# Empowerment Scheduling: A Multi-objective Optimization Approach Using Guided Local Search

Abdullah Alsheddy
School of Computer Science and Electronic Engineering
University of Essex

A thesis submitted for the degree of
*Doctor of Philosophy*

June 2011

# Abstract

Field Workforce Scheduling (FWS) is a very important and practical problem in service industries. It concerns the scheduling of multi-skilled employees to geographically dispersed tasks. In FWS, employee efficiency is highly important, and thus they have to be managed in an effective way. Employee empowerment is a relatively new and flexible management concept. It promises to benefit both organizations and employees by enhancing employee morale, satisfaction and productivity. This motivates the incorporation of empowerment when designing FWS models, which has not been thoroughly investigated.

This thesis describes the development of a new efficient *empowerment scheduling* model, called EmS, for FWS. The key feature of EmS is that it is strongly linked to the management literature on empowerment from which the requirements are derived. EmS provides employees with a simple, yet flexible and fair means of involvement in the scheduling decision, through which they can suggest their own schedules. This is formulated using a multi-objective optimization (MOO) approach where the task is to find a balance between employee empowerment and the employer's interest. To evaluate EmS, a series of empirical experiments are conducted, presenting the first extensive and in-depth study of the feasibility of empowerment in the FWS context, as well as the efficiency of an empowerment scheduling model.

To tackle the empowerment scheduling problem, a new method based on Guided Local Search (GLS) is developed. GLS is a simple, yet effective single-objective metaheuristic with few parameters to tune. As a pioneering work, we propose an extension to GLS (called GPLS) as a general method

for tackling MOO problems. In addition, a number of GPLS-based frameworks are proposed, which prove the potential of GPLS to be a central part of more advanced frameworks. GPLS and its frameworks are extensively tested on standard MOO benchmarks, and EmS. Computational results suggest that GPLS is comparable to state-of-the-art MOO metaheuristics.

To my Loving Mom, Dad and Wife!

# Acknowledgements

I would like to express in this single page my whole hearted thanks to everyone who offered any help or support to me during the three years of PhD.

Special thanks should go to my supervisor, Prof. Edward Tsang, for the professional guidance, the wise advice and the bright research ideas he gave me throughout my years of PhD study. From him I have learnt valuable research and thinking skills. He also offered me a great opportunity to work with British Telecommunication plc in an international project.

Many thanks to my colleagues and friends in Colchester for all the interesting (academic and non-academic) discussions and interactions we had. In particular Nasser Tairan, Abdullah Altayari, Mohammed Alkandri, Atif Alhejali, to name just a few.

Finally, I am grateful to my parents, brothers and sisters, who have constantly encouraged me to pursue my goals. I am also grateful to my wife for simply everything. With her, the PhD journey was a sweet dream and a pleasant experience.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Part I

# Introduction

# Chapter 1

# Introduction

## 1.1 Overview

Automating and optimizing service operations is becoming increasingly critical for many organizations and industrial companies; this motivates the development of intelligent planning and scheduling systems to manage and optimize the service operation and its associated resources [126]. A key resource that needs to be efficiently managed is people, who are the main asset in many organizations. A representative, yet very important, scheduling problem is the Field Workforce Scheduling (FWS) problem that appears in many organizations where the field workforce (e.g. technicians, engineers and salespersons) is the main resource. FWS concerns the scheduling of a multi-skilled workforce to geographically dispersed tasks in an efficient manner, whilst satisfying a range of operational constraints. It is an interesting combination of task assignment and routing problems. The importance of the FWS system as a decision support tool motivates the creation of new and efficient models, as well as advanced solution techniques. This describes the theme of this thesis, which aims to develop a new model for FWS and couple it with an advanced solution method.

### 1.1.1 Empowerment in Field Workforce Management

Organizations are becoming increasingly aware of the importance of empowering employees and involving them in decision making, in order to create an employee-friendly environment. In FWS, employee efficiency is highly critical to the effectiveness of the schedules produced by workforce scheduling systems. Nevertheless, traditional workforce scheduling models tend to apply traditional management techniques which are based on the command-and-control [51] management strategy. These models isolate employees, who have only to accept the scheduling decision, from the decision-making process.

Employee empowerment is one of the most recent and most flexible management concepts. It gives employees control over decisions related to their work, and enhances their self-efficacy. It has been argued that empowerment has promising benefits for both the organization and its employees, offering a win-win approach [51, 27, 29]. These appealing benefits motivate organizations to enhance, rather than minimize, employees' power and control, with the desire to increase productivity and quality, as well as to enhance employees' motivation and retention.

The incorporation of the principle of empowerment in designing workforce scheduling systems is still limited and challenging, particularly in the context of FWS. There have been attempts to implement empowerment in FWS (e.g. [116, 115, 102]), as well as other types of workforce scheduling (e.g. [11, 6, 35]). However, several issues have not been well-studied yet in the existing literature. Such issues include the following:

- Existing scheduling models which include empowerment vary significantly in their formalizations of empowerment at the decision-making (i.e. scheduling) stage. Such a variation reflects different conceptions of empowerment in the scheduling context. A common feature of these various conceptions of empowerment is that empowerment is treated mainly as a management practice that allows employees

to be involved (at various extent) in the allocation process. Focusing on the managerial aspect of empowerment leads many models to overlook other aspects such as enhancing employees' feeling of power and ensuring employees' trust in the model as being fair and transparent. These psychological aspects are crucial to the success of empowerment [130], and therefore it is not enough to claim empowerment by providing employees with a very limited power over their schedules, or proposing models that lack fairness.

- FWS is basically an optimization problem, where the task is to find an optimal schedule with respect to a special criterion (i.e. the organizational interest). Formulating this problem so as to empower the workforce and increase their involvement in the scheduling decision would impact, in various ways, the organizational goal. Studies that assess the impact of empowerment on the organizational objective and evaluate the various aspects (e.g. managerial and psychological aspects) of empowerment, are absent in most of the existing research. Such studies are very important in understanding the correlation between employee empowerment and employer's interest (i.e. examining the feasibility of empowerment), and evaluating the efficiency of an empowerment practice in workforce scheduling.

- A critical feature of the traditional (i.e. inflexible) scheduling approach is that the organization has full control over the optimization process, and thus the final scheduling decision. This is due to the fact that the optimization algorithm is steered by a criterion determined only by the organization, and therefore undesirable scheduling outcomes such as specific tasks being unscheduled can be avoided in an easy manner. Although this control is highly important to organizations, it disappears partially or completely from the existing flexible scheduling models that increase employee control over the optimization process. Empowering a workforce, while being in charge of the scheduling process as in traditional

scheduling, would motivate organizations to embark empowerment.

These issues suggest that incorporating empowerment effectively into a new FWS model is still challenging and requires more research.

### 1.1.2 An Advanced Optimization Technique

In real applications, the field service operation involves millions of pounds [126], and therefore a small improvement in FWS may result in large savings in real terms. This motivates the development of advanced scheduling techniques.

From an optimization point of view, we are concerned with metaheuristics [48], which are general-purpose optimization algorithms for solving search problems such as the FWS problem. Metaheuristics represent an optimization scheme that is characterized by its ease of implementation and its ability to obtain good performance in comparatively less computational time than classical optimization methods (i.e. exact methods).

Guided Local Search (GLS) is a relatively new metaheuristic method proposed by Voudouris and Tsang in 1997 [124] to solve combinatorial optimization problems. A main feature of GLS is that it is a flexible and rather simple to implement metaheuristic technique, with few parameters to tune. Additionally, it is a successful method showing a state-of-the-art performance on several applications to problems with various structures and objectives from scheduling and routing to assignment problems and constraint optimization.

Basically, GLS is a high-level strategy that applies an efficient penalty-based approach to interact with the underlying local improvement procedure. This interaction creates a process capable of escaping from local optima - solutions which are better than all the neighbours but not necessarily the best possible solution - which improves the efficiency and robustness of the underlying local search algorithms. Beyond that, several research studies have confirmed the ability of GLS to sit on top of other heuristics and

metaheuristics, and demonstrated the potential of GLS hybrids with other methods. For example, Guided Genetic Algorithm [75] demonstrates the ability of GLS to help the Genetic Algorithm (GA) to escape a local optimum population. In [84], an effective hybrid of GLS and Evolution Strategies is proposed and applied to a range of routing problems.

GLS was originally designed for solving single-objective optimization problems. It lacks the capability to handle multiple conflicting objectives simultaneously (i.e. multi-objective optimization), which is the nature of most real-world optimization problems. Multi-objective optimization problems (MOOPs) are very complex and the complexity, besides the combinatorial aspect, comes from the fact that there is no single optimum solution for these problems, but rather a set of trade-offs called Pareto optimum solutions. For instance, with relation to our research in this thesis, involving the workforce in the scheduling decision of FWS is expected to conflict with the organizational objective (e.g. maximizing the overall allocated tasks). Thus, instead of searching for the best schedule with respect to the organizational interest only, the search should target a balance between organizational objective and empowerment (i.e. the amount of employee decision power).

The application of GLS to MOOPs requires a careful adaptation of GLS's basic components. A potential direction is to focus on local search which is the basis of GLS. Pareto Local Search (PLS) [3, 95] is a straightforward, yet powerful extension of single-objective local search for solving MOOPs. As a local search method, PLS still suffers from the problem of being trapped at (Pareto) local optima, and therefore GLS may still be capable of sitting on top of PLS and guide it to escape Pareto local optima in an intelligent manner. The challenge is how to adapt GLS basic components to contain multiple objective scenarios and guide PLS.

### 1.1.3 Aim and Objectives

This thesis can be divided into two parts. In its first part, the thesis aims to develop a new modelling approach for FWS, which incorporates the empowerment concept. This aim is comprised of the following objectives:

1. Studying the *traditional* FWS model. This includes formulating a representative FWS problem and developing a problem generator to support the required empirical studies throughout the thesis.

2. Providing a rigorous formalization of empowerment (as a management principle), in workforce scheduling (as an optimization problem), derived from the management literature which is the source of empowerment. This formalization should define the constitution of empowerment and identify its requirements in the workforce scheduling context.

3. Developing a new empowerment scheduling model that carefully implements empowerment to satisfy its constitution and requirements. The model should then be applied to the FWS problem.

4. The model that is to be developed must be thoroughly evaluated in order to ensure an effective implementation of empowerment. Thus, the final objective is to conduct an extensive and in-depth series of empirical experiments to answer questions related to: (a) the cost impact of empowerment on organizational interests (i.e. the feasibility of empowerment), (b) the benefits to both employees and their employer, and (c) the efficiency and effectiveness of the proposed scheduling model.

In order to supplement the modelling aspect of the first part by an advanced scheduling technique, the second part of this thesis aims to extend the GLS metaheuristic to

solve optimization problems with multiple objectives. It includes the following objectives:

1. Investigating the adaptation of GLS to sit on top of PLS methods and guide them to escape Pareto local optima. This pioneering work results in a new extension to GLS, which is called Guided Pareto Local Search (GPLS), which allows for the application of GLS to MOOPs.

2. Formulating the new empowerment scheduling model for FWS as a biobjective optimization problem, and then solving it using GPLS. For evaluation purposes, the performance of GPLS is to be compared with various representative techniques applied to the new problem.

## 1.2 Thesis Structure

The structure of the thesis, which is depicted in Figure 1.1, is as follows. First, it begins with a background and literature review of the key terms and research areas used throughout the thesis. These are divided into two parts, the first one is provided in chapter 2 which describes the FWS problem and reviews its related literature. A great emphasis is devoted to the literature that proposes flexible management models for workforce scheduling. This includes a brief review of the literature of the management field on the employee empowerment concept.

The second part is given in chapter 3 which is dedicated to the literature related to the proposed solution techniques. It describes GLS in detail, and briefly reviews its applications. Then, the multi-objective optimization approach is introduced, and its main aspects are discussed. It also reviews the adaptations of heuristics and metaheuristics for solving MOOPs.

