpler system centered around a new set of variables: For $i = 1, \ldots, 2n+1$ we introduce the non-negative variable x_i to measure the total waiting time between time point α_1 and time point D_i (that is, the total time that the vehicle did not spent on driving before departing from the ith location). Furthermore, for $1 \leq i \leq j \leq 2n+1$ we introduce the constant Γ_i to denote the overall riding time to move through the locations $1, 2, \ldots, i$.

$$\Gamma_i \;=\; \sum_{k=1}^{i-1} \gamma_{k,k+1}.$$

Then the arrival times $A_2, \ldots, A_{2n+1}$ and the departure times $D_1, \ldots, D_{2n+1}$ can be rewritten as

$$A_i = x_{i-1} + \Gamma_i \qquad \text{and} \qquad D_i = x_i + \Gamma_i. \tag{4.7}$$

Each of the events $E_2, \ldots, E_{2n+1}$ is either a pickup or a delivery that is constrained by (4.2), (4.3), and (4.5). Combining (4.2), (4.3) results in

$$\max\{A_i, \alpha_i\} \le E_i \le \min\{D_i, \beta_i\}, \quad i = 1, \ldots, 2n+1. \tag{4.8}$$

The above interval is non-empty or E_i is feasible, if and only if $A_i \le D_i$ and $A_i \le \beta_i$ and $\alpha_i \le D_i$ hold. So we can eliminate events E_i using Fourier-Motzkin elimination by considering these inequalities and rewriting the arrival and the departure times as in (4.7). This elimination preserves the feasibility of events E_i for $i = 2, \ldots, 2n+1$ and gives us the following constraints

$$x_{i-1} \le x_i \qquad \text{and} \qquad x_{i-1} \le \beta_i - \Gamma_i \qquad \text{and} \qquad \alpha_i - \Gamma_i \le x_i. \tag{4.9}$$

If the ith event is a pickup, then we may delay it as much as possible by setting $E_i := \min\{D_i, \beta_i\}$. If the jth event is a delivery, then we may schedule it as early as possible and set $E_j := \max\{A_j, \alpha_j\}$. Then (4.5) becomes

$$\max\{A_j, \alpha_j\} = E_j \le E_i + \delta_{i,j} = \min\{D_i, \beta_i\} + \delta_{i,j}, \quad \forall \langle i, j\rangle \in \mathcal{P}. \tag{4.10}$$

The above inequalities mean that for all $\langle i, j\rangle \in \mathcal{P}$ the following conditions are satisfied: $A_j \le D_i + \delta_{i,j}$, $A_j \le \beta_i + \delta_{i,j}$, $\alpha_j \le D_i + \delta_{i,j}$, and $\alpha_j \le \beta_i + \delta_{i,j}$. Note that the last condition does not depend on any variable (and if it is violated, then the system is trivially infeasible). The other three conditions yield the following constraints.

$$x_{j-1} \;\le\; x_i + \delta_{i,j} + \Gamma_i - \Gamma_j \qquad \forall\ \langle i, j\rangle \in \mathcal{P} \tag{4.11}$$

$$x_{j-1} \;\le\; \beta_i + \delta_{i,j} - \Gamma_j \qquad \forall\ \langle i, j\rangle \in \mathcal{P} \tag{4.12}$$

$$\alpha_j - \delta_{i,j} - \Gamma_i \;\le\; x_i \qquad \forall\ \langle i, j\rangle \in \mathcal{P} \tag{4.13}$$

The remaining constraints (4.1), (4.4), (4.6) are handled as follows. Constraints (4.1) and (4.3) yield

$$0 \;\le\; x_1 \;\le\; \beta_1 - \alpha_1. \tag{4.14}$$

Constraint (4.4) is satisfied because of (4.7), and becomes vacuous. Finally the waiting time constraints (4.6) translate into

$$x_i \;\le\; x_{i-1} + \omega_i \qquad i = 2, \ldots, 2n+1. \tag{4.15}$$

Consequently, the dial-a-ride instance has a feasible solution, if and only if the linear system (4.9),(4.11)–(4.15) with $O(n)$ constraints has a feasible solution over the real numbers.

4.4 Difference constraint systems

Every inequality in (4.9),(4.11)–(4.15) is either an upper bound constraint $x_i \leq U_i$, or a lower bound constraint $L_i \leq x_i$, or a difference constraint $x_j - x_i \leq D_{i,j}$. By applying a standard technique, we will now transform all upper and lower bound constraints into difference constraints.

For this purpose, we introduce two new variables x_0 and x_{2n+2}. Variable x_0 represents the value 0, and hence is a lower bound on all other variables. Variable x_{2n+2} represents the value $K := \beta_{2n+1} - \alpha_1$, and hence is an upper bound for all other variables. We create the two new constraints

$$x_{2n+2} - x_0 \leq K \qquad \text{and} \qquad x_0 - x_{2n+2} \leq -K, \tag{4.16}$$

which together enforce $x_{2n+2} - x_0 = K$. Every upper bound constraint $x_i \leq U_i$ in (4.9),(4.11)–(4.15) is replaced by a corresponding constraint

$$x_i - x_0 \ \leq \ U_i. \tag{4.17}$$

Every lower bound constraint $L_i \leq x_i$ in (4.9),(4.11)–(4.15) is replaced by a corresponding constraint

$$x_{2n+2} - x_i \ \leq \ K - L_i. \tag{4.18}$$

We will refer to the following constraints short as the difference constraint system DCS:

- the constraints (4.16),
- the new difference constraints (4.17) and (4.18),
- the old difference constraints in (4.9), (4.11)-(4.15).

Lemma 4.4.1. *The following four statements are pairwise equivalent.*

(i) The original dial-a-ride instance has a feasible solution.

(ii) The system (4.9),(4.11)–(4.15) has a feasible solution over the real numbers.

(iii) DCS has a feasible solution with $x_0 = 0$ and $x_{2n+2} = K$.

(iv) DCS has a feasible solution.

4.4.1 Feasibility test via finding negative-weight cycles

There is a close connection between difference constraint systems and negative-weight cycles in directed graphs; as explained in Section 24.4 of Cormen et al. (2009).

We create for every variable x_i ($0 \leq i \leq 2n+2$) a corresponding vertex i. We create for every difference constraint $x_j - x_i \leq D_{i,j}$ an arc from vertex i to vertex j with weight $D_{i,j}$. The relation between negative-weight cycle and feasible of difference constraint systems is given by the following theorem.

Theorem 4.4.2. *(Chapter 24, Cormen et al. (2009)) Given a system of difference constraints, let $G = (V, A)$ be the corresponding constraint graph. If G contains no negative-weight cycles, then*

$$x = (SP(v_0, v1), SP(v_0, v_2), \ldots, SP(v_0, v_n)) \tag{4.19}$$

is a feasible solution for the system where $SP(v_0, vi)$ denotes the length of the shortest path from v_0 to v_i. If G contains a negative-weight cycle, then there is no feasible solution for the system.

In our case, for the directed graph G we have $O(|V|) = O(|A|) = n$. Hence a straightforward application of the Bellman-Ford algorithm or of the Goldberg-Radzik algorithm would yield an $O(n^2)$ time feasibility test.

4.5 A linear time feasibility test

To get a linear time algorithm, we look a little bit deeper into the structure of DCS and the corresponding directed graph G. A difference constraint $x_j - x_i \leq D_{i,j}$ is a *forward* constraint (and the corresponding arc (i, j) is a forward arc), if $i < j$ holds. Otherwise we are dealing with a *backward* constraint (and a corresponding backward arc). Now let us go through all constraints in DCS.

- The difference constraints in (4.9) are of the form $x_{i-1} - x_i \leq 0$. They are backward constraints, and their arc weights are always zero.
- The constraints in (4.11) are forward constraints. If $\delta_{i,j} < \Gamma_j - \Gamma_i$ then the system is infeasible. Hence we may assume that all corresponding arc weights are non-negative.
- The constraints (4.15) are forward constraints, and the corresponding arc weights ω_i are non-negative.
- In (4.16) we have one forward constraint with positive arc weight, and one backward constraint with negative arc weight.
- The difference constraints in (4.17) arise from upper bounds, and are forward constraints. We may assume that all corresponding arc weights are non-negative (since otherwise the system would be infeasible).
- The difference constraints in (4.18) arise from lower bounds, and are forward constraints. Again, we assume that all corresponding arc weights are non-negative (as otherwise the system would be infeasible).

Next, we summarize our above observations.

Constraint			Arc type	Weight
$x_{i-1} - x_i$	≤ 0,	$i = 2, \ldots, 2n+1$	backward	zero
$x_{j-1} - x_i$	$\leq \delta_{i,j} + \Gamma_i - \Gamma_j$,	$\forall \langle i, j \rangle \in \mathcal{P}$	forward	non-negative
$x_i - x_{i-1}$	$\leq \omega_i$,	$i = 2, \ldots, 2n+1$	forward	non-negative
$x_{2n+2} - x_0$	$\leq K$,	unique	forward	positive
$x_0 - x_{2n+2}$	$\leq -K$,	unique	backward	negative
$x_{i-1} - x_0$	$\leq \beta_i - \Gamma_i$,	$i = 2, \ldots, 2n+1$	forward	non-negative
$x_{j-1} - x_0$	$\leq \beta_i + \delta_{i,j} - \Gamma_j$,	$\forall \langle i, j \rangle \in \mathcal{P}$	forward	non-negative
$x_1 - x_0$	$\leq \beta_1 + \alpha_1$,	unique	forward	non-negative
$x_{2n+2} - x_i$	$\leq K - \alpha_i + \Gamma_i$,	$i = 1, \ldots, 2n+1$	forward	non-negative
$x_{2n+2} - x_i$	$\leq K + \Gamma_i + \delta_{i,j} - \alpha_i$,	$\forall \langle i, j \rangle \in \mathcal{P}$	forward	non-negative

4.5.1 Feasibility test via shortest paths

Summarizing, all forward arcs have non-negative weights, and with a single exception all backward arcs have weight zero and are of the form $(i, i-1)$. The only arc with negative weight is the backward arc from vertex $2n+2$ to vertex 0 in (4.16).

Hence every cycle of negative weight must consist of this arc plus some directed path from vertex 0 to vertex $2n+2$. Recall that the DCS is infeasible, if and only if graph G contains a negative-weight cycle, which is true if and only if the shortest path from vertex 0 to vertex $2n+2$ along arcs with non-negative weights has length strictly smaller than K. This observation in combination with fast shortest path algorithms in directed graphs of Fredman and Tarjan (1987) yields a time complexity of $O(n \log n)$.

Our next goal is to establish a connection to interval graphs, which will yield a linear time complexity. With every forward arc (i, j) we associate the closed interval $[i, j]$.

Lemma 4.5.1. *Among all shortest paths from vertex 0 to vertex $2n+2$ (that only use arcs with non-negative weights) let P^* be a path with the smallest number of forward arcs. For $k \geq 1$, let (i_k, j_k) denote the kth forward arc traversed by P^*. Then for any pair of consecutive forward arcs (i_k, j_k) and (i_{k+1}, j_{k+1}), the two associated intervals $[i_k, j_k]$ and $[i_{k+1}, j_{k+1}]$ have non-empty intersection.*

Proof. At vertex j_k, P^* moves either directly forward by using a forward arc or first backward by using some zero-cost backward arcs and then forward at a vertex smaller than j_k. In the former, we have $j_k = i_{k+1}$ and clearly intervals $[i_k, j_k]$ and $[i_{k+1}, j_{k+1}]$ intersect at one point. In the latter, intervals $[i_k, j_k]$ and $[i_{k+1}, j_{k+1}]$ will have an empty intersection if and only if P^* uses a number of zero-cost backward arcs such that $i_{k+1} < j_{k+1} < i_k$. In such a case, there is no point of using the forward arc (i_k, j_k) , since P^* could move directly from i_k to i_{k+1} by skipping the arc (i_k, j_k). This would yield another shortest path with a smaller number of traversed forward arcs and results in a contradiction. □

Lemma 4.5.2. *Consider a sequence of forward arcs (i_k, j_k) $(k = 1, \ldots, p)$ with overall weight W, such that the first arc starts in $i_1 = 0$ and the last arc ends in $j_p = 2n+2$, and such that for any two consecutive arcs in the sequence the associated intervals*

have non-empty intersection. Then the directed graph contains a path from vertex 0 *to vertex* $2n + 2$ *of weight* W*.*

Proof. We show that it is possible to find a path using the sequence of forward arcs (i_k, j_k) $(k = 1, \ldots, p)$. As showed in by Lemma 4.5.1, intervals $[i_k, j_k]$ and $[i_{k+1}, j_{k+1}]$ intersect, we see that $j_k \geq i_{k+1}$. Since moving backward has zero length, it is possible to move from vertex j_k to vertex i_{k+1} by a sequence of backward arcs without a cost of length. Therefore such a path has a length of W. □

Proposition 4.5.3. *In the constraint graph* G*, the shortest path problem, from vertex* 0 *to vertex* $2n + 2$*, reduces to the minimum-weight sequence of intersecting intervals problem in which every sequence starts with an interval whose first vertex* 0*, and ends with an interval whose last vertex is* $2n + 2$*.*

Proof. The proposition directly follows the Lemmas 4.5.1 and 4.5.2. □

We note that the path lengths in graph G are measured in arc weights.

Vertex-weighted interval graph. In light of Proposition 4.5.3, we construct a vertex-weighted interval graph G^*. For every forward arc (i, j) with weight w, the interval graph G^* contains the associated interval $[i, j]$, with weight w. The requirement for sequences "starting with an interval whose first vertex 0, and ending with an interval whose last vertex is $2n + 2$" is adapted by defining the degenerate intervals $[0, 0]$ and $[2n + 2, 2n + 2]$ both of weight 0. The problem turns out to be the shortest path problem is interval graph G^* from interval $[0, 0]$ to interval $[2n + 2, 2n + 2]$. Note that the length of paths is measured in interval/vertex weights.

Atallah et al. (1995) show that the length of the shortest path in a vertex-weighted interval graph can be computed in linear time:

Theorem 4.5.4. *(Atallah et al. (1995))Given a set of intervals with weights, an ordering of these intervals according to their left endpoints, and an ordering of these intervals according to their right endpoints, the single-source shortest path problem can be solved in linear time.* □

The single-source shortest path problem consists in computing the shortest paths from a given source-interval to all other intervals, where the length of a path is the sum of all interval weights along the path.

Since the endpoints of all intervals are intervals in the range 0 to $2n + 2$, it is easy to sort these intervals according to their left or right endpoints in linear time $O(n)$; this can for instance be done by counting sort or by some variant of bucket sort (see Section 8 of Cormen et al. (2009)). Altogether this yields the main result of this chapter.

Theorem 4.5.5. *The feasibility test for the dial-a-ride problem of Hunsaker and Savelsbergh under time window constraints, riding time constraints, and waiting time constraints can be performed in* $O(n)$ *time.* □

4.6 Concluding remarks

In the previous sections, we showed that the feasibility testing of the dial a ride problem can be done in linear time by solving the corresponding shortest path problem in a vertex-weighted interval graph. The vertex weights in the interval graph correspond to the waiting time bounds in the dial a ride problem, hence the length of a shortest path in the interval graph is simply the total waiting time from the first vertex to the last. Considering this, it is possible to construct a schedule for a dial a ride instance by adding degenerate zero-length intervals $[i, i]$ for $i = 1, \ldots, 2n + 1$. Then the length of the shortest path from interval $[0, 0]$ to interval $[i, i]$ will give us the value of the variable x_i in the difference constraint system DCS. The algorithm of Atallah et al. (1995) finds the shortest paths from interval $[0, 0]$ to all intervals $[i, i]$ for $i = 1, \ldots, 2n + 1$ in linear time. Once we know the total waiting time for every location i, by some straightforward backwards calculations we can determine the corresponding feasible solutions for the inequality systems in Sections 4.3 and 4.3.1. Consequently, besides the feasibility test, a feasible schedule can be constructed in linear time for a dial a ride problem instance.

Chapter 5

Analysis of a vehicle refueling problem

This chapter deals with a vehicle refueling problem that involves refueling decisions while traveling on a route with a fixed order of cities with varying fuel prices. Tank capacity of the vehicle depends on the city where the departure occurs. Additionally, it is assumed that in every city there is an upper bound on the available fuel amount. The goal is to reach the destination with minimum fuel cost.

We present an $O(n \log n)$ time greedy algorithm for this problem. Our interpretation of the greedy algorithm is very intuitive and easy to implement. This greedy algorithm solves the special case that is studied by Lin et al. (2007) in $O(n)$ time.

5.1 Introduction

In this chapter, we study a vehicle refueling problem in which the vehicle travels on the fixed route of cities. A problem instance specifies the set of cities in which every city is indexed according to their place in a fixed route. The distance from one city to the next is given as the necessary fuel amount between these cities. A solution specifying non-negative refueling amounts for every city is feasible if and only if the following constraints are respected:

Availability: The refueling amount in a city cannot be more than the upper bound of the fuel amount in that city.

Reachability: Previous refueling amounts before a city must be enough to reach that city.

Capacity: In the departure from a city, the fuel amount in tank should not exceed the tank capacity that is allowed for the travel to next city.

Refueling of airplanes with pre-determined flight routes nice fits our problem setting, especially the varying fuel prices and varying tank capacities. Airplanes visit airports of different countries and it is natural to assume that fuel prices may vary among these airports. Moreover, the maximum refueling amount at an airport depends on the number of passengers in the flight.

Our problem definition generalizes the vehicle refueling problem studied by Lin et

al. (2007) in several aspects. First of all, we allow tank capacities to vary, whereas they are all identical in the definition of Lin et al. (2007). Moreover, we consider upper bounds on the fuel amounts in cities, whereas the fuel amounts are assumed to be unlimited by Lin et al. (2007).

Results of this chapter. We represent a greedy algorithm for the vehicle refueling problem. Although in the literature we encounter algorithms that are technically the same as the one we present, our interpretation is very intuitive and easy to implement. We prove the correctness of the greedy algorithm in three ways; using duality, network flows, and convexity. Moreover we analyse solutions of the greedy algorithm and we provide a dual interpretation of the problem.

This chapter is organized as follows. Section 5.2 describes the main ingredients of the vehicle refueling problem and the LP formulation of the problem is given. In Section 5.3, we showed that the vehicle refueling problem is reduced to a minimum cost flow problem on a specific network. Our greedy algorithm is presented in Section 5.4. Correctness proof by duality is given in Section 5.4.2, by network flow in Section 5.4.3 and by convexity in Section 5.4.4. Conclusions are discussed in Section 5.5.

5.2 Problem description

A vehicle travels along a fixed-route of cities $S = \{1, \ldots, n\}$ in the given order. The travel from city w to $w+1$ requires d_w units of fuel amount. In city w, the available fuel amount is U_w units, the price of one unit fuel amount is $p_w \geq 0$. Tank capacity between cities w and $w+1$ is T_w units of fuel amount. It is assumed that all aforementioned parameters are integers.

Objective. A solution to our problem specifies x_w for every city $w \in S$ that denotes the purchased fuel amounts bought in cities. The objective is minimizing the travel cost: $\sum_{w \in S} p_w x_w$. Below, we give the LP formulation of the vehicle refueling problem.

$$\begin{aligned}
&\text{minimize} && \sum_{w \in S} p_w x_w \\
&\text{subject to:} \\
&&\sum_{t=1}^{w} x_t \geq \sum_{t=1}^{w} d_t, && \forall w \in S && (5.1)\\
&&\sum_{t=1}^{w} x_t \leq \sum_{t=1}^{w-1} d_t + T_w, && \forall w \in S && (5.2)\\
&& x_w \leq U_w, && \forall w \in S && (5.3)\\
&& x_w \geq 0, && \forall w \in S && (5.4)
\end{aligned}$$

Constraints (5.1) ensure that every city on the route is visited. Constraints (5.2) force that the fuel amounts in the tank do not exceed the allowed tank capacities during departures. Finally, constraints (5.3) do not allow the refueling amounts to be more than the upper bounds of fuel amounts in cities.

Lin et al. (2007) consider a special case of our vehicle refueling problem and propose a greedy algorithm for this special case.

Theorem 5.2.1. *(Lin et al. (2007)) A special case of the vehicle refueling problem with $T_w = T$ and $U_w \equiv \infty$ for all $w \in S$ can be solved in linear time.*

The authors propose an algorithm in which refueling decisions are made in a complicated way. Every city distinguishes two cities among the successive ones: the *first cheaper* city and the *last reachable* city with full tank. In a city, if the first cheaper city is later than the last reachable city, the tank is filled to the full capacity and the vehicle goes to next city, otherwise the refueling is done, if necessary, in an amount to reach the first cheaper city and the vehicle directly goes to the first cheaper city.

The vehicle refueling problem is equivalent to a specific single-item lot sizing problem. The equivalence can be easily seen if the *fuel* is considered as the *item* to produce and/or to keep in the inventory such that every *city* corresponds to a *time interval* and the *distance* to travel from a certain city to the next one becomes the *demand* in the corresponding time interval. Then every *refueling* can be seen as a *production* with zero setup cost and with the linear cost function. Both production and inventory are capacitated and the capacities vary over time intervals.

The equivalent lot sizing problem has been studied by Sedeo-Noda et al. (2003) and the authors propose an $O(n \log n)$ time greedy algorithm that is technically the same as the one we present in this chapter.

The vehicle refueling problem can expressed as a flow problem on a specific network that is given by Sedeo-Noda et al. (2003). If the general algorithms are used, the minimum cost flow problem on this network can be solved in $O(n^2 \log^2 n)$ time. Ahuja and Hochbaum (2008) give a minimum cost flow algorithm that is specially designed for the specific network for the vehicle refueling problem. The algorithm uses red-black tree data structure (for details see Cormen et al. (2009)) and runs in $O(n \log n)$ time.