The computational studies in this thesis require a formulated FWS problem, with a problem generator, to develop large benchmark datasets. Due to the lack of a public

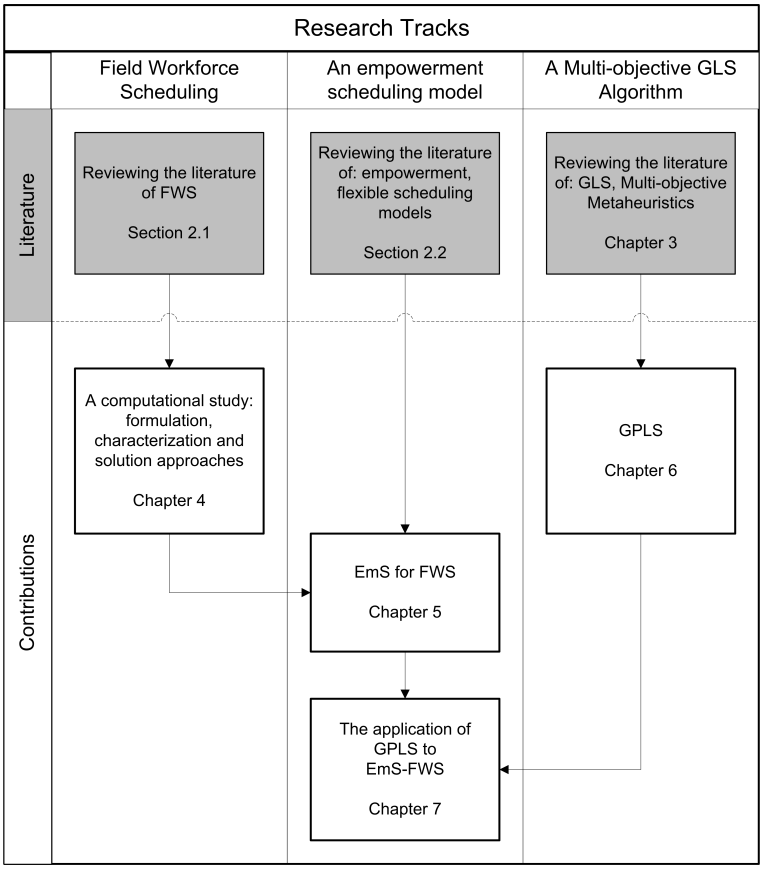



Figure 1.1: Thesis Structure

problem generator for FWS, a step in this direction is made in chapter 4, which formulates a *traditional* scheduling model of the FWS problem and then develops a problem generator. It also reports on an empirical study conducted to compare, under various characteristics, the performance of well-know assignment and routing heuristics applied to the formulated FWS problem.

Chapter 5 describes the development of the new scheduling model for FWS. It begins by establishing the term *Empowerment Scheduling* as a new conceptualization of empowerment in workforce scheduling, and proposing a new empowerment scheduling model, which we name EmS. Then, it describes the application of EmS to the FWS problem whose formulation is extended accordingly. It also reports on the extensive

computational study carried out to evaluate the model in comparison with the *traditional* model of FWS.

Chapter 6 proposes GPLS as an adaptation of GLS to tackle MOOPs. It outlines the algorithm and details its components. Moreover, it describes several GPLS-based frameworks. Then, it reports on the computational experiments conducted to examine the performance of GPLS and its frameworks on standard multi-objective optimization benchmarks, in comparison with various state-of-the-art techniques. These experiments are concluded with some parametric analysis of GPLS and its frameworks.

Chapter 7 presents the meeting point of the two main contributions of this thesis, namely EmS and GPLS. It describes the implementation of GPLS and its frameworks for the problem that results from applying EmS to FWS. It also details the conducted empirical experiments and discusses their results.

The thesis is concluded with chapter 8. It gives a summary of the thesis, lists its main contributions, and discusses possible directions for further works.

## 1.3 Publications

Some of the original material used in this thesis has been published in the following peer-reviewed papers:

1. A. Alsheddy and E. P. K. Tsang: Empowerment Scheduling for a Field Workforce. Journal of Scheduling, Springer Netherlands, 2011.

2. A. Alsheddy and E. P. K. Tsang: Empowerment-based Workforce Scheduling Problem. In MISTA 2009 Conference, Ireland, Dublin, 2009.

3. A. Alsheddy and E. P. K. Tsang: Guided Pareto Local Search and its Application to the 0/1 Multi-objective Knapsack Problems. In The Metaheuristic Internatinal Conference (MIC09), Germany, Hamburg, 2009.

4. A. Alsheddy and E. P. K. Tsang: Guided Pareto Local Search based Frameworks for Biobjective Optimization. In The Congress on Evolutionary Computation (CEC2010), Spain, Barcelona, IEEE Press, 2010.

5. A. Alsheddy and E. P. K. Tsang: A Guided Local Search Based Algorithm for The Multiobjective Empowerment-based Field Workforce Scheduling. In The UKCI 2010 Workshop, UK, Colchester, IEEE Press, 2010.

6. C. Voudouris, E. P. K. Tsang, and A. Alsheddy: Guided Local Search. Chapter 11, in M. Gendreau and JY. Potvin (ed.), Handbook of Metaheuristics, Springer US, 2010.

7. C. Voudouris, E. P. K. Tsang, and A. Alsheddy: Guided Local Search. Wiley Encyclopaedia of Operations Research and Management Science, 2010.

8. C. Voudouris, E. P. K. Tsang, and A. Alsheddy: Effective Application of Guided Local Search. Wiley Encyclopaedia of Operations Research and Management Science, 2010.

# Part II

# Background and Literature Survey

# Chapter 2

# Field Workforce Scheduling and Empowerment

This chapter reviews the literature related to Field Workforce Scheduling (FWS). The first part (section 2.1) is dedicated to the FWS problem. It begins with a short overview of workforce scheduling problems in general, and then the FWS problem is described and its variants and applications are reviewed. The second part (section 2.2) focuses on workforce scheduling models that attempt to apply flexible management techniques (e.g. employee empowerment) with the aim to empower the workforce to express their wishes and preferences about their schedules. The chapter concludes with section 2.3.

## 2.1 Field Workforce Scheduling

### 2.1.1 A General View

Workforce (or personnel) scheduling concerns the process of constructing work schedules for employees in an efficient and optimized manner in order to meet a range of internal and external requirements [41]. This process may involve several stages, each of which is itself a complex and highly constrained optimization problem. However, there could be

an overlap between these stages, which makes it difficult to define a unified classification of these optimization sub-problems. Among the several classifications (e.g. [111, 19, 41]), a general taxonomy is proposed by Ernst et al. [41] who suggested six modules associated with the process of workforce scheduling. Each module may have several models needed for specific applications. For a particular application, some or all of these modules may require consideration. These modules are defined as follows:

1. Demand Modelling. This is a planning problem that involves the determination of the staffing requirements to cover the planning horizon. An application of such a problem appears in organizations where the workforce demand varies during the week as well as the day. An example of such organizations is the call centre [46].

2. Days Off Scheduling. This is about determining which days of the week an employee should be assigned to be off. This is a practical problem for organizations that operate seven days a week.

3. Shift Scheduling. It concerns the selection of which shift an employee should be assigned each working day. There may be also a need to decide the size of the workforce required for each shift to meet the target demand. This problem deals with scenarios where multiple shifts are required per day.

4. Line of Work Construction. This is also known as *tour scheduling* (i.e. a combination of Days off and Shift scheduling), and it deals with the problem of deciding the work line, or the schedule pattern, for each employee over the whole scheduling horizon. This is a practical problem for organizations that operate seven days a week and 24 hours a day. These organizations include hospitals and airlines.

5. Shift Assignment. This is the process of assigning employees to work lines.

6. Task Assignment. This involves the assignment of tasks to employees during each

shift, whilst satisfying constraints such as skills and service time-windows.

According to this classification, the FWS problem, which is the concern of this thesis, fits into the *task assignment* module. It is a task assignment problem characterized by a range of requirements such as a multi-skilled workforce, a service time-window and tasks with different geographical locations. Therefore, we will focus on the *task assignment* module from here onward. For other modules, there is a large body of research on these problems in the literature of Artificial Intelligence and Operational Research. The reader is referred to survey papers on workforce scheduling in general such as [41, 2], and other reviews of particular applications such as that of nurse scheduling [23] and call centres [46].

### 2.1.2 Problem Description

In the telecommunication domain, Lesaint et al. [76] describes field workforce scheduling as "sending the right engineer to the right customer at the right place at the right time with the right equipment at any time and in any operational environment." More generally, we define the field workforce scheduling problem as a combination of task assignment and routing problems, where the task is to schedule a workforce to geographically dispersed tasks in an efficient manner, whilst satisfying a wide range of operational constraints. Such constraints are real-life requirements which normally exist in the application domain, and they describe the scheduling context (or environment) of the considered FWS problem. These requirements can be categorized into the following groups:

- Scheduling context related requirements:
  - *Planning horizon:* This refers to the time period of how far into the future the organization schedules its workforce. This can range from a single work day to multiple days (e.g. weeks or months).

- Resource related requirements:
  - *Workforce qualification:* This indicates whether employees vary in the skills they possess, and thus whether each task requires a particular skill.
  - *Shift:* This describes the working hours of an employee, given that employees may vary in their working hours per working day.
- Task related requirements:
  - *Service time-window:* This indicates whether each task should be scheduled within a predefined time-window.
  - *Dependencies:* This describes the inter-task dependencies, if any. This includes for example precedence and parallel constraints.
  - *Priority:* This corresponds to the importance of each task to the organization, if they vary. This is a practical requirement, particularly when it is not necessarily possible to schedule all tasks on the same scheduling unit (e.g. day).
- Routing requirements:
  - *Depot:* Employees may start their work from a defined depot, multiple depots, their homes or mixture of homes and multiple depots.

Some or all of these requirements may require consideration in order to model a particular application of FWS. Thus, the above classification of requirements helps to describe the different variants of FWS. It also justifies the existence of different models, solution quality measurements (i.e. objective functions) and solution techniques applied to various FWS problems in the literature.

A central objective of the FWS problem is normally to find a schedule that maximizes the productivity (i.e. the service quality) expressed in terms of the number of

allocated tasks with respect to their priorities. This is attributed to the existence of many constraints, in which finding a feasible schedule of all tasks is almost impossible. Another important objective of the FWS problem is to minimize the operational cost, e.g. travelling cost and over-time cost.

The FWS problem is a practical problem for a wide range of organizations, particularly service companies where the field service engineer is a key resource to be managed. The domain of such companies include telecommunication, maintenance, utility, construction and health care. To illustrate the importance of the FWS problem, British Telecommunications plc (BT), for example, employs thousands of field engineers across the UK to maintain networks, repair faults, and provide service to customers [76]. By developing an intelligent solution to allocate the workforce efficiently, the company was able to save about $150 Million a year on operational cost. This example illustrates the importance of this problem.

### 2.1.3 Related Work: FWS models

A major feature that characterizes the FWS problem is its involvement to two interdependent sub-problems: task assignment and routing. Therefore, the literature related to FWS can be divided into three groups: literature that deals with the assignment problem, literature that focuses on the routing problem, and those which deal simultaneously with problems that involve both aspects.

#### 2.1.3.1 Variants of FWS

A number of FWS problems that consider both the assignment and routing aspects have been defined and formulated in the literature. Most of this literature has considered real-world problems. These are reviewed here with an emphasis on problem formulation.

Azarmi and Abdul-Hameed [7] presented the workforce management problem of BT. The problem is to find optimized tours for the field engineers to serve a set of

tasks, so as to serve as many tasks as possible and minimize the travel time. A data set with 250 tasks and 118 engineers was considered in this work, which then was used as a benchmark by other researchers who proposed various solution techniques, including constraint logic programming [7, 78, 133], Simulated Annealing [9], Genetic Algorithm [90] and Guided Local Search [114]. The BT problem is a typical FWS problem as it involves most of the real requirements such as multi-skilled engineers with multiple depot, tasks with different service time-windows and priorities. The dynamic counterpart of the BT FWS problem has also been considered in [77, 16]. Similarly, the Field Technician Scheduling Problem (FTSP) was introduced and formulated by Xu and Chiu [131]. The FTSP considers maximizing both the allocated tasks as a primary objective, and the remaining time for each technicians as a secondary objective. Both terms were modelled in a single objective with weighting coefficients. In [131], several heuristic procedures including GRASP-based heuristic were designed to solve the FTSP. In addition, a relevant problem named as *mobile workforce management*, that arises in a large service orientated telecommunication enterprise, was introduced and formulated with multi-agents in [26]. In the same domain, Dutot et al. [39] presented the Technician and Task Scheduling Problem (TTSP) in collaboration with France Télécom. The problem was introduced as the subject of the 2007 challenge set up by the French Operational Research Society (ROADEF)[1]. In [31], Cordeau et al. described their entry in this competition.