5.3 Network representation

In this section we show that the vehicle refueling problem is equivalent to minimum cost flow problem on the network in Figure 5.1. The network in Figure 5.1 is transportation network with a source node s, a sink node t, and transshipment nodes between s and t. Every city in the vehicle refueling problem is represented by two nodes. For city w, firstly the refueling is done at node v_w and the fuel to reach $w+1$ is dropped at node $v_{w,w+1}$. The arc capacities are defined accordingly.

Ahuja and Hochbaum (2008) developed an $O(n \log n)$ time algorithm to solve the minimum cost flow on the network in Figure 5.1.

Next, we give the decision versions of the vehicle refueling problem and the minimum cost flow problem before reducing the former to the latter.

Problem: Vehicle refueling problem

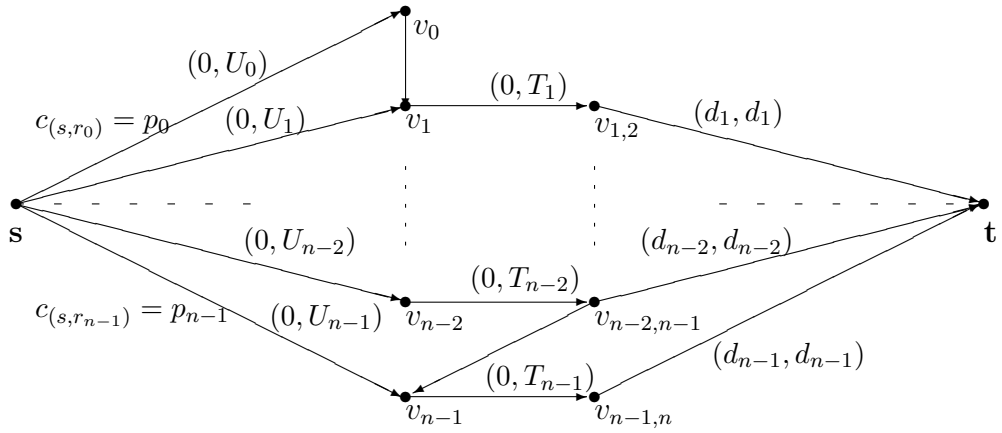

Figure 5.1: Network representation of the vehicle refueling problem

INSTANCE: Given a numerical bound k, the set $S = \{1, \ldots, n\}$ of cities such that the indices give tell us the fixed route to travel, the set $T = \{T_1, \ldots, T_n\}$ of tank capacities, the set $P = \{p_1, \ldots, p_n\}$ of non-negative fuel prices, the set $D = \{d_1, \ldots, d_n\}$ of distances, and the set $U = \{U_1, \ldots, U_n\}$ of available fuel amounts in cities. The vehicle refuels $0 \leq x_w$ units of fuel in city w, for all $w \in S$.

QUESTION: Does there exists a refueling policy, satisfying the constraints (5.1)-(5.4), such that $\sum_{w \in S} p_w x_w$ is no more than k?

PROBLEM: MINIMUM COST FLOW PROBLEM (MCF)

INSTANCE: Given a numerical bound k, the set $N = \{s, \ldots, t\}$ of nodes, and the set A of arcs. For an arc (v, w) in A, it costs $c_{(v,w)}$ to send one unit flow, the lower bound of any flow is $l_{(v,w)}$ and an upper bound is $u_{(v,w)}$.

QUESTION: Does there exists a flow f that satisfies conservation constraints $\sum_{(v,w) \in A} f_{(v,w)} = \sum_{(w,y) \in A} f_{(w,y)}$ for all $w \in S \setminus \{s, t\}$ and respects bounds on every arc, such that $\sum_{(v,w) \in A} c_{(v,w)} f_{(v,w)}$ is no more than k?

Theorem 5.3.1. *Vehicle refueling problem $\propto MCF$ on the network in Figure 5.1.*

Proof. First of all we explain how to construct the network in Figure 5.1 from a vehicle refueling problem instance. Next we will show that YES instance of the vehicle refueling problem corresponds to YES instance of the MCF and vice versa.

- Create the nodes $\{s, v_0, t\}$.
- For every city $w \in S \setminus \{n\}$, create transshipment nodes v_w, $v_{w,w+1}$.
- For every city $w \in S \setminus \{n, n-1\}$, create the arc $(v_{w,w+1}, v_{w+1})$.
- For every city $w \in S \setminus \{n\}$, create the arc (s, v_w) with cost $c_{(s,v_w)} = p_w$ and upper bound $u_{(s,v_w)} = U_w$, the arc $(v_w, v_{w,w+1})$ with upper bound $u_{(v_w,v_{w,w+1})} = T_w$, and the arc $(v_{w,w+1}, t)$ with lower and upper bounds $l_{(v_{w,w+1},t)} = u_{(v_{w,w+1},t)} = d_w$.
- Costs, lower bounds, and upper bounds of the arcs, not mentioned above, are zero.

Let a flow f be given for a YES instance of the MCF. We know f satisfies the conservation at transshipment nodes and respects the are capacities. We convert the flow f to a vehicle refueling solution by $x_w = f_{(s,v_w)}$ for every city $w \in S \setminus \{n\}$.

Availability constraints. Due to the upper bounds of arcs (s, v_w) for all $w \in S \setminus \{n\}$, since we have $x_w = f_{(s,v_w)} \leq u_{(s,v_w)} = U_w$.

Capacity constraints. Let us consider the conservation at nodes $v_{(w,w+1)}$ and $v_{(w+1)}$, so we have

$$f_{(v_{w+1},v_{w+1,w+2})} - f_{(v_w,v_{w,w+1})} = f_{(s,v_{w+1})} - f_{(v_{w,w+1},t)} = x_{w+1} - d_w \tag{5.5}$$

The above equation shows us a recursion by substituting the term $f_{(v_w,v_{w,w+1})}$ using the same type equation, and we obtain

$$f_{(v_{w+1},v_{w+1,w+2})} = \sum_{t=1}^{w+1} x_t - \sum_{t=1}^{w} d_t \tag{5.6}$$

The upper bound constraint of arc $(v_{w+1}, v_{w+1,w+2})$ provides us the same inequality as the capacity constraint in (5.2).

Reachability constraints. Now we consider the flow conservation at nodes $v_{w,w+1}$ and v_w. If we combine both conservation equations, we obtain

$$f_{(v_{w,w+1},v_{w+1})} - f_{(v_{w-1,w},v_w)} = f_{(s,v_w)} - f_{(v_{w,w+1},t)} = x_w - d_w \tag{5.7}$$

.

The above equation shows us a recursion by substituting the term $f_{(v_{w-1,w},v_w)}$ using the same type equation, and we obtain

$$f_{(v_{w,w+1},v_{w+1})} = \sum_{t=1}^{w} x_t - \sum_{t=1}^{w} d_t \tag{5.8}$$

.

The lower bound constraint $(v_{w,w+1}, v_{w+1})$ implies $0 \leq f_{(v_{w,w+1},v_{w+1})}$ and (5.8) imply the reachability constraints in (5.1).

Refueling cost. We have $\sum_{w\in S\setminus\{n\}} p_w x_w = \sum_{w\in S\setminus\{n\}} f_{(s,v_w)}$.

Consequently, we showed that a YES instance of the MCF corresponds to a YES instance of the vehicle refueling problem.

Next let us show that YES instance of the vehicle refueling problem corresponds to a YES instance of the MCF. In a solution of the YES instance, we have refueling values x_w for every city in $S \setminus \{n\}$. We define a flow on the network in Figure 5.1 with the following values

$$f_{(s,v_w)} = x_w, f_{(v_{w,w+1},t)} = d_w, \quad \forall w \in S \setminus \{n\} \tag{5.9}$$

$$f_{(v_w,v_{w,w+1})} = \sum_{t=1}^{w} x_t - \sum_{t=1}^{w-1} d_t, f_{(v_{w,w+1},v_{w+1})} = \sum_{t=1}^{w} x_t - \sum_{t=1}^{w} d_t, \forall w \in S \setminus \{n\} \tag{5.10}$$

.

Arc capacities. Lower bounds of arcs $(v_w, v_{w,w+1})$ and $(v_{w,w+1}, v_{w+1})$ are satisfied by the reachability constraints in (5.1). Upper bound constraints of the arcs (s, v_w) are satisfied by availability constraints in (5.3) and the upper bounds constraints of the arcs $(v_w, v_{w,w+1})$ are satisfied by capacity constraints in (5.2).

Flow conservation. Let us check the conservation at nodes v_w and $v_{w,w+1}$.

$$f_{(s,v_w)} + f_{(v_{w-1,w},v_w)} - f_{(v_w,v_{w,w+1})} = x_w + (\sum_{t=1}^{w-1} x_t - \sum_{t=1}^{w-1} d_t) - (\sum_{t=1}^{w} x_t - \sum_{t=1}^{w-1} d_t) = 0 \tag{5.11}$$

$$f_{(v_w,v_{w,w+1})} - f_{(v_{w,w+1},t)} - f_{(v_{w,w+1},v_{w+1})} = (\sum_{t=1}^{w} x_t - \sum_{t=1}^{w-1} d_t) - d_w - (\sum_{t=1}^{w} x_t - \sum_{t=1}^{w} d_t) = 0 \tag{5.12}$$

In the above equations, we substitute the flow amounts defined in (5.9) and in (5.10). Consequently, we showed that a YES instance of the vehicle refueling problem corresponds to a YES instance of the MCF. □

5.4 The greedy algorithm

In this section we explain a greedy algorithm for the vehicle refueling problem.

Fuel amounts separately kept in tank. In the greedy algorithm, the fuel amounts from different cities are assumed to be kept without being mixed each other in the tank. These amounts can be decreased partially whenever needed. The strategy of the algorithm is to keep the maximum amount of the fuel in tank tentatively. We note that keeping a fuel amount in tank does not imply that it will certainly be bought later.

Removals and purchases. Upon an arrival at a city, if the fuel amount in the tank plus the available fuel amount in that city is higher than the tank capacity, the

excess is removed. In this *removal* process, the most expensive fuel amount in the tank is firstly removed. Removals continue until the total fuel amount drops exactly to the tank capacity. Next, the fuel amount that is needed to travel to next city is purchased. The *purchase* process starts with the cheapest fuel in tank. If the cheapest fuel amount is not enough to reach the next city, the next cheapest fuel is purchased. The purchases continue until the total amount of purchased fuel is exactly enough to reach the next city. Removal process occur whenever necessary, but purchase processes occur in every city on the route. Finally, in the last city, all fuel in the tank is removed.

Table 5.1: Sets, indices, parameters, and variables

Set	
S	Set of cities, $S = \{1, 2, \ldots, n\}$
Indices	
v, w, z	City indices,
List	
L	List of pairs $\langle v, y_v \rangle$, where $\langle v, y_v \rangle \in S \times Z_+$
Parameters	
p_w	Fuel price in city w, $w \in S$
U_w	Upper bound on the fuel amount in city w, where $U_w \in \mathbb{Z}_+$
T_w	Tank capacity from city w to city $w+1$, where $T_w \in \mathbb{Z}_+$
d_w	Fuel amount needed to travel from w to $w+1$, $d_w \in \mathbb{Z}_+$

Algorithm 1.

Input: See Table 1.

$w \leftarrow 1$, $L \leftarrow \emptyset$; // initialization
for $w < n$ **do**
  $L \leftarrow L \cup \{\langle w, U_w \rangle\}$;
  RemoveExcess(L, T_w); // removals
  Purchase(L, d_w); // purchases
  $w \leftarrow w + 1$;
end
RemoveExcess$(L, 0)$; // removals in destination

RemoveExcess(L,T)

while $\sum_{\langle v, y_v \rangle \in L} y_v > T$ **do**
  $t \leftarrow \text{Argmax}\{p_v | \langle v, y_v \rangle \in L, y_v > 0\}$;
  if $y_t > \sum_{\langle v, y_v \rangle \in L} y_v - T$ **do**
    $y_t \leftarrow y_t - (\sum_{\langle v, y_v \rangle \in L} y_v - T)$;
  else
    $y_t \leftarrow 0$;
  end
end

Purchase(L,d)

while $d > 0$ **do**
  $t \leftarrow \text{Argmin}\{p_v | \langle v, y_v \rangle \in L, y_v > 0\}$;
  if $y_t > d$ **do**
    $d \leftarrow 0$, $y_t \leftarrow y_t - d$;
  else
    $y_t \leftarrow 0$, $d \leftarrow d - y_t$;
  end
end

The greedy algorithm is described by Algorithm 1. The input of Algorithm 1 is listed in Table 5.1. In step 3, we see that all available fuel amount in the city is inserted into the list L upon arrival and the removal process starts in step 4. In step R1, it is checked if the total fuel amount exceeds the allowed tank capacity. If tank capacity is exceeded, the most expensive fuel in tank is determined in step R2. This fuel is removed partially in step R4, if the excess is less than the amount of that fuel. Otherwise, all amount of the most expensive fuel is removed in step R6 and the removal process continues by finding next most expensive fuel in step R2. Purchasing process follows removal process in step 5. The cheapest fuel is determined in step P2 and it is bought partially in step P4, if the required fuel amount is less than the amount of cheapest fuel in tank. Otherwise the cheapest fuel is bought completely in step P6 and the algorithm proceeds to the next cheapest fuel in step P2. In the last city of the route all fuel is removed as in step 8.

Running time. If the list of fuel amounts L is implemented as binary heaps, then inserting or deleting a pair preserving the heap structure can be done in $O(\log n)$ time. The running time of the algorithm is $O(n \log n)$, since each fuel is inserted and deleted once.

5.4.1 Definitions and observation

In this section, we give definitions of several concepts that will be used in the correctness proof of our greedy algorithm. All definitions in this section are based on two variables χ_{wt} and π_{wt} that are the units of fuel amount from city t removed and purchased in city w respectively. Table 5.2 gives the functions "*RemoveExcess2*" and "*Purchase2*" that enables us to find the values of the decision variables χ_{wt} and π_{wt} for $t, w \in S$. We note that the greedy algorithm with these function has $O(n^2)$ running time, but this version is only for our analysis. In the rest of this chapter, "fuel type" refers to the index of the city where the fuel comes from.

Definition 5.4.1. *(Refueling amount)*
The refueling amount in city w is given by $x_w = \sum_{z \geq w} \pi_{zw}$.

Definition 5.4.2. *(Fuel pass and fuel flow)*
In city w, the "fuel pass" is the set of fuel types that are brought from earlier cities and purchased in later cities. Fuel flow is the total amount of in the fuel pass. Fuel pass and fuel flow are denoted by Φ_w and ϕ_w respectively.

We have $\Phi_w = \{t \in S | t \leq w, \sum_{z \geq w} \pi_{zt} > 0\}$, and $\phi_w = \sum_{z \geq w} \sum_{t \in \Phi_w} \pi_{zt}$.

Definition 5.4.3. *(Fuel miss and fuel drop)*
In city w, the "fuel miss" is the set of fuel types that are brought from earlier cities and removed in later cities. Fuel drop is the total amount of in the fuel miss. Fuel miss and fuel drop are denoted by Θ_w and θ_w respectively.

We have $\Theta_w = \{t \in S | t \leq w, \sum_{z > w} \chi_{zt} > 0\}$, and $\theta_w = \sum_{z > w} \sum_{t \in \Theta_w} \chi_{zt}$.

Definition 5.4.4. *(Removal set)*
In city w, the removed fuel types are given by $\Lambda_w = \{t \in S | \chi_{wt} > 0\}$.

Table 5.2: Additional variables

Variables	
χ_{wt}	Fuel amount from city t removed in city w, $\chi_{wt} \geq 0$
π_{wt}	Fuel amount from city t purchased in city w, $\pi_{wt} \geq 0$

RemoveExcess2(L,T,w)	*Purchase2*(L,d,w)
S1: **while** $\sum_{\langle v,y_v\rangle \in L} y_v > T$ **do**	T1: **while** $d > 0$ **do**
S2: $t \leftarrow \text{Argmax}\{p_v \vert \langle v, y_v\rangle \in L, y_v > 0\}$;	T2: $t \leftarrow \text{Argmin}\{p_v \vert \langle v, y_v\rangle \in L, y_v > 0\}$;
S3: **if** $y_t > \sum_{\langle v,y_v\rangle \in L} y_v - T$ **do**	T3: **if** $y_t > d$ **do**
S4: $\chi_{wt} \leftarrow \sum_{\langle v,y_v\rangle \in L} y_v - T$;	T4: $\pi_{wt} \leftarrow d$;
S5: **else**	T5: **else**
S6: $\chi_{wt} \leftarrow y_t$;	T6: $\pi_{wt} \leftarrow y_t$;
S7: **end**	T7: **end**
S8: $y_t \leftarrow y_t - \chi_{wt}$;	T8: $y_t \leftarrow y_t - \pi_{wt}$, $d \leftarrow d - \pi_{wt}$;
S9: **end**	T9: **end**

Definition 5.4.5. *(Empty-tank and full-tank)*
In city w, empty-tank (full-tank) event occurs if and only if $\phi_w = d_w$ ($\phi_w = T_w$).

The sets $F = \{w \in S \vert \phi_w = T_w\}$, $E = \{w \in S \vert \phi_w = d_w\}$ denote the cities where full-tank and empty-tank events occur respectively.

Definition 5.4.6. *(Highest-consumed and lowest-removed)*
In a city, the most expensive (cheapest) fuel type in the fuel pass (fuel miss) is called "highest-consumed" ("lowest-removed").

The highest-consumed function $h : S \mapsto S$ is given by $h(w) = \text{Argmax}_{t\in\Phi_w}\{p_t\}$, and the lowest-removed function $l : S \mapsto S$ is given by $l(w) = \text{Argmin}_{t\in\Theta_w \cup \{h(w)\}}\{p_t\}$.

Definition 5.4.7. *(Dominant fuel type)*
The highest-consumed fuel type in city $w \in E \cup F$ is said to be dominant.

The set $D = \{w \in S \vert \exists y \in E \cup F \text{ such that } h(y) = w\}$ contains dominant fuel types. The event function $e : D \mapsto E \cup F$ returns the city of event and it is given by $e(w) = \{t \in E \cup F \vert h(t) = w\}$.

Definition 5.4.8. *(Region)*
A region is defined by $R_w = \{t \in S \vert \max\{0, e(\max_{v\in D}\{v < w\}) + 1\} \leq t \leq e(w)\}$ for dominant fuel type $w \in D$.

Note that $R_w \cap (E \cup F) = \{e(w)\}$ and $D \cap R_w = \{w\}$ for all $w \in D$.

Definition 5.4.9. *(Regional price)*
The function $r : S \mapsto Z_+$ returns the regional price and we have $r(v) = p_w$ for $v \in R_w$.

Definition 5.4.10. *(Regional price decrease, regional price increase)*
The function $r^-(w) = \max\{0, r(w) - r(w+1)\}$ ($r^+(w) = \max\{0, r(w+1) - r(w)\}$) is called regional price decrease (increase) function.

Observation 5.4.1. $\Lambda_w \neq \emptyset \Rightarrow \phi_w + \theta_w = T_w$

Observation 5.4.2. $\Lambda_w \neq \emptyset$ *and* $\Theta_w = \emptyset \Rightarrow w \in F$.

Observation 5.4.3. $p_v \leq p_t \leq p_z, \quad \forall (v,t,z) \in \Phi_w \times \Theta_w \times \Lambda_w$.

Observation 5.4.4. $p_{l(w)} \leq p_t, \quad \forall t \in \Theta_w \cup \Lambda_w \cup \{h(w)\}$.

Observation 5.4.5. $d_w < \phi_w \Rightarrow h(w) \in \Phi_{w+1}$.

5.4.2 Correctness proof using duality

In this section, let $s,t,v,w,z \in S$ and we make the following two assumptions to avoid the degeneracy.

Assumption 5.4.6. $E \cap F = \emptyset$.

Assumption 5.4.7. $x_w < U_w$ *for all* $w \in D$.

Lemma 5.4.8. *Let* $v, v+1 \in R_w \setminus e(w)$. *If* $\Theta_v \neq \emptyset$, *then* $p_{l(v+1)} \leq p_{l(v)}$.

Proof. First of all, we claim that $\Theta_{v+1} \neq \emptyset$, otherwise we would have $\Theta_v \subseteq \Lambda_{v+1}$ and by Observation 5.4.2 we get a contradiction: $v+1 \in F$. We know that $\Theta_v \subseteq \Theta_{v+1} \cup \Lambda_{v+1}$ and by Observation 5.4.4 we have $p_{l(v+1)} \leq p_t$ for all $t \in \Theta_{v+1} \cup \Lambda_{v+1}$. □

Corollary 5.4.9. *Let* $v \in R_w \setminus e(w)$ *such that* $\Theta_v \neq \emptyset$. *One has*

$$p_w \leq p_{l(e(w)-1)} \leq \cdots \leq p_{l(v+1)} \leq p_{l(v)} \tag{5.13}$$

Proof. Applying Lemma 5.4.8 repeatedly for $v, v+1, \ldots, e(w)-1$ shows the correctness of inequalities except the left-most one. To see the correctness of the left-most inequality, we observe that $\Theta_{e(w)-1} \subseteq \Theta_{e(w)} \cup \Lambda_{e(w)}$, $l(e(w)-1) \in \Theta_{e(w)} \cup \Lambda_{e(w)}$, and finally $p_w = p_{h(e(w))} \leq p_{l(e(w)-1)}$. □

Lemma 5.4.10. *Let* $v, v+1 \in R_w$. *One has* $p_{h(v)} \leq p_{h(v+1)}$.