Naveh et al. [91] modelled a relevant FWS problem as a constraint satisfaction problem. The model concerns the scheduling of a highly-skilled workforce, and takes into account more resource related considerations such as skill level, language and retraining. The model was applied to workforce management in IBM Global Services.

Cowling et al. [33] formulated a FWS problem as a Resource Constrained Project Scheduling Problem (RCPSP), which is justified since the considered problem involves

[1] http://challenge.roadef.org/2007/

rich inter-task dependencies such as precedence and parallel constraints. The problem is solved using a multi-objective optimization approach to optimize multiple objectives simultaneously. These objectives are the maximization of the productivity (i.e. the scheduled tasks with respect to their priorities) and the minimization of the operational costs including the total travelling time. A Genetic Algorithm based multi-objective approach is proposed as the solution technique.

Tang et al. [107] solved a FWS problem that appears in a manufacturer that has to provide a maintenance service for their advanced equipments at customer sites. The service region is clustered into small areas. Each area is assigned to a particular technician. Then, each technician will perform the tasks in the assigned area over a scheduling horizon which can span over multiple days. The aim is to find a multiple routes for each technician (i.e. a route for each day) while maximizing the collected rewards during the scheduling horizon. There is no qualification constraint in this problem, and thus all technician can do any task. Moreover, the service time-window can be considered as a soft constraint which can be violated at an incurred cost. Tang et al. [107] modelled this problem as a *multiple tour maximum collection problem with time-dependent rewards*, which is a variant belonging to the OP class. They focused on the routing problem, assuming that tasks are already assigned to a technician and the objective is to find an optimized tour for each day. Thus, optimizing the tours for each technician can be solved independently. A tabu search based heuristic embedded in an adaptive memory procedure was devised as a solution.

The Home-Health-Care (HHC) problem, i.e. nursing patients in their homes, [15] is to find a feasible working roster for each nurse to complete a set of geographically dispersed jobs, which has to respect a variety of hard and soft constraints and preferences. The HHC problem includes both routing and assignment constraints. However, a distinguishing feature of this problem is the treatment of the allocation of all jobs as a hard constraint that must be satisfied. It also defines for each routing and assign-

ment aspect (e.g. nurse qualification, job's time-window, and nurses' working time) hard and soft constraints; whereas in the general FWS problem, only the hard constraints on these aspects are defined. Normally, there is no priority defined for the completion of jobs in the HHC problem, and thus its objective is to minimize both the violated soft constraints and travel costs. In [15], Bertels and Fahle proposed an integrated approach that interweaves two stages: assigning jobs to nurses, and finding an optimized sequence for each nurse. A combination of linear programming, constraint programming and metaheuristics (a Tabu Search and Simulated Annealing algorithms) was proposed to tackle this problem. In relation to HHC, the development of a decision support system applied to a real HHC problem is described in [43].

Summarizing, there has been a considerable amount of research on modelling and solving various FWS problems which are treated as a combination of assignment and routing. Most of these studies considered real problems (e.g. the FWS problems of BT [7], IBM [91], Vidus Ltd. [33]. and United Technologies Corporation (UTC) [107]), and thus they might be unable to expose real world data due to confidentiality issues. As a result, these studies focused mainly on problem formulation and solution techniques, leading to a lack of a theoretical research that studies the characteristics of problem instances. The latter would help in identifying properties that describe hard instances or at least the performance difference between solution approaches. Another advantage of such a theoretical study is its help in developing a general benchmark for further scientific research.

#### 2.1.3.2 Task Assignment

In the assignment problem [72, 96], the task is to find a minimum cost assignment of a set of jobs to a set of agents, such that each job is assigned to exactly one agent, subject to agents' available capacities. Each job-to-agent assignment may have individual cost and resource requirements. If time is a coordinate in the assignment process, then the

problem may be called scheduling or timetabling [99].

The FWS problem can be considered as an assignment problem, particularly the Generalized Assignment Problem (GAP) which is the most basic variant of the assignment problem that allows an agent to be assigned many jobs while respecting the agent requirement constraint [96, 132]. Similarly, the FWS problem concerns the scheduling of multiple jobs (i.e. tasks) with different costs (i.e tasks' priorities) to the available agents (i.e. employees/technicians) with capacity (i.e. time) constraints. The resource requirement of each task includes the summation of task duration and the travelling time a resource needs to get to the site of that task.

However, there are key differences that makes the FWS problem is more general than the GAP, the first difference is the routing sub-problem involved in FWS which is itself a non-trivial optimization problem. Second, the possibility of finding an assignment for all tasks is normally difficult in the FWS scenario, due to the large number of tasks compared to the available resources, as well as the many existing constraints. Thus, the objective in the FWS problem is to maximize the number of allocated tasks with respect to their importance (i.e. priority). This is not the case in the GAP where finding an assignment for each task is a hard constraint that must be satisfied. Finally, the FWS problem involves additional constraints which are not considered by the classical GAP. These constraints include the service time-window for each task, and the multi-skilled resources. An attempt to consider the GAP with multi-skilled agents is presented in [24].

#### 2.1.3.3 Vehicle Routing

The vehicle routing problem (VRP) [112] is a general, well-defined benchmark, which can be described as follows: given a fleet of vehicles that is based at a depot, and a set of geographically dispersed customers, the task is to construct a set of routes for the vehicles that optimizes an objective function, whilst satisfying operational constraints

such as capacity and route length. The objective function is mainly to minimize the operational costs, represented usually by the number of vehicles and total travelling distances. The vehicle routing problem with time windows (VRPTW) [30] is an extension of the VRP with latest, earliest and service times for customers. In the last few decades, the VRP and its variants have received much attention due to their importance and practicality in many real applications such as transportation and network domains. Recent surveys on the VRP can be found in [112, 30, 74].

The FWS problem can be thought of as a generalization of the VRPTW. Similar to the VRPTW, the FWS problem considers the scheduling of multiple vehicles to customers with time-window constraints. However, in FWS the fleet is not homogeneous: resources have different numbers and types of skills, as well as different depots. In addition, tasks vary in their skill requirements, and may have different priorities. The actual duration of a task in FWS is considerably longer than it is in the case of the VRPTW. Finally, while minimizing the operational cost (e.g. travel cost) is the major objective of the VRPTW, it is still an important objective of FWS, besides the main objective which is to maximize productivity (i.e minimize the unallocated tasks with respect to their priorities). There have been attempts to generalize the VRPTW to deal with such additional constraints. For instance, the multiple depot and heterogeneous fleet variant of the VRP is considered in [38].

Another variant of the VRP which has attracted the attention of researchers recently is the Orienteering Problem (OP) [44] which also known as the travelling salesperson with profit. In the OP, each location is given a score/profit and the goal is to determine a route (or multiple routes in the vehicle routing problem with profit) to visit a subset of a given set of locations. The objective is to maximize the collected scores, given that the travel cost does not exceed a pre-set maximum value. Similar to the FWS problem, the OP involves both assignment and routing problems, and has a similar objective. In contrast to the FWS problem, the OP does not deal with other requirements such

as skill matching and service time-window constraints on the visits, non-neglected task duration, and resources with multiple depots.

## 2.2 Flexible Workforce Management

In FWS (and many workforce scheduling problems), employee efficiency is highly critical to the effectiveness of the schedules produced by the workforce scheduling systems. This is because time is a coordinate in the process. Unless employees are highly motivated and efficient, they could easily introduce delays which would significantly impact upon the schedule of subsequent tasks, as well as the overall schedule. In traditional scheduling models, however, the involvement of employees in the scheduling process is very limited. Such models tend to apply traditional (inflexible) management techniques which are based on the command-and-control management strategy [51]. These systems isolate employees from the decision-making process, and leave them powerless in such a critical system.

Organizations are becoming increasingly aware of the importance of implementing a flexible management technique which empowers employees to express their preferences and influence the decision-making process. Employee empowerment is a relatively new, comprehensive management approach that gives employees more freedom and control over decisions related to their work. There have been attempts to develop scheduling models that try to implement, at various extents, employee empowerment, with the desire to increase productivity and quality, as well as to enhance employee motivation and retention. These models vary significantly in their conceptions of empowerment in workforce scheduling. This motivates us to study empowerment as a management concept from its source (i.e. the management literature), and then formalize this concept in the scheduling context. This approach would greatly help in understanding the essence of empowerment and in evaluating the current empowerment practices implemented in

scheduling models. For this purpose, this section introduces the empowerment concept from a management perspective, and then scheduling models with flexible management in the literature of workforce scheduling are reviewed.

### 2.2.1 Empowerment: A Management Perspective

#### 2.2.1.1 What Does Empowerment Mean?

Empowerment as a management concept is an elastic term which has been loosely defined and used [130] [51]. Essentially, employee empowerment is a management concept which provides employees with a certain amount of freedom and flexibility to make decisions related to their work.

Two concepts of empowerment are reflected in the literature [29] [51] [37] [1]. The first approach conceptualizes empowerment as a managerial relation, and defines this term as the process of enhancing employee authority and control over decisions related to their tasks. From this perspective, empowerment is a broad concept which encompasses other management ideas such as delegation, job enrichment, decentralization of decision making and participatory management. The second approach, however, emphasizes the psychological values of empowerment, and refers to empowerment as the process of enhancing the motivational concept of self-efficacy. Other researchers tend to combine the two approaches, viewing empowerment as enhancing both employee decision power and self-efficacy.

#### 2.2.1.2 Benefits of Empowerment

The empowerment literature delivers convincing arguments concerning the benefits for employee empowerment [53]. Empowerment is seen to be beneficial for both the organization and employees. As Greasley et al. [51] summarize, the benefits for an organization are the remarkable improvements in cost control, flexibility, productivity and quality; where the benefits of empowered employees are enhanced job satisfaction,

motivation and organizational loyalty. These mutual benefits present a win-win scenario which is considered as the main motive for making a move towards empowerment [27].

The literature is rich with success stories from organizations that have embarked on this journey. Examples of such stories are reported in [103, 58, 1, 53].

An important aspect of empowerment that has been studied is the employee perception of empowerment (e.g. [51]). Wilkinson [130] stated that "It is taken for granted in much of the prescriptive literature that employees will welcome and indeed be committed to the new approach. Indeed there is evidence that workers welcome the removal of irritants (e.g. close supervision) and welcome the opportunity to address problems at source as well as the ability to decide work allocation." Yet, it has been argued that employee's perception and commitment varies according to several factors such as education, experience, skilfulness, and personal characteristics [37, 58].

#### 2.2.1.3 Practices and Implementation Issues

The management literature has reported numerous empowerment initiatives and practices. Wilkinson [130] classified empowerment initiatives into five categories: information sharing in which the organizational goals are shared with employees who are encouraged to express their views to the management; upward problem solving through which the employees' ability to make customer-related decisions is enhanced; task autonomy through team-working or self-management teams; attitudinal changes in which employees are educated to feel empowered even though there is no organizational change; and self-management. However, there could be an overlap between these classes. Some writers emphasize changes in attitude and self-efficacy as being the core of any form of empowerment.

Another important issue in implementing empowerment, and any new approach, is the measurement process, which has been discussed in [51]. This process requires three different measures for measuring three aspects of empowerment: implementation

efficiency, organizational benefits, and employee benefits. It is easier to measure the benefits for the organization than it is to measure that of the employee, though both are difficult processes. The reason for this is that organizational benefits can be measured by using objective "facts" such as cost and performance, whereas employee benefits are much more subjective where certain facts can be applied such as absence rate [51].

#### 2.2.1.4 Review of the Empowerment Literature

One major observation of our review of empowerment practices in the management literature is that most empowerment practices are principally human-centric, in which the amount of redistributed power, alongside the exercises and effectiveness of the empowerment strategy, depends entirely on the people in the organization, i.e. managers and employees. It is possible to say that the contribution of technology to this management style is still modest. The main reason behind this is that the empowerment practices suggested in the management literature focus on enhancing employee power and control over task-related decisions, but have not, to any great extent, been extended to include decisions made by supportive subsystems, particularly scheduling systems.