Proof. Note that $v < e(w)$, hence $v \notin E$ which implies $d_v < \theta_v$. By Observation 5.4.5, we have $h(v) \in \Phi_{v+1}$. □

Corollary 5.4.11. *Let* $w \in D$ *and let* s *be the first city of* R_w. *One has*

$$p_{h(s)} \leq p_{h(s+1)} \leq \cdots \leq p_{h(e(w))} = p_w \tag{5.14}$$

Lemma 5.4.12. *Let* $v \neq w \in R_w$. *One has*

$$x_v = \begin{cases} 0 & \text{if } p_w < p_v \\ U_v & \text{if } p_v < p_w \end{cases} \tag{5.15}$$

Proof. We distinguish the following cases

Case $p_w < p_v$: Suppose that $0 < x_v$, so $v \in \Phi_v$. By Corollary 5.4.11, we get the contradiction: $p_v \leq p_{h(v)} \leq p_w$.

Case $p_v < p_w$: Suppose that $x_v < U_v$, so $v \in \Theta_v$. By Corollary 5.4.9, we get the contradiction: $p_w \leq p_{l(v)}$. □

Proposition 5.4.13. Empty-tank *event results in regional price decrease.*

Proof. Let $w \in E$ and $w+1 \in R_z$, so $r(w) = p_{h(w)}$ and $r(w+1) = p_z$. Considering $h(w) \in \Theta_{h(w)}$ and Corollary 5.4.9, we see $h(w) = l(w) = l(w-1)$. Since $h(w) = l(w-1) \in \Lambda_w \cup \Theta_w$, we have $\Theta_w \neq \emptyset$, otherwise we would have $w \in F$ by Observation 5.4.2. Then it must be true that $h(w) \in \Theta_w$, since $(\Lambda_w \cup \Theta_w) \cap \Phi_w = \{h(w)\}$. Since $h(w) \in \Theta_w$, we have $h(w) \in \Lambda_{w+1} \cup \Theta_{w+1}$ which implies $p_{l(w+1)} \leq p_{h(w)}$. By Corollary 5.4.9, we get $p_z \leq p_{l(w+1)} \leq p_{h(w)}$. So $p_{h(w)} = r(w) > r(w+1) = p_z$. □

Proposition 5.4.14. Full-tank *event results in regional price increase.*

Proof. Let $w \in F$ and let $w+1 \in R_z$, so the regional prices are $r(w) = p_{h(w)}$ and $r(w+1) = p_z$. By Assumption 5.4.6, $d_w < \phi_w$, hence $h(w) \in \Phi_{w+1}$ which implies $p_{h(w)} \leq p_{w+1}$. By Corollary 5.4.11, we see that $p_{w+1} \leq p_z$. Finally, we obtain $p_{h(w)} = r(w) < r(w+1) = p_z$. □

Fuel flow equation

Now let $t, w \in D$ such that $w = \min_{z \in D}\{t < z\}$. Let us rewrite fuel flow relation between cities $e(t)$ and $e(w)$ considering Lemma 5.4.12 as follows

$$\phi_{e(w)} = \phi_{e(t)} + \sum_{v \in R_w} x_v - \sum_{z=e(t)}^{e(w)-1} d_z = \phi_{e(t)} + \sum_{v \in R_w : p_v < p_w} U_v + x_w - \sum_{z=e(t)}^{e(w)-1} d_z \tag{5.16}$$

Note that in the above equation $e(w), e(t) \in E \cup F$, so $\phi_{e(w)}$ and $\phi_{e(t)}$ are constants, being equal either to the distance or to the tank capacity. The only variable in the equation is x_w that is the refueling amount of the dominant fuel type w.

Dual of the LP formulation

In the dual of the LP model, the variables α_w, β_w, and γ_w correspond constraints (5.1), (5.2) and (5.3) respectively. The dual problem is as below.

$$\text{maximize} \qquad \sum_{w \in S} \left(\sum_{t=1}^{w} d_t \alpha_w - \left(\sum_{t=1}^{w-1} d_t + T_w \right) \beta_w - U_w \gamma_w \right) \tag{5.17}$$

subject to

$$\sum_{t=w}^{n-1} (\alpha_t - \beta_t) - \gamma_w \leq p_w, \quad \forall w \in S \tag{5.18}$$

$$\alpha_w, \beta_w, \gamma_w \geq 0, \quad \forall w \in S \tag{5.19}$$

Complementary slackness

Let $x_w^*, \alpha_w^*, \beta_w^*$, and γ_w^* for all $w \in S$ be the optimal solution values of primal and dual problems.

$$\left(\sum_{t=1}^{w} x_t^* - \sum_{t=1}^{w} d_t\right) \alpha_w^* = 0, \quad \forall w \in S \tag{5.20}$$

$$\left(\sum_{t=1}^{w} x_t^* - \sum_{t=1}^{w-1} d_t - T_w\right) \beta_w^* = 0, \quad \forall w \in S \tag{5.21}$$

$$(x_w^* - U_w)\, \gamma_w^* = 0, \quad \forall w \in S \tag{5.22}$$

$$x_w^* \left(\sum_{t=w}^{n-1} (\alpha_t^* - \beta_t^*) - \gamma_w^* - p_w\right) = 0, \quad \forall w \in S \tag{5.23}$$

According to the complementary slackness condition, in optimal solutions, the dual variable values $\alpha_w^* > 0$, $\beta_{w'}^* > 0$, and $\delta_z^* > 0$ correspond to $w \in E$, $w' \in F$, and city z such that $p_z < r(z)$ by Lemma 5.4.12.

In mathematical programming, dual variables are interpreted as derivatives of the optimal value of the objective function with respect to the elements of the right-hand-side. In the following Lemmas, we will prove that the regional price change functions correspond to dual variables and vice versa.

Lemma 5.4.15. *One has* $\alpha_w^* := r^-(w)$.

Proof. By complementary slackness condition, $\alpha_w^* > 0$ for city $w \in E$ and otherwise $\alpha_w^* = 0$. In Proposition 5.4.13 we showed that in city $w \in E$ regional price decreases, so $r^-(w) > 0$. If city $w \notin E \cup F$ or $w \in R_z \setminus e(z)$ for any $z \in D$, then there is no change in regional price, hence $r^-(w) = 0$. In city $w \in F$, regional price increases and $r^-(w) = 0$. Having showed that they are positive only in empty-tank events, let us show that they have the same value. We will show this by using the fuel flow equation in (5.16). Let us consider an infinitesimal change $0 < \varepsilon$ in the right hand side of the constraint 5.1. Note that ε is strictly smaller than any value in the solution. Let the *empty-tank* event occur in city w such that $\alpha_w^* = \frac{\partial Cost}{\partial Distance}$ where $Cost = \sum_{t \in S} p_t x_t$ and $Distance = \sum_{t=1}^{w} d_t$. Let fuel type $v = \max_{t \in D}\{t < h(w)\}$ be the previous dominant fuel type in the solution and without loss of let $e(v) \in F$. Then the fuel flow equation becomes

$$\sum_{t=e(v)}^{w} d_t = T_v + \sum_{t \in R_{h(w)} : p_t < p_{h(w)}} U_t + x_{h(w)} \tag{5.24}$$

Now we change the distance d_w to $d_w + \varepsilon$ such that $\partial Distance = \varepsilon$ and other parameters stay the same. The change ε is so small such that all fuel passes, fuel misses, removal sets stay the same as well. Therefore fuel types that are not used are again not use and the ones used completely are again used completely. If we examine the flow equation in (5.24), wee see that the refueling amount of dominant fuel type $h(w)$ is the only variable in the equation. By the fuel flow equation, the ε distance

change in the left side causes ε change of refueling amount of dominant fuel type $h(w)$ in the right side, so it becomes $x_{h(w)} + \varepsilon$.

Note that the distances to other cities must stay the same to have the same right hand sides of other constraints. Therefore, d_{w+1} must become $d_{w+1} - \varepsilon$. Let $z = \max_{t \in D}\{h(w) < t\}$ and without loss of generality let $e(z) \in E$. The corresponding fuel flow equation is given below

$$\sum_{t=w}^{e(z)} d_t = d_w + \sum_{t \in R_z : p_t < p_z} U_t + x_z \tag{5.25}$$

We see that the change from d_{w+1} to $d_{w+1} - \varepsilon$ causes a change in the refueling amount of fuel type z from x_z to $x_z - \varepsilon$. Consequently, the change in the refueling cost is $\partial Cost = \varepsilon(p_{h(w)} - p_z)$, hence we have $\alpha_w^* = \frac{\varepsilon(p_{h(w)} - p_z)}{\varepsilon} = p_{h(w)} - p_z$ which is equal to $r(w) - r(w+1)$. By Proposition 5.4.13, this amount is positive, so $r^-(w) = p_{h(w)} - p_z = \alpha_w^*$.

□

Lemma 5.4.16. *One has* $\beta_w^* := r^+(w)$.

Proof. By complementary slackness condition, $\beta_w^* > 0$ for city $y \in F$ and otherwise $\beta_w^* = 0$. In Proposition 5.4.14 we showed that in city $w \in F$ the regional price increases, so $r^+(w) > 0$. If city $w \notin E \cup F$ or $w \in R_z \setminus e(z)$ for any $z \in D$, then there is no change in regional price, hence $r^+(w) = 0$. In city $w \in E$, regional price decreases and $r^+(w) = 0$. Having showed that they are positive only in full-tank events, let us show as in previous proof that they have the same value.

The fuel flow equation is used again. Now the change will be in the tank capacity T_w by ε where $w \in F$. In the fuel flow equation, the change in tank capacity causes an increase in refueling amount of fuel type $h(w)$ by ε and it becomes $x_{h(w)} + \varepsilon$. The change in tank capacity from T_w to $T_w + \varepsilon$ results in a decrease of the successive dominant fuel z, being $x_z - \varepsilon$. Finally, we obtain $\beta_w^* = -\frac{\partial Cost}{\partial TankCapacity} = -\frac{\varepsilon(p_{h(w)} - p_z)}{\varepsilon} = p_z - p_{h(w)}$. By Proposition 5.4.14, this amount is positive, so $r^+(w) = p_z - p_{h(w)} = \beta_w^*$.

□

Lemma 5.4.17. *One has* $\gamma_z^* := \max\{0, r(z) - p_z\}$.

Proof. By complementary slackness condition, $\gamma_z^* > 0$ for city z such that $x_z = U_z$. In Lemma 5.4.12, we showed that this only happens if $p_z < r(z)$. Then $\max\{0, r(z) - p_z\} = r(z) - p_z$ and for all other cities $\max\{0, r(z) - p_z\} = 0$, so is γ_z^*.

Let us show that the value of γ_z^* is equal to the value of $\max\{0, r(z) - p_z\}$ whenever both are positive. To do this, we change U_z to $U_z + \varepsilon$ such that ε is smaller than any value in the solution. The dual variable γ_z^* is given by $-\frac{\partial Cost}{\partial AvailableFuel}$. Without loss of generality, we assume that $z \in R_w$ and $e(w) \in E$ and $v = \max_{t \in D}\{t < w\}$ wit $e(v) \in F$. So the fuel flow equation becomes as follows

$$\sum_{t=e(v)}^{e(w)} d_t = T_v + \sum_{t \in R_z : p_t < p_w} U_t + x_w \tag{5.26}$$

We see that the change from U_z to $U_z + \varepsilon$ causes a change in the refueling amount of fuel type w from x_w to $x_w - \varepsilon$ and a change in refueling amount of fuel z from U_z to $U_z + \varepsilon$. Consequently, the change in the refueling cost is $\partial Cost = \varepsilon(p_z - p_w)$, hence we have $\gamma_z^* = -\frac{\varepsilon(p_z - p_w)}{\varepsilon} = p_w - p_z$ which is equal to $r(z) - p_z$. This amount is positive, so $\max\{0, r(z) - p_z\} = r(z) - p_z = \gamma_z^*$.