Efficient extensions to the scope of empowerment practices to include such decision-support subsystems would, on one hand, create more opportunities for employees to get involved. On the other hand, it could be beneficial for those supportive systems to apply new decision-making strategies that could help improve the service quality.

### 2.2.2 Related Work: Flexible Scheduling Models

There have been attempts to implement the empowerment approach in workforce scheduling in general, and task assignment in particular. Among these attempts, *self-scheduling* and *preference scheduling* are the most common approaches which have attracted researchers in the last few decades.

#### 2.2.2.1 Self-scheduling

Self-scheduling is an empowerment practice designed to enhance employee involvement in decision making [87, 60]. This flexible scheduling model is designed mainly for dealing with rostering problems. Various implementation practices of self-scheduling have been proposed, most of which consider nurse rostering [109, 8]. The idea of self-scheduling is basically to enable a group of staff nurses to make their own schedule and select their shifts in accordance with the staffing requirements as determined by a manager.

To illustrate this employee empowerment tool, Hung [60] suggested a nurse self-scheduling practice as follows:

> " The self-scheduling process is more or less as follows:
>
> 1. A large worksheet for the next schedule period, typically 4 to 6 weeks, is posted weeks in advance, with guidelines or requirements (for example, number of weekend shifts one must work, maximum number of consecutive shifts, and a shift's required staff coverage). The worksheet may be blank or partly filled (this happens, for example, when some nurses have been working the same shift patterns for years and they don't want to change their shift patterns).
> 2. Nurses are given 1 to 2 weeks to fill in the blanks. When a nurse fills the worksheet, she/he looks at what has already been filled and tries to avoid violating guidelines or requirements by making a concession or trading shifts with other nurses. Filling the worksheet can be done on a first-come first-serve basis or seniority basis. Some divide nurses into several groups with rotating priority (for example, the group that had second priority last time will have first priority this time and least priority next time, etc.) to achieve more fairness. The simple first-come first-serve basis probably works well since peer pressure would

prevent anyone from regularly signing all the good shifts. In addition, on the first-come first-serve basis, those who really need certain days off can increase their chance of getting them by signing up first.

3. When the sign-up period expires, the nurse manager or schedule facilitator looks at the filled worksheet to see if guidelines or requirements are met. If not, the worksheet is posted again, with problems highlighted. Nurses negotiate and make changes.

4. If the resulting worksheet has no problems, the nurse manager or schedule facilitator approves it. Otherwise, the manager makes necessary adjustments, perhaps after consulting with those affected. The final schedule is posted well before it is effective so that nurses can plan their activities well in advance.

5. Subsequent schedule changes are possible as long as nurses and management agree."

The scheduling process in self-scheduling is done manually, and requires sufficient time to resolve conflicts between staff choices. These make self-scheduling an acceptable empowerment practice for tractable problems (i.e. units with very few nurses). However, there are difficulties in implementing self-scheduling for large staff groups (i.e. more than 70) [8], as well as in short-term scheduling problems. In addition, most self-scheduling models lack transparency and fairness, which impacts negatively on employee trust in the system. Teahan [109] stated that a major negative outcome of implementing the self-scheduling was employees' complaints of favouritism by the schedulers irrespective of their objectivity and fairness. Therefore, there is considerable room for further improvements and research to develop an automated, general self-scheduling practice.

#### 2.2.2.2 Preference Scheduling

Preference Scheduling is an automated scheduling approach that attempts to accommodate individual preferences when creating schedules. Preferences are quantitatively measured, and are then considered by the scheduling system as an objective to maximize. Several models of preference scheduling were proposed to solve scheduling problems such as employee tour scheduling [2, 82, 134], nurse scheduling [23, 11, 6, 35], and field workforce scheduling [114, 115].

We classify these models into four main approaches. The first class enhances the quality of the schedule by extending the objective function (as a performance indicator) to include an employee satisfaction measure. The main feature of this approach is that the new measure is defined by the organization without explicit employee involvement. The definition of the employee's satisfaction measure follows one of the following two ways:

1. Incorporating information about the preferred working patterns for employees. Common preferred working patterns are general information which can be captured easily with or without employee input. For example, in rostering problems, a common preferred pattern is to have two consecutive off days. The preferred patterns are normally modelled as soft constraints that incur penalties when violated. The violation costs are defined by the model (without employee involvement) using a grouping approach that identifies different groups of violations [22, 11]. Each group is then assigned a cost coefficient.

2. Enhancing the fairness of the model by defining a function that quantifies the fairness of a schedule. An example of such a function is to maintain a load balancing between employees [114]. This function, then, is added as a term in the global objective function.

It is possible to say that these models claim empowerment by defining (based on the

organizational perspective) an employee satisfaction measure without explicit employee involvement.

The second approach allows employees to express their preferences by associating weights to a particular property of the scheduling item (e.g. type of tasks/shifts). For instance, in a FWS problem with multi-skilled engineers [114], each engineer is able to rank the skills that (s)he prefers using on a numbering scale that ranges from one to the total number of skills that the engineer has, such that the higher the number the more preferred the skill. This enables engineers to suggest the types of tasks they wish to do, as tasks are described by their skill requirements. This ranking mechanism influences the scheduling process since the model adds, to the global objective, a term that considers maximizing the collected preference scores specified by the engineers. An alternative ranking mechanism applied to nurse scheduling models is via *points* [97]. In this approach, all preferences are enumerated and an employee is asked to decide their violation penalties by allocating 100 points among them.

Models in this class empower employees by providing a limited involvement practice that enables employees to express their preferences about the scheduled item.

The third approach includes models that extend the scope of preferences in the second approach. Alongside the employees' general preferences that are to be considered in every schedule, this approach enables each employee to make a request once every schedule/roster [11, 42, 35]. This request gives employees the chance to express their specific requirements for a particular day.

These models claim empowerment by providing employees with an enhanced, flexible empowerment practice which enables employees to suggest their specific schedules. A critical issue that is underestimated by this approach is fairness, neglecting that it could be the case where the preferences and requests of particular employees are continually being violated, while others are always satisfied.

The fourth approach includes few models that enhance the fairness of the third

approach. Eveborn and Rönnqvist [42] proposed a simple mechanism to maintain fairness, by defining a maximum number of violations for each employee schedule. Fairer alternatives have been proposed in nurse scheduling [11, 35]. These models enhance the balance of employees' individual satisfaction by automatically controlling the cost coefficient of violating each preference via incorporating knowledge from the satisfaction history of employees using the output of previous schedules. The idea is that employees who were unsatisfied in previous schedules, would be satisfied in the current one.

Bard and Purnomo [11] implemented this idea by defining the maximum number of violations in a new schedule for each employee. This number is calculated as a function of the mean and standard deviation of the number of violations in the previous schedule. When the number of violations for an employee is above the mean plus the standard deviation, the maximum number of violations for this employee in the next schedule is limited to the mean minus the standard deviation. A central assumption for this model is that a nurse can be partially satisfied when a subset of his/her set of preferences was considered in the final roster. Thus, it cannot deal with situations where employee preferences can be either satisfied or not in the final schedule. This model, moreover, considers only the last schedule to derive the values for the new schedule. A fairer model would need to increase the history of employee satisfaction.

Another implementation of this idea is via auction. De Grano et al. [35] suggested, in their auction-based model for nurse scheduling, that nurses be given the chance to bid on work shifts and rest days using 'points'. These points can be controlled by allowing nurses to roll over both the unused points and points associated to unsuccessful bids to the next schedules. A major concern about this model, as an auction-based approach, is its high susceptibility to game playing. This is due to the fact that employees vary in their education, skilfulness and personal characteristics, which make those who are talented enough to game the system have an advantage over the unwilling employees.

#### 2.2.2.3 Miscellaneous Models

It has been reported that an on-line voting system was used as an empowerment practice, through which employees were given a set of activities and allowed to vote on-line for the activities they most wished to do [103]. Although voting systems are normally designed to support industrial democracy, it can also be considered as a supportive practice to involve workers in the decision-making process. Nevertheless, this technique is suitable for teamwork, rather than individual, contexts where democracy could be a very practical practice. Moreover, task assignment problems usually deal with very short-term, partially dynamic jobs, in which there is no sufficient time to utilize this technique.

In solving the FWS problem, the first explicit initiative that adopted empowerment was introduced by Tsang et al. [116, 117]. Their approach was to model the problem's entities (e.g. manager, engineers and jobs) with intelligent agents, each of which has its own interests which are in conflict with others. This approach formulates the problem as a distributed scheduling problem, allowing each agent to look at its interests, while the manager agent looks at the organizational interest. Giving engineers the chance to pursue their interests is claimed to be the empowerment practice in this model. However, employees' powers to attain their interests are strongly controlled by the manager agent who decides the amount of power given to each agent. A schedule is generated as a response to the manager's decision of the weights of engineers' objectives. Since generating the weights is modelled as an optimization problem, it is most likely to have satisfaction imbalance between engineers.

For dynamic FWS problems, Shah et al. [102] propose a new allocation procedure using the so-called *pull* strategy, in contrast to the traditional *push* strategy. In the push approach, the employee has only to accept the allocation decision made by the scheduler. Instead, a pull strategy is proposed as an empowerment practice through which an employee requests from the scheduler a set of tasks that (s)he can do next, and

then orders these tasks based on his/her preference. The main motivational assumption of applying the pull strategy is the enhancement of employees' performance, since the commitment of undertaking a task is initiated by the employee rather than the scheduler. Although the authors explicitly claim the implementation of empowerment, we see this model to be within the *preference scheduling* class, however, with a new way for employees to express their preferences about which tasks they wish to do. The main concerns with preference scheduling still exist in this model, such as flexibility, transparency and fairness. For instance, as described in [102], the criterion upon which the scheduler decides which set of tasks to be sent to a technician merely reflects the organizational interests (e.g. allocating tasks with high importance). This would limit the employees' control over the allocation decision, since the provided set does not necessarily enable employees to describe their specific requests and preferences.

## 2.3 Conclusions

Field workforce scheduling (FWS) is a task assignment problem that involves routing as a sub-problem, since tasks are geographically dispersed. FWS considers many real-life operational requirements, including multi-skilled employees and tasks with different priorities and time-windows. The FWS problem is a very practical problem that appears in a wide range of service companies where the field service engineer is a key resource to be managed. Different variants of the problem were extensively studied, focusing on tackling real applications of FWS problems (e.g. BT and IBM). Due to confidentiality issues, instances related to real world data could not be exposed and hence there is no public benchmark dataset or problem generator for FWS. Thus, we have to make efforts to formulate a FWS problem and define its characteristics, which would help to generate relevant benchmarks, and to describe the performance differences between various solution approaches. These efforts are presented in chapter 4.

The importance of FWS motivates the continuous development of new and efficient scheduling models. Traditional scheduling models tend to be inflexible in managing employees who have no say in the scheduling process. Organizations are moving towards applying flexible management approaches that empower employees to express their preferences for the tasks they wish to do. Examples of such flexible scheduling models are self-scheduling and preference scheduling, both of which try to incorporate, to various extents, the empowerment concept. However, these models reveal that there is no consensus in defining what constitutes empowerment in workforce scheduling. Therefore, it is an important step to study empowerment as a management concept from its source and then establish a formalization of empowerment in the scheduling context. The first aspect has been given in this chapter, which provides an overview of empowerment from the management literature. This helps understanding the essence of empowerment, which is a combination of a managerial relation (i.e. enhancing employee power over decision making) and a psychological value (i.e. enhancing their self-efficacy).

Formalizing the empowerment concept in the workforce scheduling context, and designing an efficient flexible scheduling model accordingly, are challenges which are addressed in chapter 5. Moreover, as to be shown later in the thesis, solving the new scheduling problem (with the empowerment concept incorporated) is also rather challenging.

# Chapter 3

# Guided Local Search and Multi-Objective Optimization

From an optimization perspective, our research is concerned with investigating the application of Guided Local Search, which is a well-known single-objective metaheuristic, to a multi-objective based model for the FWS problem. Thus, this chapter provides an introduction to Guided Local Search and reviews its application to workforce scheduling, section 3.1. In section 3.2, the multi-objective optimization is defined and its related literature on the adaptation of metaheuristics to tackle multi-objective optimization problems are reviewed. The chapter concludes with section 3.3.