□

Dual interpretation

In the dual problem, the route is partitioned into regions. A region starts and ends with an event that is either an empty-tank or a full-tank event. Every region has a regional price that is the price of the dominant fuel type. As a convention, we define a special region that includes only the city n and has regional price 0. This convention applies to all solutions. The empty-tank event in city $n-1$ in all optimal solutions makes this convention meaningful.

In this study, we provide closed form interpretations of the variables in the dual of LP model (5.1)-(5.4). Below we summarize these interpretations

$$\alpha_w^* = r^-(w), \beta_w^* = r^+(w), \gamma_w^* = \max\{0, r(w) - p_w\}, \quad \forall w \in S \tag{5.27}$$

By aforementioned convention, $r(n) = 0$ and the regional price in city w is found by backtracking the changes as below

$$r(w) = r(n) - \sum_{t=w}^{n-1}(r^+(t) - r^-(t)) = \sum_{t=w}^{n-1}(\alpha_t^* - \beta_t^*), \quad \forall w \in S \tag{5.28}$$

Theorem 5.4.18. Algorithm 1 *solves the vehicle refueling problem optimally in* $O(n \log n)$ *time.*

Proof. We provide the argument that the running time of the greedy algorithm is $O(n \log n)$ in Section 5.4. Next, let us show that it constructs optimal solutions. Note that for any $w \in S$ we have $\gamma_w^* \geq r(w) - p_w$ by the interpretation in (5.27). We substitute the right side of equation (5.28), the inequality becomes $\gamma_w^* \geq \sum_{t=w}^{n-1}(\alpha_t^* - \beta_t^*) - p_w$ which is the dual constraint (5.18), hence the dual solution is dual feasible. The correctness of the values of dual variables α_w^* and β_w^*, and γ_w^* for all w in S is shown in Lemmas 5.4.15, 5.4.16, and 5.4.17 by using the complementary slackness condition. □

Remark 5.4.19. Algorithm 1 *solves the vehicle refueling problem studied by Lin et al. (2007) in* $O(n)$ *time using doubly linked list.*

5.4.3 Correctness proof using network flow

In this section we prove that the greedy algorithm produces optimal solutions by showing that there is no negative cost directed cycle on the network in Figure 5.1 if the refueling solution is converted into a flow on it. To do this, let us first explain

how removals and purchases of the greedy algorithm turn to a feasible flow on the network.

Removal and purchase decisions are made on the nodes v_w and $v_{w,w+1}$ respectively for all $w \in S \setminus \{w\}$. In fact, the removals are done for satisfying the upper bound constraints on the arcs $(v_w, v_{w,w+1})$, since they are equal to tank capacities. Purchases are done to satisfy lower and upper bounds of arcs $(v_{w,w+1}, t)$, since they are equal to the distances to next cities. So the greedy algorithm starts from nodes v_1 and v_{12} and constructs a feasible flow gradually. Let us define the flow values on the arcs after the greedy algorithm finishes removals and purchases in cities. They are given by

$$f_{(s,v_w)} = \sum_{w \leq y} \pi_{wy} = x_w, \quad f_{(v_{w,w+1},t)} = \sum_{t \leq w} \pi_{wt} = d_w, \quad \forall w \in S \setminus \{n\} \tag{5.29}$$

$$f_{(v_w,v_{w,w+1})} = \sum_{y \geq w} \sum_{t \in \phi_w} \pi_{yt} = \phi_w, \quad \forall w \in S \setminus \{n\} \tag{5.30}$$

where Φ_w is the fuel pass in city w (see Definition 5.4.2). For all $w \in S \setminus n$, $f_{(v_{w-1,w},v_w)} = \phi_{w-1} - d_{w-1}$ and these flow values depend on the defined flow values above. It is easy to see that the purchase π_{wt} is an augmenting flow between trough on the directed path $\{s, v_t, v_{t,t+1}, \ldots, v_w, v_{w,w+1}, t\}$.

Now we define the *residual network*. There are two arcs originating from arc (i, j) in the residual network: (i, j) and (j, i). The residual capacity on arc (i, j) is given by $r_{(i,j)} = u_{(i,j)} - f_{(i,j)}$ with cost $c_{(i,j)}$. The capacity of arc (j, i) is the flow on arc $f_{(i,j)}$ and the cost on it is $-c_{(i,j)}$. A directed cycle on a network is a set of arcs in which every node is entered and left equal times. For example, the set $\{(i, j), (j, k), (k, i)\}$ is directed path of three arcs. In network flow theory, one interpretation of optimality condition for a flow to have minimum cost states that there is no negative cost directed cycle in the residual network.

Theorem 5.4.20. *(Chapter 9, Ahuja et al. (1993)) A feasible flow f is an optimal solution of the minimum cost flow problem if and only if it satisfies the negative cost cycle optimality conditions: namely, the residual network contains no negative (directed) cycle.*

Let us now consider the residual network of the flow that is found by the greedy algorithm. Note that we have $f_{(v_{w,w+1},t)} = d_w$, hence the residual capacity of arcs $(v_{w,w+1}, t)$ are zero for all $w \in S \setminus \{n\}$. This implies that the sink t cannot be in any direct cycle, since there is no residual arc capacity to reach it. This fact fortunately reduces the number of cases to consider, and the only cycles in the residual network in Figure 5.1 for the flow found by greedy algorithm is given by $\{(s, v_w), (v_w, v_{w,w+1}), (v_{w,w+1}, v_{w+1}), \ldots, (v_y, s)\}$ between cities w and y. Having figured out the potential cycle type in the residual network, we state the following Lemma.

Lemma 5.4.21. *There is no negative cost cycle in the residual network of the flow defined in (5.29) and in (5.30).*

Proof. The proof is by contradiction. Suppose that there is a negative cost cycle $\{(s, v_w), (v_w, v_{w,w+1}), (v_{w,w+1}, v_{w+1}), \ldots, (v_y, s)\}$ in the residual network such that $w < y$. This negative cost cycle tells us directly $x_w < U_w$ and $0 < x_y$ due to the arc capacities in the residual network and $p_w < p_y$ due to the negativity of the cycle cost. Moreover, the feasibility of the cycle requires that the arcs $(v_t, v_{t,t+1})$, $(v_{t,t+1}, v_{t+1})$ have some residual capacity where $w \leq t < y$, hence, by equation (5.30), we see that $\phi_t < T_t$ for $w \leq t < y$. Moreover, we must have $0 < f_{(v_{t,t+1}, v_{t+1})}$ for all $w \leq t < y$.

By Lemma 5.4.12, we see that fuel type w is dominant (see Definition 5.4.7). Here we distinguish two cases:

Case $y \in R_w$: Firstly, we see that $0 < x_y$ implies that $p_y \leq p_{h(y)}$. By Corollary 5.4.11, we have $p_{h(y)} \leq p_w$ in the region R_w. Combining these two inequalities results in $p_y \leq p_w$ which contradicts the non-negativity of the directed cycle.

Case $y \notin R_w$: In this case we have $w \leq e(w) < y$ where $e(w)$ is either empty-tank or full-tank event of the dominant fuel type w (see Definition 5.4.5). If a full-tank event occurs in city $e(w)$ such that $e(w) \in F$, this implies that $\sum_{t=1}^{e(w)} x_t - \sum_{t=1}^{e(w)-1} d_t = \phi_{e(w)} = T_{e(w)}$. The fuel amount on arc $(v_{e(w)}, v_{e(w),e(w)+1})$ is equal to $T_{e(w)}$ by (5.30), and it is also equal to the upper bound $u_{(v_{e(w)}, v_{e(w),e(w)+1})}$. This contradicts the feasibility of the negative cost cycle. If an empty-tank event occurs in city $e(w)$ such that $e(w) \in E$, we have $\sum_{t=1}^{e(w)} x_t - \sum_{t=1}^{e(w)} d_t = 0 = f_{(v_{e(w),e(w)+1}, v_{e(w)+1})}$. This contradicts the feasibility of the negative cost cycle that is mentioned above.

□

Having showed that there is no negative cost cycle in the residual network of the flow found by the greedy algorithm, we state the final result as

Theorem 5.4.22. Algorithm 1 *solves the vehicle refueling problem optimally in* $O(n \log n)$ *time.*

Proof. We provide the argument that the running time of the greedy algorithm is $O(n \log n)$ in Section 5.4. The optimality of the solutions of our greedy algorithm follows Lemma 5.4.21 and Theorem 5.4.20. □

5.4.4 Correctness proof using convexity

In this section we show that a solution found by our greedy algorithm has the minimum cost in its neighborhood. The cost function of the problem is linear, hence convex, and we know already that if a point is a local minimum in a convex set, then it is also global minimum. The optimality of the solutions found by greedy algorithm will follow this argument. Now, we explain how a neighborhood of a solution can be obtained.

On the solutions of the greedy algorithm. By Lemma 5.4.12, in any non-degenerate solution of the greedy algorithm, we have three different groups of fuel types: those *non-purchased*, those *all-purchased*, and those *partially-purchased*. The first group includes the fuel types with price higher than the regional price. The second group includes the fuel types with price less than the regional price, and the third group is merely the set of dominant fuel types.

Neighborhood of a solution. We will define the neighborhood of a solution $x \in R^n$ by considering a sufficiently small change in the refueling amount of a fuel type. We denote a sufficiently small increase in an amount by $\varepsilon \uparrow$, and decrease by $\varepsilon \downarrow$. A change in a refueling amount must be compensated by changing another fuel type to have the same total refueling amount (due to fixed route length $\sum_{t=1}^{n-1} d_t$). We define the neighborhood of the solution x as below.

$$N_1(x) = \left\{ x^\circ \in R^n | \exists w, y \in S : \begin{array}{lll} x_w^\circ = & x_w - \varepsilon & \\ x_y^\circ = & x_y + \varepsilon & \\ x_t^\circ = & x_t, & \forall t \in S \setminus \{w, y\} \end{array} \right\} \tag{5.31}$$

$$N_2(x) = \left\{ x^\circ \in R^n | \exists w, y \in S : \begin{array}{lll} x_w^\circ = & x_w + \varepsilon & \\ x_y^\circ = & x_y - \varepsilon & \\ x_t^\circ = & x_t, & \forall t \in S \setminus \{w, y\} \end{array} \right\} \tag{5.32}$$

The neighborhood of the solution x is merely $N(x) = N_1(x) \cup N_2(x)$. Here, the important point is to see how other fuel types are effected, if a particular fuel type is either increased or decreased in sufficiently small amount. We use the *flow equation* to figure out this point.

Now we distinguish the following fuel types: $s, y \in D$ with $s = \max_{v \in D}\{v < y\}$, $v, w \in S$ with $x_v = 0$, $x_w = U_w$, and $v, w \in R_y$ as non-purchased, all-purchased, and partially-purchased fuel types. Having distinguished fuel types, we rewrite the flow equation

$$\phi_{e(s)} - \phi_{e(y)} = \sum_{t=e(s)}^{e(y)-1} d_t + \sum_{t \in R_y : t \neq w, p_t < p_y} U_t + U_w + x_y \tag{5.33}$$

Note that in above equation $\phi_{e(s)}$ is equal to either d_s or T_s, and similarly, $\phi_{e(y)}$ is equal to d_y or T_y. So both are constants. In the proofs of the following Lemmas, we will use fuel types s, v, w, and y without loss of generality.

Lemma 5.4.23. *$\varepsilon \uparrow$ of a "non-purchased" fuel leads to $\varepsilon \downarrow$ of a "dominant" fuel.*

Proof. The refueling amount $x_v + \varepsilon$ (an increase in non-purchased) leads to either $x_w - \varepsilon$ (a decrease in all-purchased fuel) or $x_y - \varepsilon$ (a decrease in dominant fuel) by the flow equation in (5.4.23). Note that the former would cause a higher cost drop than the latter due to $p_w < p_y$. Therefore, a decrease in dominant fuel occurs.

□

Lemma 5.4.24. *$\varepsilon \downarrow$ of a "all-purchased" fuel leads to $\varepsilon \uparrow$ of a "dominant" fuel.*

Lemma 5.4.24 can be proven by using symmetric arguments in the proof of Lemma 5.4.23, and we do not give its proof for the sake of brevity.

Lemma 5.4.25. *$\varepsilon \downarrow$ of a "dominant" fuel with empty-tank event leads to $\varepsilon \uparrow$ of a fuel with higher price.*

Proof. We have $e(y) \in E$ in solution x. In order to reach city $e(y)+1$, we must have $x_t + \varepsilon$ where $t \neq y$ and $t \leq e(y)$. Now, suppose that $p_t < p_y$. Then it is clear that $t \leq e(s)$, since all cheaper fuel types than p_y are all-purchased in R_y. Now, we have two cases: $e(s) \in F$ or $e(s) \in E$. The former is impossible due to the tank capacity constraint in $e(s)$. In the latter, we have $p_s > p_y$ and all non-purchased fuel types have also higher price than p_s. Consequently, we must have $p_t > p_y$.

□

Lemma 5.4.26. *$\varepsilon \uparrow$ of a "dominant" fuel with empty-tank event leads to $\varepsilon \downarrow$ of a fuel with lower price.*

Proof. We again have $e(y) \in E$ in solution x. Now we must have a refueling amount $x_t - \varepsilon$ for $t \neq y$. Suppose that $p_t > p_y$. We distinguish two following cases: either $t > e(y)$ or $t < e(y)$. The former is impossible, since all purchased fuel types in the next region have lower prices by Proposition 5.4.13. In the latter, we must have $t \in R_y$ and all such fuel types must have lower price than p_y, so we get the contradiction. □

Lemma 5.4.27. *$\varepsilon \downarrow$ ($\varepsilon \uparrow$) of a "dominant" fuel with full-tank event leads to $\varepsilon \uparrow$ ($\varepsilon \downarrow$) of a fuel with higher (lower) price.*

Lemma 5.4.27 can be proven by using similar arguments in the proofs of Lemma 5.4.25 and Lemma 5.4.26, hence we do not give its proof for the sake of brevity.

Proposition 5.4.28. *One has $f(x) \leq f(x^\circ)$ for all $x^\circ \in N(x)$.*

Proof. It is easy to see that for all cases in Lemmas 5.4.23-5.4.27 the cost of the solution x° is higher. □

Theorem 5.4.29. Algorithm 1 *solves the vehicle refueling problem optimally in $O(n \log n)$ time.*

Proof. We provide the argument that the running time of the greedy algorithm is $O(n \log n)$ in Section 5.4. The optimality of the solutions of our greedy algorithm follows Proposition 5.4.28, since a local minimum implies global minimum in case of convexity. □

5.5 Concluding remarks

In this chapter, we present an intuitive greedy algorithm for a vehicle refueling problem. The problem under consideration is a slightly generalized version of the ones studied in the literature. We see that the equivalent problems to the vehicle refueling problem in lot sizing and in network flow are also solved in $O(n \log n)$ time. The question coming to mind is that "Is $O(n \log n)$ the shortest possible time for this problem? Can this problem be solved in linear time by any other approach?".

Perspectives

Multi-skill workforce scheduling is a highly complex problem in combinatorial optimization. Solving this problem to optimality is usually not an option to consider. In the literature, we encounter a number of complicated local search techniques to tackle complex scheduling problems. We observe that these techniques are successful in finding good-quality solutions to some extent, whereas their convergence speed seems to be sensitive to implementation and they are in general not robust to varying instance types. This point has also been observed for the particular multi-skill workforce scheduling problem of France Telecom in the 2007 ROADEF Challenge.

In this thesis, we designed a flexible matching model to tackle the scheduling problem of France Telecom. This model is basically a formulation of the matching problem on bipartite graphs. Firstly, this approach enabled us to have a global way of assigning technicians to tasks. Secondly, any well-thought improvement (to increase the solution quality whenever different instance types are encountered) ended up with higher solution quality in all instances. In our opinion, this thesis may inspire developments of combinatorial approaches to tackle other complex scheduling problems. For example, other problems in multi-skill workforce scheduling may be tackled by using our models with slight modifications.

In Chapter 2, the relaxation of the lower bound model (LBM) may provide us important information for the skill management. This MIP model determines the completion time of every priority class and makes the outsourcing decisions. If there is "rare expertise" within the technician group, usually hard tasks are outsourced. For this reason, rare expertise makes the project completion times dependent on the external companies and it leads to high outsourcing expenses. Skill training seems promising in preventing these undesired cases. If it is possible to train the technicians with a cost that is covered by near-future outsourcing costs, the dependency to external companies will decrease without using an extra budget. The LBM may be modified in order to design a tactical-level decision tool to manage skills with hierarchical levels. Moreover, the decisions made at tactical level should be revised by using the feedback from operation level.

In *stable workforce assignment problem*, we considered two different sets of players; technicians and jobs, and we defined the blocking case in an assignment based on the Marriage problem. It is also natural that technicians may rank each other, since they work in teams. This point reminds us the stable roommates problem, which is slightly different than the Marriage problem. Actually, if both technician-job preferences and technician-technician preferences are considered, the stable assignment problem turns

to a more complicated problem then each of the Marriage problem and the Roommate problem. If all these preferences are considered, defining the problem and developing an approach may be further directions in this topic. In this problem, the graph-theoretic approaches seem more promising than the MIP-based approaches.

Equivalent problems to the *vehicle refueling problem* in Chapter 5 is also approached by other researchers. The best running time achieved so far is $O(n \log n)$. The question coming into mind is that "Is $O(n \log n)$ the shortest possible running time for this problem?". This question can be answered in two ways; (1) by reducing a problem with a running time $O(n \log n)$ to the vehicle refueling problem for which $O(n \log n)$ is shown the shortest possible running time, or (2) by developing another approach with the hope of obtaining a shorter running time.

Bibliography

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. (1993). *Network flows: Theory, algorithms and applications*. New Jersey: Prentice-Hall, Englewood Cliffs.

Ahuja, R.K., Hochbaum, D.S. (2008). Solving linear cost dynamic lot-Sizing problems in $O(n \log n)$ time. *Operations Research 56*, 255-261.

Atallah, D.Z. Chen, Lee, D.T. (1995). An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications. *Algorithmica 14*, 429-441.

Avramidis, N.A., Chan, W., Gendreau, M., L'Ecuyer, P., Pisacane, O. (2010). Optimizing daily agent scheduling in a multi-skill call center. *European Journal of Operational Research 200*, 822-832.

Baïou, M., Balinski, M. (2000a). The stable admissions polytope. *Mathematical Programming 87*, 427-439.

Baïou, M., Balinski, M. (2000b). Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics 101*, 1-12.

Baïou, M., Balinski, M. (2002). The stable allocation (or ordinal transportation) problem. *Mathematics of Operations Research 27*, 485-503.

Ballou, D., Tayi, G. (1996). A decision aid for the selection and scheduling of software maintenance projects. *IEEE Transactions on Systems Man and Cybernetics Part A Systems and Humans 26*, 203-212.

Baker, K.R. (1974). *Introduction to sequencing and scheduling.* John Wiley & Sons.

Bellenguez, M.O. (2006). *Methods to solve multi-skill project scheduling problem.* Ph.D. Thesis, Francois Rabelais University, Tours, France.

Bellenguez, M.O., Neron. E. (2004). Lower Bounds for the multi-skill project scheduling problem with hierarchical levels of skills. *Lecture Notes in Computer Science: Practice and Theory of Automated Timetabling V*, Springer Berlin / Heidelberg, 229–243.

Bellenguez, M.O., Neron. E. (2007). A Branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO Operations Research 41*, 155-170.

Bertsekas, D.P. (2003). *Nonlinear Programming (2nd edition).* Athena Scientific, Belmont, Massachusetts.

Bertsimas, D., Tsitsiklis, J.N. (1997). *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts.

Brucker, P., Drexl, A., Möhring R., Neumann K., Pesch E. (1999). Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research 22*, 3-41.

Brucker, P., Knust, S. (2000). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research 127*, 355-362.

Brucker, P. (2007). *Scheduling algorithms (5th edition).* Berlin: Springer-Verlag.

Brucker, P., Knust, S. (2009). *Complexity results for scheduling problems.* http://www.informatik.uni-osnabrueck.de/knust/class/.

Cai, X., and Li, K.N. (2000). A genetic algorithm for scheduling staff of mixed skills under multi-criteria. *European Journal of Operational Research 125*, 359-369.

Caro, Y., Sebő, A., Tarsi, M. (1996). Recognizing greedy structures. *Journal of Algorithms 20*, 137-156.

Cook, S. (1971). The complexity of theorem proving procedures. *Proceedings Third Annual ACM Symposium on Theory of Computing* , 151-158.

Cordeau, J.F., Laporte, G., Pasin F., Ropke, S. (2010). Scheduling technicians and tasks in a telecommunication company. *Journal of Scheduling 13*, 393-409.

Cormen,T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2009). Introduction to algorithms (3rd edition). The MIT Press.

De Reyck, B., and Herroelen, W.S. (1999). The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research 119*, 538-556.

Drezet, L.E., and Billaut, J.C. (2008). A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics 112*, 217-225.

Dutot, P., Laugier, A., Bustos, A. (2006). Technicians and interventions scheduling for telecommunications. France Telecom R &D.

Edmonds, J. (1962). Covers and packings in a family of sets. *Bulletin of the American Mathematical Society 68*, 494-499.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics 17*, 449-467.

Estellon, B., Gardi, F., Nouioua, K. (2009). High-Performance local search for task scheduling with human resource allocation. *Lecture Notes In Computer Science. 5752*, 1-15.