## 3.1 Guided Local Search

Many optimization problems found in the real world are combinatorial explosion problems. Classical methods such as complete search often encounter great difficulty in solving such problems within reasonable computational times. This motivates the development of heuristic methods that sacrifice completeness. Some of the best known heuristics are local search methods. Local search (LS) is the basis of most heuristic

search methods. LS works by starting with an initial solution (randomly or heuristically generated), and then iteratively moving to 'neighbour' solutions that improve the objective function. A 'neighbour' solution is usually obtained by making local (often small) changes to the current solution. LS terminates when it reaches a local optimum (i.e. a state where the current solution is superior to all its neighbours) or computational resources run out.

LS can obtain good quality solutions very quickly. However, a major issue that negatively affects its efficiency is that it can be trapped in local optima which halt the search, preventing the algorithm from reaching the global optimum. To overcome this problem a class of techniques, which is referred to as metaheuristics [48], has been introduced and grown in popularity over the years. Simulated Annealing (SA) [70], Tabu Search (TS) [49] and Genetic Algorithm (GA) [50] are well-known and representative techniques in this class.

Guided Local Search (GLS) [124, 128, 127] is another, relatively new, general metaheuristic algorithm. It is a higher level procedure that can sit on top of other heuristic methods to guide them to escape locally optimum solutions. One key concept in GLS is to replace the objective function used by LS with an augmented objective function which includes penalties associated to features of the candidate solutions. When LS is trapped in a local optimum, GLS is dynamically modifying the augmented objective function by selectively increasing penalties for features present in this local optimum. This causes LS to escape by forcing it to move towards neighbour solutions that exclude these features.

Different metaheuristics apply different strategies to improve search and overcome the problem of local optima as shown in Table 3.1. SA and GA employ a randomness component in a probability-based and population-based schema respectively to modify the movement strategy. TS exploits historical information in a memory-based approach to modify the set of neighbour solutions (i.e. neighbourhood). Like TS, GLS employs

Table 3.1: GLS versus other metaheuristics

| Metaheuristic | High-level Strategy | Basic Component | Tuning Parameters |
|---|---|---|---|
| **GLS** | Using a penalty-based approach to dynamically modify the cost function | Features, Features' cost | Lambda or Lambda Coefficient [89] |
| **TS** | Using a memory-based approach to dynamically change the neighbourhood method | Solution Attributes, Procedure for Tabu list | Tabu list procedure, Tabu list size, [Aspiration threshold] |
| **SA** | Using a probabilistic-based approach to allow none-improvement moves | Annealing Schedule | Initial Temperature, Cooling rate |
| **GA** | Using a population-based approach that includes a mutation operator to involve (partially) random solutions | Genetic operators: reproduction, crossover and mutation, Replacement strategy | Population size, Crossover rate, Mutation rate |

search history. In GLS, history is captured in penalties which dynamically modify the objective function.

GLS has been successfully applied to a wide range of problems and has achieved state-of-the-art results in a number of benchmarks. The problem domains of GLS include routing [125, 123], task assignment [75, 89], resource scheduling [114], constraint optimization [88]. A recent survey of GLS and its applications is presented in [128, 129].

### 3.1.1 Statement of Algorithm

GLS applies a penalty-based approach that can be superimposed on a LS algorithm with the aim of guiding it to escape local optima. The objective function is augmented with penalties which are increased when LS settles in a local optimum. Penalties are

associated to solution features. A key component in GLS is the definition of solution features. Solution features are defined to distinguish between solutions with different characteristics. GLS aims to penalize poor characteristics that hopefully to be removed by LS. Features are problem-dependant and can be derived directly from the objective function which usually comprises of one or more features. An indicator is used to determine whether a solution exhibits a feature. GLS associates a cost and a penalty to each feature. The costs can often be defined by taking the terms and their coefficients from the objective function. For example, in the Travelling Salesman Problem (TSP) [52], the objective is to find a route that visits a set of cities in a short total travelling distances. the defined feature can be *"whether the candidate tour travels immediately from city X to city Y"*. The set of features then includes all possible links between any two cities. The cost of each feature is the distance of the associated link. The penalty of a feature is initialized to 0 and will be incremented every time LS settles on a local optimum and the feature is nominated to be penalized.

When features and costs are defined, GLS defines a function $h$ that will be used by LS (replacing $g$):

$$h(s) = g(s) + \lambda \times \sum i \in F(p_i \times I_i(s)) \tag{3.1}$$

where $s$ is a candidate solution, $g$ an objective function that maps every candidate solution $s$ to a numerical value, $\lambda$ is a parameter to the GLS algorithm, $i$ ranges over the features in $F$, $p_i$ is the penalty for feature $i$ (all $p_i$'s are initialized to 0) and $I_i$ is an indication of whether $s$ exhibits feature $i$:

$$I_i(s) = 1 \text{ if } s \text{ exhibits feature } i\text{; 0 otherwise.} \tag{3.2}$$

The basic procedure of GLS can be described as follows, (Figure 3.1): starting from an initial solution, a local search algorithm is applied until it reaches a local optimum. GLS augments the cost function by adding penalties to selected features. Then, the

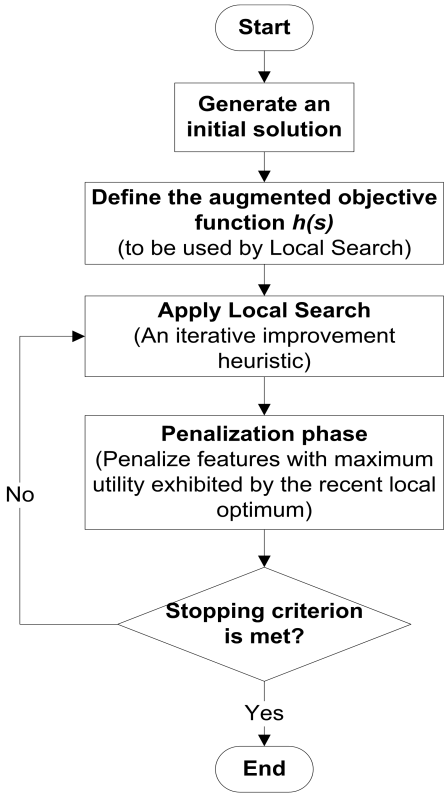



Figure 3.1: The Guided Local Search Procedure

local search restarts from the same local optimum using the updated objective function.

The novelty of GLS is mainly in the way that it selects which features to penalize. The intention is to penalize unfavourable features or features that "matter most" when a local search settles in a local optimum. The feature that has high cost affects the overall cost more. Another factor that should be considered is the current penalty value of that feature (i.e. the frequency of penalizing a feature). The utility of penalizing feature $i$ ($util_i$) under a local optimum $s^*$, is defined as follows:

$$util_i(s^*) = I_i(s^*) \times \frac{c_i}{1 + p_i} \tag{3.3}$$

where $c_i$ is the cost and $p_i$ the current penalty value of feature $i$. In other words, if

a feature is not exhibited in the local optimum (indicated by $I_i$), then the utility of penalizing it is 0. The higher the cost of this feature (the greater $c_i$), the greater the utility of penalizing it. Besides, the more times that it has been penalized (the greater $p_i$), the lower the utility of penalizing it again. At a local optimum, the feature with the greatest *util* value will be penalized. When a feature is penalized, its penalty value is always increased by 1. The scaling of the penalty is adjusted by $\lambda$.

The lambda parameter ($\lambda$) is the only parameter to GLS. It has been observed that, for several problems, lambda can be calculated as a function of a local optimum and the average number of features present [125, 89]. In these problems, lambda is dynamically computed after the first local optimum and before penalties are applied to features for the first time. Providing an $\alpha$ parameter (called "Lambda Coefficient"), which is relatively instance independent, lambda is calculated by the following formula:

$$\lambda = \alpha * g(s^*)/F_{s^*} \tag{3.4}$$

where $g$ is the objective function of the problem, $s^*$ a local optimum and $F_{s^*}$ the number of features present in $s^*$. Tuning alpha can result in lambda values, which work for many instances of a problem class.

### 3.1.2 Applications of GLS to Field Workforce Scheduling

The most significant results of GLS are probably in the routing and scheduling domain. In the routing domain, GLS achieves an outstanding performance in the well-known TSP [125] and vehicle routing problems [69]. This motivates researchers to apply GLS or a hybrid of GLS with other metaheuristics to other variants of routing problems, see for example [137, 108, 85].

GLS has also been successfully applied to many assignment problems. In [75], the Guided Genetic Algorithm (GGA) was proposed to solve the GAP, obtaining very

competitive results to the state-of-the-art algorithm. GGA demonstrated the ability of GLS to sit on top of GA. GLS and other GLS hybrids have been proposed to the related quadratic assignment problem [89, 135, 54] and the multidimensional knapsack problem [57].

In workforce scheduling, Tsang and Voudouris [114] applied GLS to the BT field workforce scheduling problem. In their algorithm, GLS holds the best-published results in a public benchmark problem [7] provided by BT's laboratory.

## 3.2 Multi-objective Optimization

### 3.2.1 Concepts and Definitions

Most real-world optimization problems are multi-objective in nature. The multi-objective optimization problem (MOOP) concerns the optimization of two or more objectives simultaneously. Instead of searching for a global optimum solution as in single-objective optimization problems, the search in MOOPs targets a set of solutions representing the optimum set of trade-offs between the objectives. This set is known interchangeably as the Pareto optimum set or the efficient solutions, and the objective values of these solutions are located at the Pareto front (PF). Efficient solutions are non-dominant solutions in the sense that improving the value of any one of their objectives must be at the expense of degrading the quality of one or more of the other objectives. Thus, all efficient solutions are considered equivalent as long as there is no further information regarding the relative importance of each of the objectives.

Before going further into this topic, the following terms and concepts need to be defined:

**Definition 1** (Pareto Dominance) *In a multi-objective optimization, a solution $x$ dominates a solution $y$ if and only if $x$ is no worse than $y$ in all objectives and $x$ is strictly better than $y$ in at least one objective.*

**Definition 2** (Efficient Solution) *A solution that is non-dominated by any other solution in the search space is called an efficient solution.*

**Definition 3** (Pareto Front (PF)) *This refers to the image of the set of all efficient solutions in the objective space, forming the best trade-offs among the objectives.*

**Definition 4** (Pareto Local Optimum) *A solution $s$ is a Pareto local optimum when it is non-dominated by another solution in its local neighbourhood.*

**Definition 5** (Pareto Local Optimum Set [95]) *A Pareto local optimum set is a set of non-dominant solutions, and each solution is a Pareto local optimum with respect to its neighbourhood.*

**Definition 6** (Ideal point) *The point $z^I$ in the objective space is called an ideal point if it dominates all efficient solutions.*

**Definition 7** (Nadir point) *The point $z^N$ in the objective space is called a nadir point if it is dominated by all efficient solutions.*

### 3.2.2 Performance Measures

Due to the nature of multi-objective optimization problems, the quality of an approximation to the Pareto front is evaluated by measuring both the convergence towards the true Pareto front and the even spread over the whole front [139]. Thus, multiple performance indices should be used for comparing the performances of different algorithms. Major performance indices which are widely used in literature to measure the convergence and spread of an approximation are defined as follows:

**Definition 8** (Set Coverage (C-metric)) *Let $X$ and $Y$ be two approximations to the PF of a MOOP, $C(X,Y)$ is defined as the percentage of the solutions in $Y$ that are dominated by at least one solution in $X$. When $C(X,Y) = 1$, all solutions in $Y$ are*

*dominated by some solutions in $X$. On the other hand, $C(X,Y) = 0$ means that all solutions in $Y$ are non-dominated by any solution in $X$. $C(X,Y)$ is not necessarily equal to $1 - C(Y,X)$. This measure evaluates the convergence of an approximation towards the PF, but not necessarily the even distribution over the PF.*

**Definition 9** (Distance from Representatives in the PF (D-metric) [121]) *Let $P^*$ be a set of uniformly distributed points along the PF, or an upper approximation of the PF. Let $A$ be an approximation to the PF, the average distance from $P^*$ to $A$ is defined as follow:*

$$D(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \tag{3.5}$$

*where $d(v, A)$ is the minimum Euclidean distance between $v$ and the points in $A$. The lower the value of $D(A, P^*)$, the closer $A$ to the PF will be. D-metric can be used to evaluate both the convergence as well as the coverage of the whole front.*

**Definition 10** (The $R$ measure [64]) *This is a unary index that evaluates an approximation to the PF by the mean of the best value of the weighted Tchebycheff utility function over a set of normalized weight vectors. The $R$ measure is normalized between 0 and 1, and the higher this value, the better the approximation will be. This measure evaluates both the convergence and the even spread of an approximation. It assumes the ideal ($z^I$) and nadir ($z^N$) points for the objectives are known.*

**Definition 11** (The hypervolume $H$ [138]) *This is a unary index that approximates the volume included under the curve formed by the non-dominant set in the objective space. It measures both the convergence and the distribution of the approximation set. The higher this value, the better the approximation will be. The $H$ measure requires only the nadir point ($z^N$) to be known in advance.*

### 3.2.3 Metaheuristics for Multi-objective Optimization

Metaheuristics [48] represent an optimization scheme that is characterized by its ease of implementation and its ability to obtain good performance in comparatively less computational time than classical optimization methods (i.e. exact methods). These features encourage the application of metaheuristics to multi-objective combinatorial optimization problems [118]. Vast amounts of research on the development of multi-objective metaheuristics (MOMH) has been conducted during the last three decades. The pioneers of such research include the work of Schaffer in 1984 [100] and 1992's algorithm of Serafini [101]. There are a number of surveys on this topic, e.g. [40, 94]. The majority of the proposed MOMHs are population-based algorithms, in particular multi-objective evolutionary algorithms (MOEAs) [67]. The popularity of MOEA techniques is attributed to their population-based nature which enables the finding of multiple optima simultaneously. Single-point-based (i.e. local search based) metaheuristics, on the other hand, have also been proposed to tackle MOOPs, however, to a lesser extent. The majority of such algorithms are based on local search, TS and SA. Here, we will briefly review pioneering research in MOMHs, with a great emphasis on the local search based MOMHs.

The task of MOMHs is to find a set of solutions that approximates the whole PF. Thus, most MOMH algorithms maintain an 'archive' that stores the non-dominant solutions discovered during the search. In order to enhance the performance in terms of convergence to, and spread over, the PF, different algorithms employ different fitness assignment approaches to compare different solutions in the objective space. There are two common approaches, namely Pareto dominance and aggregation of objectives. Discussions on these two approaches are given in [32, 59].

Aggregation-based MOMHs assign a single fitness value to each solution using a single objective function; the latter is constructed by combining all objectives using a weighted method. In this case, multiple objectives are mapped to a single objective,

which allows applying single-objective metaheuristics to MOOPs in an easy manner. This approach is based on the argument that a Pareto optimum solution to a MOOP, under mild conditions, could be an optimal solution of a scalar (i.e. an aggregation of all objectives using a weight vector) optimization problem. Therefore, in order to find a well-distributed set of solutions that approximates the PF, some strategies should be used to maintain a set of scalar functions that represents the whole PF. Several aggregation methods can be found in the literature (e.g., [86]), the most popular ones among which include the weighted-sum approach and Tchebycheff. The issue with these methods is how to choose (a uniform set of) weights in advance.

On the other hand, Pareto-based algorithms rank solutions by comparing them using the concept of Pareto dominance. However, domination does not define a complete ordering among the solutions in the objective space. To maintain a diverse set of non-dominant solutions that represents the whole PF, these algorithms employ an additional strategy to estimate the density of non-dominant solutions. Several of such strategies have been employed by Pareto-based MOMHs, the pioneers of which were in the MOEAs literature such as [71, 140, 36]. Two examples of such strategies are *crowding distance* [36] and *adaptive grid*[71], both of which are widely used in the literature. They are defined as follows:

**Definition 12** (The Crowding Distance [36]) *This estimates the density of solutions surrounding the individual in the population by calculating the average distance of two points on either side of the individual along each of the objectives. The crowding distance of a solution with the best value in any objective is set to infinity. The idea is that when two solutions non-dominate each other, the one residing in less crowded region (i.e. has large crowding distance value) is more favourable.*

**Definition 13** (The Adaptive Grid [71]) *This is a crowding procedure that recursively divides the objective space. Each solution is placed in a certain grid location based on*

*the values of its objectives. The number of solutions that reside in each grid location determines the density of this grid. Again, the idea is that when two solutions non-dominate each other, the one that resides in less crowded region (i.e. located in a less crowded grid location) is more favourable.*

#### 3.2.3.1 Local Search based MOMHs

A simple, intuitive adaptation of local search, to contain multiple objectives, can be described as follows: for each weight vector from a given set of weights, a single-objective local search algorithm starts from a randomly/heuristically generated solution, and iteratively improves this solution using the correspondence scalar function. Each local search stops at a local optimum solution with respect to the maintained scalar function. The obtained local optima could be an approximation of the PF. This idea has been developed further in [17, 93]. For example, the two-phase local search (TPLS) [93] considers the weight vectors in a sequential manner. Therefore, the current scalar objective is arguably the closest to the previous one (i.e. both have quiet similar weight vector), and thus the local optimum obtained in the previous scalar problem is used as a starting point for the next problem.

A simpler multi-objective local search algorithm is to apply the Pareto domination as an acceptance criterion when comparing the current solution to the new one. An archive of non-dominant solutions discovered during the search is maintained in order to produce an approximation to the PF. The algorithm stops when the neighbourhood of all solutions in the archive have been explored, i.e. the archive is a Pareto local optimum set (Definition 5). Recently, this idea was termed Pareto Local Search (PLS) and has been developed and extended by several researchers such as in [3, 95]. Due to its high relevance to our research presented in this thesis, PLS is thoroughly discussed and its related literature is reviewed later in this chapter (section 3.2.4).

Local search based MOMHs refer to the extensions of single-objective metaheuristics

(SOMHs) that employ the local search as the core search engine to contain multiple objectives. Since the local search is the basis of such metaheuristics, they are referred to sometimes as neighbourhood-based MOMHs [40]. Several local search based MOMHs have been proposed in the literature [94], the majority of which employ TS and SA. Algorithms vary in their searching strategies, particularly the applied high level approach to help the underlying local search algorithm to overcome the problem of getting trapped at Pareto local optima. An overview of the pioneering works in this topic is provided as follows.

The first TS based method for MOO was introduced in 1997 by Gandibleux et al. [45]. It is an aggregation-based approach where a series of tabu processes is performed, each process approximates a part of the PF corresponds to the optimized scalar objective. The method guides the local search process to escape local optima with respect to the current scalar objective by allowing moves to inferior solutions guided by a tabu list on the decision space and another list in the objective space. Another TS based MOMH was proposed by Hansen [55]. The method incorporates ideas form population-based methods. It starts with a set of solutions which are optimized simultaneously. Each solution represents a separate tabu process that maintains its own tabu list and is guided in the objective space by a scalar objective. At each step, tabu processes perform a single move in a sequential manner. A key feature of this method is the way it continuously updates the weight vector of scalar objectives for tabu processes. The method incorporates knowledge from the archive to calculate the new weights and then steer the search of the current tabu process towards promising parts of the PF. On this direction, other implementations of TS based MOMHs have been developed afterwards, e.g. [68].

On the other hand, there have been a number of Pareto-based implementations of TS for MOOPs, first of which were proposed by Baykasoglu et al. [13] and Ben Abdelaziz et al. [14]. The latter is a single-point-based algorithm in which the neighbours of

the current solution are enumerated and those non-dominated by the current solution become candidates for addition to the maintained archive. A move from the current solution is performed to any mutually non-dominant neighbour that is non-tabu. A diversification strategy is implemented periodically to guide the search to explore other areas of the search space. Other TS based algorithms that follow this approach have been developed in [62, 4].

SA is another popular local search based metaheuristic that has been studied in the context of multi-objective optimization [106]. The key component of SA is the definition of the energy measure that is used as an acceptance criterion. SA based methods that have been developed vary in the presentation of the energy function. Similar to TS, the first generation of such SA based methods contains the multiple objective by implementing a weighted-sum approach in the energy measure. Works in this stream include methods developed in [101, 119, 34, 105, 79, 98], the pioneering algorithm of which is introduced by Serafini [101]. The idea of this single-point-based method is to optimize one weighted scalar function at each step. To find well-distributed non-dominant solutions, the weight vector is modified slightly and randomly during the search. The second generation of SA based MOMHs is characterized by deploying the notion of the Pareto dominance into the definition of the energy measures. Such algorithms include those proposed in [104, 106, 10]. In [104], for example, the energy measure was formulated to consider the amount of domination between two solutions, when comparing a new solution to the current one.

Attempts to extend other local search based metaheuristics are still modest. Adaptations of the Greedy Randomized Adaptive Search algorithm (GRASP) are proposed in [5, 80]. An iterated local search based multi-objective algorithm can be found in [47].

#### 3.2.3.2 Population-based MOMHs

Instead of managing a single solution, the searching behaviour of population-based algorithms relies on maintaining a population of solutions. In order to find an optimum approximation of the PF, such algorithms start from an initial population and iteratively generate new offspring using principles such as independent evolution and cooperation between individuals. Independent evolution principles include genetic operators such as mutation and crossover. These are employed by MOEAs, which constitute the greater part of the literature of MOMHs. The first population-based MOMHs was introduced by Schaffer in 1984 [100], who proposed the Vector Evaluated Genetic Algorithm (VEGA). Since then, extensive research on the development of MOEAs has been done, a review of this research is provided in [28]. The fitness assignment strategy has been the main issue of the research on MOEAs. While VEGA proposed alternating objective-based fitness assignment, more effective alternatives have been developed, including aggregation of objectives (such as Multi-objective Genetic Local Search (MOGLS) [64, 61] and Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [136]) and Pareto-based ranking (such as Non-dominated Sorting Genetic Algorithm II (NSGA2)[36] and Strengthen Pareto Evolutionary Algorithm II(SPEA2)[138]).

MOEA/D [136] is a representative aggregation-based MOEA that employs GA. It optimizes multiple scalar functions simultaneously, each of which is considered as a sub-problem. Each sub-problem is associated with one solution which is the best-so-far found solution with respect to the correspondence scalar function. The key feature of MOEA/D is that these sub-problems cooperate with each other during the search. Each sub-problem produces new offspring by using solutions from 'neighbour' sub-problems with quite similar scalar functions.

On the other hand, NSGA2 [36] is a representative Pareto-based MOEA. The novelty of NSGA2 is mainly in the procedure of ranking a population. It ranks its popula-

tion based on two values: (1) a non-domination level of an individual in the population, and (2) an individual's crowding distance (Definition 12) value which estimates the density of solutions surrounding the individual in the population. The idea is that non-dominant solutions are preferred over dominated ones, and if two solutions have the same non-domination rank, the one resides in less crowded region is more favourable.

### 3.2.4 Pareto Local Search

Quite recently, the Pareto Local Search (PLS) method has been proposed as a very simple algorithm for solving MOOPs. Its simplicity stems from the fact that PLS is a straightforward extension to local search algorithms for MOOPs, no parameter setting is required, and it does not involve any aggregation of objectives. This is achieved by incorporating the concept of archiving where a set of potentially efficient solutions discovered during the search is maintained. The basic idea of PLS is to iteratively improve this set by exploring the neighbourhood of its solutions. The acceptance criterion depends on the notion of Pareto dominance. In addition, PLS has a natural stopping condition that occurs when the neighbourhoods of all solutions in the archive have been explored.

```
Pareto Local Search([g_1, .., g_k])
  s_0 ← InitialSolution();
  archive ← s_0
  while ∃s ∈ archive such that Visited(s) = false do
    for all s' ∈ Neighbourhood(s) do
      Evaluate(s', [g_1, .., g_k])
      if s' is non-dominated by any solution in archive then
        UpdateArchive(s')
      end if
    end for
    Visited(s) = true
  end while
  return archive
```

**Algorithm 3.1:** Pseudo-code for Steepest Pareto Local Seach (SteepPLS)

Such features have motivated researchers to study and develop this idea, and therefore several variants of PLS have been proposed in the last decade [94]. In [95], Paquete and Stützle proposed a PLS method which works as follows (Algorithm 3.1): an archive is initialized with an initial solution. Then, a solution from the archive is chosen randomly and its neighbourhood is fully explored. The archive is updated with the new neighbours. After that, the current solution is marked as 'visited' so as not to be selected again. The PLS here has a natural stopping condition, that is when all solutions in the archive are examined (i.e. marked as 'visited'). A major characteristic of this PLS is that it applies a steepest neighbourhood visiting approach in which the neighbourhood of the current solution is fully explored; thus, we call this version here and onward SteepPLS.