Firat, M., Hurkens, C.A.J. (2011a). An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling.* , DOI: 10.1007/s10951-011-0245-x.

Firat, M., Woeginger, G.J. (2011b). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters 39*, 32–35.

Firat, M., Hurkens, C.A.J., Laugier, A. (2011c). Stable multi-skill workforce assignments. *Annals of OR.* , revised submission.

Fleiner, T., Irving, R.W., Manlove, D.F. (2007). Efficient algorithms for generalized stable marriage and roommates problems. *Theoretical Computer Science 381*, 162-176.

Fredman, M.L. and Tarjan, R.E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM 34*, 596-615.

Gale, D. and Shapley, L.S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly 69*, 9-15.

Gale, D. and Sotomayor, M. (1985). Some remarks on the stable matching problem. *Discrete Applied Mathematics 11*, 223-232.

Gantt, Henry L. (1903). A graphical daily balance in manufacture. *Transactions of the American Society of Mechanical Engineers 24*, 1322-1336.

Garey, M.R., Johnson, D.S. (1979). *Computers and intractability - a guide to NP-completeness*. San Fransisco: W.H. Freeman and Company.

Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T. (2010). Local search for stable marriage problem. *Proceedings of COMSOC 2010* , Düsseldorf, Germany.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287–326.

Gutjahr, W.J., Katzensteiner, S., Reiter, P., Stummer, C., Denk, M. (2008). Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research 16*, 281–306.

Hartmann, S., Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *Central European Journal of Operations Research 207*, 1–14.

Heilmann, R., Schwindt, C. (1997). Lower bounds for RCPSP. *Technical report WIOR-511, Universitaet Karlsruhe*, Germany.

Heimerl, C., Kolisch, R. (2010). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum 32*, 343–368.

Hochbaum, D.S., Shmoys, D.B. (1988). A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal of Computing 17*, 539-551.

Hoogeveen, J.A., Schuurman, P., Woeginger, G.J. (2001). Non-approximability results for scheduling problems with minsum criteria. *INFORMS Journal on Computing 13*, 157–168.

Hunsaker, B., Savelsbergh, M. (2002). Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters 30*, 169–173.

Hurkens, C.A.J. (2009). Incorporating the strength of MIP modeling in schedule construction. *RAIRO Operations Research 43*, 409–420.

Irving, R.W. (1994). Stable marriage and indifference. *Discrete Applied Mathematics 48*, 261–272.

Iwama, K., Manlove, D.F., Miyazaki, S., Morita, Y. (1999). Stable marriage with incomplete lists and ties. *Lecture Notes in Computer Science, Springer, Berlin 1644*, 443–452.

Iwama, K., Miyazaki, S., Okamoto, K. (2004). A $(2 - c(\log N/N))$-approximation algorithm for the stable mmarriage problem. *Lecture Notes in Computer Science, Springer, Berlin 3111*, 349–361.

Iwama, K., Miyazaki, S., Yamauchi, N. (2008). A $(2 - c(1/\sqrt{N}))$-approximation algorithm for the stable marriage problem. *Algorithmica 51*, 902–914.

Jackson, J.R. (1955). Scheduling a production line to minimize maximum tardiness. Management Science Research Project 43. University of California, Los Angeles.

Jans, R., Degraeve, Z. (2008). Modeling industrial lot sizing problems: a review. *International Journal of Production Research 46*, 1619-1643.

Johnson, S.M. (1954). Optimal two- and three-stage production with setup times included. *Naval Research Logistics Quarterly 1* , 61–68.

Khachiyan, L.G. (1979). A polynomial algorithm in linear programming. *Soviet Mathematics Doklady 20*, 191–194.

Karp, R.M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations* , 185-103.

Kellerer, H., Tautenhahn, T., Woeginger, G.J. (1996). Approximability and nonapproximability results for minimizing total flow time on a single machine. *SIAM Journal of Computing 28*, 191–194.

De Klerk, E., Roos, C., Terlaky, T. (2005). Lecture Notes: Continouous Optimization. Delft: Delft University of Technology, the Netherlands.

Lawler, E.L. (1978). Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics 2*, 75–90.

Lenstra, J.K., Rinnooy Kan, A.H.G. (1978). Complexity of scheduling under precedence constraints. *Operations Research 26*, 22–35.

Lenstra, J.K., Rinnooy Kan, A.H.G. (1980). Complexity results for scheduling chains on a single machine. *European Journal of Operational Research 4*, 270–275.

Lenstra, J.K., Shmoys, D.B., Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming 46*, 259–271.

Li, H., Womer K. (2009). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. *Journal of Scheduling 12*, 281–298.

Lin, S.H., Gertsch, R., Russell, J.R. (2007). A linear-time algorithm for finding optimal vehicle refueling policies. *Operations Research Letters 35*, 290–296.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L. (1998). An exact algorithm for the multiple resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science 44*, 714–729.

Möhring, R.H., Schulz, A.S., Stork, F., Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science 49*, 330–350.

Papadimitriou, C.H., Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity (2nd edition)* . New Jersey: Prentice-Hall, Englewood Cliffs.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems (3rd edition)* . New York: Prentice Hall.

Sedeo-Noda, A., J. Gutirrez, B. Abdul-Jalbar, Sicilia, J. (2004). An O(T log T) algorithm for the dynamic lot size problem with limited storage and linear costs. *Computational Optimization and Applications 28*, 311–323.

Schuurman, P., Woeginger, G.J. (2002). A PTAS for single machine scheduling with controllable processing times. *Acta Cybernetica 15*, 369-378.

Smith, W.E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly 3*, 59–66.

Tang, J., Kong, Y., Lau, H., Ip, A.W.H. (2010). A note on "Efficient feasibility testing for dial-a-ride problems". *Operations Research Letters 38*, 405–407.

Vande Vate, J.H. (1989). Linear programming brings marital bliss. *Operations Research Letters 8*, 147–153.

Valls, V., Perez, A., Quintanilla, S. (2009). Skill workforce scheduling in service centers. *European Journal of Operational Research 193*, 791–804.

Van den Heuvel, W. (2006). *The economic lot-sizing problem: new results and extensions.* PhD thesis, Erasmus University, Rotterdam, the Netherlands.

Williamson, D.P., Shmoys, D.B. (2011). The design of approximation algorithms. Cambridge University Press.

Woeginger, G.J. (2003). Exact algorithms for NP-hard problems: A Survey. *Combinatorial Optimization*, 185–208.

Woeginger, G.J. (2004). Inapproximability results for no-wait job shop scheduling. *Operations Research Letters 32*, 320–325.

Wu, M.C., Sun, S.H. (2006). A project scheduling and staff assignment model considering learning effect. *The International Journal of Advanced Manufacturing Technology 28*, 1190–1195.

Yoshimura, M., Fujimi, Y., Izui, K., Nishiwaki, S. (2006). Decision-making support system for human resource allocation in product development projects. *International Journal of Production Research 44*, 831–848.

Index

Summary

Workforce Scheduling and Planning: A Combinatorial Approach

This thesis investigates solution methodologies for concrete combinatorial problems in *scheduling* and *planning*. In all considered problems, it is assumed that the available information does not change over time; hence these problems have a deterministic structure.

The problems studied in this thesis are divided into two groups; "workforce scheduling" and "planning". In workforce scheduling, the center problem is to build a schedule of *tasks* and *technicians*. It is assumed that the time line is split into workdays. In every workday, tasks must be grouped as *sequences*, each being performed by a *team* of technicians. *Skill requirements* of every task in a sequence must be met by the assigned team. This scheduling problem with some other aspects is difficult to solve quickly and efficiently. We developed a Mixed Integer Programming (MIP) based heuristic approach to tackle this complex scheduling problem. Our MIP model is basically a formulation of the matching problem on bipartite graphs and it enabled us to have a global way of assigning technicians to tasks. It is capable of revising technician-task allocations and performs very well, especially in the case of rare skills.

A workday schedule of the aforementioned problem includes many-to-one type workforce assignments. As the second problem in workforce scheduling, *stability* of these workforce assignments is investigated. The stability definition of *Gale-Shapley* on the Marriage model is extended to the setting of multi-skill workforce assignments. It is shown that finding stable assignments is NP-hard. In some special cases stable assignments can be constructed in polynomial time. For the general case, we give linear inequalities of binary variables that describe the set of stable assignments. We propose a MIP model including these linear inequalities. To the best of our knowledge, the Gale-Shapley stability is not studied under the multi-skill workforce scheduling framework so far in the literature. The closed form description of stable assignments is also the first embedding of the Gale-Shapley stability concept into an NP-complete problem.

In the second problem group, two vehicle related problems are studied; the "dial a ride problem" and the "vehicle refueling problem". In the former, the goal is to check whether a list of pick-up and delivery tasks can be achieved under several timing constraints. It is shown this feasibility testing can be done in linear time using interval graphs. In the vehicle refueling problem, the goal is to make refueling decisions to

reach a destination such that the cost of the travel is minimized. A greedy algorithm is presented to find optimal refueling decisions. Moreover, it is shown that the vehicle refueling problem is equivalent to a flow problem on a special network.

Curriculum vitae

Murat Fırat was born in Istanbul, Turkey on October 3, 1978. In 1996, he completed his high school education at the *Üsküdar Anadolu Gymnasium* in Istanbul. In the same year, he enrolled at Boğaziçi University to study *Mechanical Engineering* and obtained his bachelor degree in 2001.

During his master program, he was as a research and teaching assistant of *Systems and Control Engineering* and he worked in the Boğaziçi University Flexible Automation and Intelligent Manufacturing (BUFAIM) Laboratory of *Industrial Engineering*. He was in the organizing committee of "Mathematics for Industry" (MathInd) project and worked with Yorgo Istefanopulos in 2001 and 2002. He obtained his Master's degree under the supervision of Ümit Bilge at the same university with the thesis entitled: A fuzzy logic approach to dynamic routing problem under varying levels of routing flexibility.

From August 2003 until February 2006, he worked for Otokar, a vehicle manufacturer, as management trainee and project engineer; for General System Design, a valve producer, as project coordinator.

In 2006, he decided to continue in the academia and he started working again in the BUFAIM Laboratory as a research and teaching assistant.

September 2007 Murat started as a Ph.D. student in the Combinatorial Optimization group at Eindhoven University of Technology under the supervision of Cor Hurkens and Gerhard Woeginger. His Ph.D. position was funded by the France Telecom/TU Eindhoven collaboration. As a Ph.D. student he was a member of BETA (Research School for Operations Management and Logistics) and the LNMB (Dutch Network of the Mathematics of Operations Research). For both he successfully completed all the required Ph.D. level courses. He also successfully completed the courses "Combinatorial Optimization at Work" (TU Berlin) and "Topics in Combinatorics" (Technical University of Denmark). As a member of the team with Cor Hurkens, Christian Eggermont, and Maciej Modelski, he won the 1st prize in the junior category of the ROADEF Challenge 2009.

Starting from April 2012 he will be working as a postdoctoral researcher in the field of combinatorial optimization with Alexandre Laugier at France Telecom.