Angel et al. [3] proposed another variant of PLS which is closely related to SteepPLS. The only difference is in the selection scheme of the current solution. Instead of using a single current solution as in SteepPLS, all non-explored solutions in the archive forms *the current population* in Angel's PLS. The neighbourhood of every member of the current population is fully explored, and the archive is updated accordingly. This means that the neighbourhood of every solution added to the archive is examined even if it is found to be dominated by a neighbour of another solution in the current population.

Another PLS variant is the Pareto Archived Evolution Strategy (PAES) which was proposed by Knowles and Corne [71]. PAES employs an advanced, more complex acceptance criterion when comparing a new neighbour to the current solution. Beside the Pareto domination, the fitness of a solution incorporates a density estimation measure, that is the adaptive grid (Definition 13). Starting from a random solution, PAES iteratively mutates the current solution to generate a neighbour. A neighbour is accepted to replace the current solution if it dominates the latter, or if it is non-dominated by the current solution, but resides in a less crowded region in the current approximation to

the PF. In this way, PAES differs than the aforementioned PLS variants in the following components:

1. The selection scheme that chooses the current solutions from the archive to explore their neighbourhoods. PAES is a pure single-point-based algorithm and the next explored solution can only be a neighbour to the current one. Thus, the use of solutions in the archive is made only by the adaptive grid measure. The other PLS variants, on the other hand, select the next examined solution(s) from the archive independently after exploring the neighbourhood of the current examined solution(s).

2. The neighbourhood exploration strategy. PAES applies a partial neighbourhood visiting strategy by which only some (i.e. one in the case of PAES) neighbours to the current solution are examined. The other PLS variants fully explore the neighbourhood of the current solution.

3. The acceptance criterion when comparing different solutions. While PLS variants depend merely on the notion of Pareto dominance, PAES employs a density estimation measure as a second factor.

4. The stopping condition. In the absence of a computational time limit, unlike the other PLS variants, PAES has no explicit termination condition.

Enumerating and identifying these components would help in comparing and contrasting different PLS variants. On this direction, Liefooghe et al. [81] defined a unified view of PLS variants by identifying nine basic components that can be used to characterize different PLS variants.

There are scenarios where PLS (and any multi-objective algorithm) has to maintain a bounded archive. When the archive exceeds a size limit, a procedure is applied to choose which solutions to remove. The task is to take out the solutions with the least

effects on the quality of the approximation to the PF. Several procedures have been developed to address this issue, the basic idea of most of which is to remove solutions that reside in crowded areas in the approximation. Examples of such procedures include the crowding distance (Definition 12), the adaptive grid (Definition 13), and clustering algorithms [63] such as single linkage and average linkage. In order to minimize excessive calls to such clustering procedures and enhance the quality of the archive, the idea of soft and hard size limit can be applied [10]. The size of the archive is allowed to exceed the hard limit up to a soft limit. Every time the soft limit is reached, the clustering procedure is applied to reduce the size of the archive to the hard limit.

The applications of PLS have shown its ability to obtain a good approximation of the PF as a standalone technique [95, 3]. However, a major drawback of the PLS is the slow convergence that makes its computational time much higher than other state-of-the-art techniques [95, 83]. Besides, one can anticipate that PLS, as a local search method, can also be trapped in local optima points. This would become more serious when a simple neighbourhood function is defined, or a partial neighbourhood visiting strategy is used. To overcome such limitations, Lust and Teghem [83] have shown that the performance of PLS can be improved when it is coupled with an efficient initial population generator. They have proposed a two-phase algorithm, the first phase finds a good approximation of the PF by solving weighted-sum single-objective optimization problems, and the second phase applies a PLS that starts by adding the obtained solution to the archive. Another multi-phase algorithm has been introduced by Jaszkiewicz and Zielniewicz [66]. In their algorithm, the first phase employs an aggregation-based GA to find a good approximation for the PLS (i.e. the second phase). Another approach towards enhancing the performance of PLS is to combine it with another evolutionary algorithm (i.e. a global optimizer) forming a hybrid algorithm. This has been shown in [12, 73] where hybrids of PLS and GA are introduced.

### 3.2.5 GLS for Solving Multi-objective Optimization

To our knowledge, there has been no attempt to adapt GLS for tackling combinatorial optimization with multiple objectives. GLS is a simple, general algorithm with few parameters to tune. It has been shown that GLS can sit on top of local search algorithms as well as other metaheuristics. Similarly, GLS has the potential to sit on top of PLS with the hope of guiding PLS to escape Pareto local optimum sets. The definition of the penalization scheme, as well as features and their costs, can be extended to contain multiple objectives, and thus PLS is guided towards more promising areas of the solution space that would enhance the quality of the approximation of the PF. This describes a research objective of this thesis, which is discussed in details in chapter 6.

## 3.3 Conclusions

GLS is a local search based metaheuristic that is characterized by its generality and simplicity. It is a very successful algorithm with a wide range of application domains, including routing and scheduling optimization problems. There is only one parameter to tune, and luckily GLS is found in many cases not to be sensitive to this parameter. The GLS approach has been extended to be superimposed on other heuristics and metaheuristics. Nevertheless, GLS has not been adapted to tackle multi-objective optimization problems. The latter are of a complex nature in which the task is to find a set of solutions that forms the best trade-offs (i.e. the PF) between different objectives. The literature is rich with efficient heuristics and metaheuristics that are designed to produce a good approximation of the PF. Among these algorithms, PLS is a simple extension to the single-objective local search that employs the notion of Pareto optimality for solving MOOPs. It can be a standalone solution approach or a central component in multi-phase or hybrid algorithms. Similar to single-objective local search algorithms, PLS suffers from the problem of settling at (Pareto) local optima. Metaheuristics such

as TS, SA and GLS can help the PLS to avoid or escape this situation. Guided Pareto Local Search (to be introduced in chapter 6) proves this by confirming the ability of GLS to sit on top of PLS and guide it to escape Pareto local optimum sets.

# Part III

# Thesis Contributions

# Chapter 4

# Field Workforce Scheduling: A Computational Study

A central aim of our research is to develop a new flexible model for FWS. In order to pave the road in this direction, the first research step is the study of a representative case of FWS problems, which is a three-fold task. The first is to formulate a representative, traditional (i.e. inflexible) FWS problem, which will be the case study of this thesis. The second is to develop benchmarks using a problem (instance) generator, for further scientific studies on FWS, in particular the required computational studied in this thesis. The third task concerns the solution techniques for the FWS problem. FWS can be treated as an assignment problem or as a routing problem, as it involves characteristics from both classes of problem. Hence, an attempt is made here to systematically examine the performance difference between representative assignment and routing heuristics, under various problem characteristics. The hope is that varying problem characteristics will help to explain the performance differences between the two solution approaches.

This chapter describes these objectives in detail. The FWS problem is formulated in section 4.1, and then the details of the problem generator are given in section 4.2.

Section 4.3 discusses the solution approaches that are to be examined. The design of the computational experiments and their results are discussed in section 4.4. The conclusions are given in section 4.5.

## 4.1 Problem Formulation

The problem formulation presented here is motivated by a real FWS problem, one that appears in a leading telecommunication company [16, 120].

FWS is basically the problem of allocating a set of technicians (resources): $R = \{r_1, r_2, ..., r_{|R|}\}$, to a set of tasks: $T = \{t_1, ..., t_{|T|}\}$. A task $t$ is described by a tuple:

$$< c_t, dur_t, reqSkill_t, [start_t, end_t], loc_t >$$

Where $c_t$ is a predefined priority which determines its importance to the company. The higher the value of $c_t$, the more important the task will be, and $c_t \in \mathbb{R}^+$. $dur_t$ is the expected duration a technician requires to finish this task. Each task requires a technician with a particular skill $reqSkill_t \in SkillSet$, where $SkillSet$ is the set of all skills: $SkillSet = \{skill_1, ..., skill_{|SkillSet|}\}$. A task $t$ must be serviced within a predefined discrete time-window described by $[start_t, end_t]$. Tasks are geographically distributed, and the location of a task is denoted by $loc_t$.

Each technician $r \in R$ is described by a tuple:

$$< [start_r, end_r], skills_r, loc_r >$$

Each technician has limited shift hours where the beginning and end of the shift are expressed by $[start_r, end_r]$. $skills_r$ denotes the skill(s) a technician has, where $skills_r \subseteq SkillSet$. Each technician has a base location where (s)he starts the working day. Technicians vary in their base location as they can start from home or a predefined

depot. The base location of a technician is denoted by $loc_r$.

For simplicity, without loss of generality, the travelling time between any two locations ($trvl_{loc_1 loc_2}$) is calculated as the Euclidean distance divided by speed ($v$).

There are two sets of decision variables in FWS: the allocation variables $X = \{x_{rt} | r \in R; t \in T; x_{rt} \in \{0,1\}\}$ and the service times $ServTime = \{st_t | t \in T\}$. A variable $x_{rt}$ is set to 1 if technician $r$ is allocated to task $t$, and 0 otherwise. A variable $st_t$ denotes the start time of the service for task $t$.

Having decided these variables, a set of routes $\pi = \{\pi_r | r \in R\}$ are defined. A route $\pi_r$ is a sequence of tasks ($\subseteq T$) that are to be visited by the technician $r$; $\pi_r = (\pi_{r1}, \cdots, \pi_{r|\pi_r|}), 0 \leq |\pi_r| \leq |T|$.

FWS consists of various and often conflicting objectives. For the scope of this thesis, we focus on perhaps the main objective, namely maximizing the productivity rate which is expressed in terms of the number of allocated tasks with respect to their priorities, i.e. minimizing the expected cost of unscheduled tasks. Therefore, the goal is to find an assignment of resources to tasks that maximizes the importance of the tasks scheduled, while satisfying all assignment and routing constraints.

Another objective, that can be considered as an important organizational interest in FWS problems, is the minimization of the travelling cost [114, 33]. This objective is not explicitly captured in our formulation. However, the travelling distance is still somehow considered in the optimization process, given that minimizing the travelling time would create more available time to serve other tasks, and then increase the productivity rate.

The FWS problem can, then, be mathematically modelled as follows:

$$max \qquad \frac{\sum_{r \in R} \sum_{t \in T} c_t x_{rt}}{\sum_{t \in T} c_t} \tag{4.1}$$

subject to

$$\sum_{r \in R} x_{rt} \leqslant 1 \quad \forall t \in T \tag{4.2}$$

$$reqSkill_t \in skills_r \quad \forall x_{rt} = 1, t \in T, r \in R \tag{4.3}$$

$$start_t \leqslant st_t \quad \forall t \in T \tag{4.4}$$

$$st_t + dur_t \leqslant end_t \quad \forall t \in T \tag{4.5}$$

$$start_r \leqslant st_{\pi_{ri}} \quad \forall r \in R, i = \{1..|\pi_r|\} \tag{4.6}$$

$$st_{\pi_{ri}} + dur_{\pi_{ri}} \leqslant end_r \quad \forall r \in R, i = \{1..|\pi_r|\} \tag{4.7}$$

$$start_r + trvl_{loc_r loc_{\pi_{r1}}} \leqslant st_{\pi_{r1}} \forall r \in R \tag{4.8}$$

$$st_{t_i} + dur_{t_i} + trvl_{loc_{t_i} loc_{t_{i+1}}} \leqslant st_{t_{i+1}} \quad \forall t_i \in \pi_r, \pi_r \in \pi \tag{4.9}$$

The objective function is represented by Equation 4.1 which is the sum of the costs of scheduled tasks, normalized by dividing it by the sum of the costs of all tasks. Equation 4.2 imposes that each task is visited once at most. The skill constraint is expressed in Equation 4.3. The time-window constraints of all tasks are assured by Equation 4.4 and Equation 4.5. Equation 4.6 and Equation 4.7 ensure that all tasks which are assigned to a technician must be within the technician's working time. Finally, Equation 4.8 and Equation 4.9 ensure route validity by considering the travelling time between a technician's base location and the first task, as well as between subsequent tasks in the technician's route.

## 4.2 Problem Generator

The nature of the computational experiments in this chapter, and throughout the thesis, requires the generation of several sets of instances with various characteristics. As pointed out in section 2.1.3.1, there is no public benchmark dataset for FWS yet, at least to our knowledge. Thus, we developed a problem generator which is partially

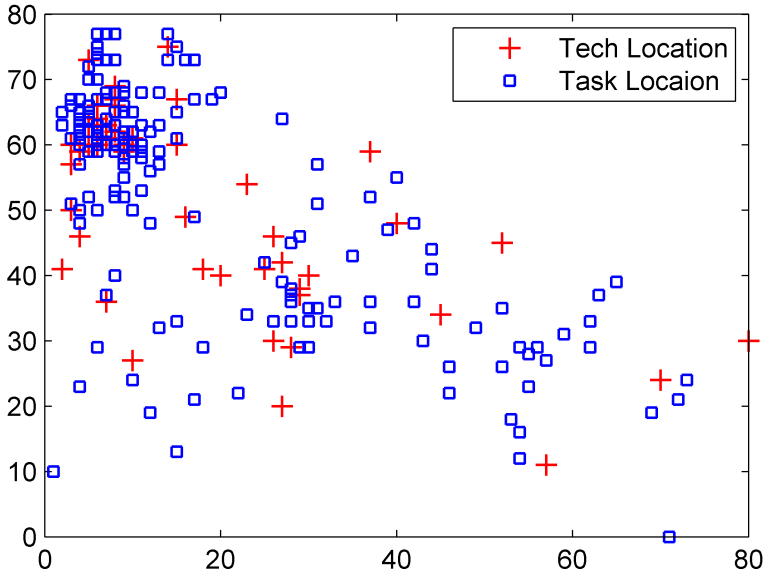



Figure 4.1: An example to illustrate the density of task and resource locations in a generated problem instance

inspired by a real FWS problem [16, 77].

Instances are constructed as follows: the duration of a day is measured in the number of minutes from midnight. A day typically starts at 480 (8:00 am) and ends at 1020 (5:00 pm).

The locations of tasks as well as the base locations of technicians are sampled uniformly at random from a sample of geographical coordinates. The geographical distribution of these coordinates is based on a large sample of real geographical coordinates (latitude and longitude) taken from a region in the UK. Each coordinate represents a postcode. The sample represents, therefore, the densities of houses in this region, as illustrated in Figure 4.1. Travel time (between any two coordinates) is measured as the Euclidean distance divided by speed ($v$). The default value of the speed is set to 15 km/h.

Each problem instance includes $|R|$ technicians and $|T|$ tasks per day. They are described as follows:

- Task:
  - There are $|T|$ tasks per instance. By default, each instance has 150 tasks.
  - Each task is assigned an estimated duration $dur$. The default is to sample the duration with a normal distribution with a mean of 120 minutes and a standard deviation of 30 minutes.
  - There are two main types of tasks: high priority (i.e. urgent) tasks, and appointment tasks (e.g. tasks that require a technician to visit a customer).
  - For high priority tasks, they represent 20% of the set of all tasks. They are associated with a cost of 40 (i.e. $c = 40$). Their time-windows start differently during the day, and must be served within 180 minutes.
  - For appointment tasks, they represent the other 80% of the set of all tasks. Appointment tasks are in turn divided into three types: (1) a whole day task which can be served any time during the day, (2) a morning task which should be finished by 13:00, and (3) an afternoon task which cannot be started before 13:00. A whole day task is given the lowest priority ($c = 10$); a morning and afternoon tasks both have the same level of priorities $c \in \{20,30\}$.
  - The percentage of appointment tasks (i.e. the 80% of tasks) are distributed as follows: 20% are whole day tasks and their time-window are set to $[480, 1020]$, 30% are morning tasks and their time-window are set to $[480, 780]$ and the remaining 30% of tasks are afternoon tasks and the time-window is $[780, 1020]$.
  - Each task requires a technician with a particular skill. There are 10 different skills ($|SkillSet| = 10$). The required skill of each task is sampled uniformly at random from $skill_1 \cdots skill_{10}$.
- Technician:

- There are $|R|$ technicians per instance. The default value of $|R|$ is 50.
- Each technician has a limited working hours (i.e. shift). By default, all technicians have a full-day shift, i.e. $[start_r, end_r] = [480, 1020]$.
- Each technician can have one or more skills. By default, the number of skills that technicians in the workforce have follows a binomial-like distribution (Table 4.1).
- The type of skills for each technician is sampled uniformly at random from $skill_1 \cdots skill_{10}$.

These describe the problem generator that is used in our research to generate benchmarks for computational experiments. The default values of the defined parameters are always applied unless it is explicitly stated.

Table 4.1: The probability distribution used to sample the number of skills ($|skills_r|$) for each technician

| $\|skills_r\|$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| probability | 2% | 5% | 8% | 15% | 20% | 20% | 15% | 8% | 5% | 2% |

## 4.3 Scheduling Algorithms

As reviewed in chapter 2, FWS can be seen as a combination of assignment and routing problems. It generalizes the well known GAP and VRP (refer to section 2.1.3). These two sub-problems are interdependent, which makes handling them separately as two different problems impractical. The literature on routing problems and that on assignment problems are rich with efficient optimization techniques, including exact and heuristic methods [20, 21, 96]. Since FWS involves characteristics from both classes of problems, predicting the performance differences between an assignment method and a routing method, both applied to FWS, is not trivial. In this study, the objective is

to study these performance differences. The study will concern local search methods, since local search is the basis of many state-of-the-art heuristics and metaheuristics developed for routing or assignment problems [20, 96]. Thus, a representative local search algorithm is chosen from the literature of assignment and that of routing problems. Both methods and their application to our formulation of the FWS problem are introduced below.

### 4.3.1 Local Search - An Assignment Approach ($LS_a$)

There have been many local search based algorithms developed for solving assignment problems. They vary in solution representation and move (i.e. neighbourhood) operators designed in such local search methods [96]. One possible solution representation is to represent a solution as a sequence (i.e. permutation) of jobs [114]. A sequence can be a heuristic or a random order of jobs, that undergoes a deterministic, heuristic scheduling procedure. To improve the solution quality, a move operator based on a small change (e.g. single swap) to the order of jobs is performed. Another approach is to associate a list of jobs to each agent. In this approach, move operators such as $\lambda$-interchange (i.e. swapping two subsets of customers from two different routes, such that the size of each set does not exceed $\lambda$) [92] and ejection chain approach [132] can be applied to improve the solution quality. This approach has been considered as a component to solve some routing problems, as well.

The former approach is chosen to be a representative of this class of heuristics in this study. This is due to its deterministic scheduling procedure which produces only feasible solutions, where the other move operators may produce infeasible solutions. Another motivation to adopt this approach is its success in obtaining high quality solutions in an example of a real FWS problem [114].

In order to apply this approach to the FWS problem, the basic components of the local search ($LS_a$) are defined as follows:

- Solution Representation: each candidate solution, i.e. state, in the search space is represented by a permutation of tasks. Such a permutation specifies the order of tasks to be considered in the scheduling process. The first task in the permutation is scheduled first, then the second one, and so on and so forth. Scheduling a particular task follows a deterministic procedure which can be summarized as "allocate this task to the nearest technician with respect to all constraints". At the beginning, a list of technicians is associated to each task, which involves those who are qualified to perform the associated task. The list will be sorted before the searching starts, and it remains static during the search. The sorting criterion is the distance from a technician's base location to the task's location.

- Neighbourhood Function: the Neighbourhood function is defined as any new permutation that may be obtained from the current permutation by performing a single swap between any two tasks.

- Cost Function: the cost (i.e. objective) function is defined in Equation 4.1.

- The initial solution: the initial solution is generated heuristically based on three rules:

    1. High priority tasks are scheduled first. This rule is very practical since the objective function takes into consideration the priorities of tasks ($c_t$).
    2. For two tasks $t_i, t_j$ with the same priority (i.e. $c_{t_i} = c_{t_j}$), if the number of available and qualified technicians for $t_i$ is less than that for $t_j$, then $t_i$ is scheduled first, otherwise $t_j$ (i.e. Smallest-domain-first heuristic).
    3. For two tasks $t_i, t_j$ with the same priority and domain size, the shorter in terms of the estimated duration $dur_t$ is scheduled first (i.e. greedy principle).

### 4.3.2 Local Search - A Routing Approach ($LS_r$)

Heuristic approaches developed for the VRP can be categorized into route construction, route improvement and methods that combine both route construction and route improvement [20, 21]. In route improvement (e.g. local search), several move operators have been proposed, including 2-Opt (deleting two arcs from a route, and replacing them with the only two arcs that reform the route) and $\lambda$-interchange [110].

An attempt is made here to adapt a local search that employs these routing-based move operators to be applied to the FWS problem. The issue is that these move operators have been designed for routing problems where the objective is to reduce the total travelling time, while the target objective of the FWS problem is to minimize the cost of unallocated tasks (i.e. maximize service productivity). An answer to this issue is that these two objectives are not completely in conflict with each other. This is attributed to the fact that the number of technicians (i.e. vehicles) is fixed, and there is no need to minimize this number. Indeed, reducing the travelling time by using special heuristics such as the 2-Opt, would still be useful for creating more availability time for a technician to serve more tasks. Therefore, minimizing the total travelling time is applied as an objective that guides the moves in the 2-Opt. Moreover, this objective is employed as a secondary helper objective to guide other move operators. That is when a neighbour solution is equal to the current solution with respect to the main objective (Equation 4.1), but it minimizes the total travelling time, the move to this neighbour is accepted.

The local search $LS_r$ is defined as follows:

- Solution Representation: a solution is represented by a set of lists of customer visits, each list corresponds to a particular technician and describes his/her route. A *virtual* route (i.e. technician) is introduced, to which all unallocated tasks are assigned. The constraints such as skill matching and time related constraints are

not imposed on this *virtual* route.

- Neighbourhood Function: two types of improvement heuristics are applied here: (1) the 2-Opt that operates within a route, and (2) $\lambda$-interchange that operates between routes. The value of $\lambda$ is set to one; thus, three move operators are performed: relocate $((1,0),(0,1))$ and swap $(1,1)$. The order of these move operators are $(0,1),(1,0)$, and then $(1,1)$. Every time a route is changed, the 2-Opt is applied to improve that route.

- Cost Function: the cost (i.e. objective) function is defined in Equation 4.1. In the 2-Opt, the *total travelling time* for the examined route is used as the objective function, since changing the order of tasks in a route has no implication on the FWS's main objective. The total travelling time is also used as a helper objective in the $\lambda$-interchange operator.

- The initial solution: an initial solution can be simply generated by assigning all tasks to the virtual tour. However, in this study $LS_r$ uses the same initial solution generated by the $LS_a$, as describe earlier. This allows us to have a fair comparison between the two types of improvement heuristics.

## 4.4 Computational Experiments

### 4.4.1 Problem Characteristics

Problem characterization refers to the process of identifying the properties of problem instances that are likely to distinguish between disparate sets of instances. Once the properties of a problem are properly understood, it is possible to explain the performance difference between different solution approaches applied to FWS.

To date, two main characteristics of problems were studied in relation to problem hardness, namely the size of the problem and the different level of constraints. Compu-

tational complexity, for example, studies how difficult it is to solve (in the worst case) a problem as a function of the instance's *size*. On the other hand, Cheeseman et al. [25] showed that for many NP-hard problems the level of constraints imposed by various parameters is correlated with problem hardness. In light of these studies, we identify three main characteristics which are likely to influence differently the performance of any applied solution approach. The characteristics are defined as follows:

1. **Problem size.** The size of the solutions space depends on the solution representation which depends, in the FWS context, on the number of tasks or technicians, or both of them. The problem size of FWS can be expressed as a function of the number of technicians in the workforce ($|R|$) and the amount of jobs received per day ($|T|$). It can also be expressed in terms of the technician-to-task ratio ($\sigma$), that is how many tasks on average for each technician per day.

2. **Tightness of skill constraints ($\tau$).** A task must be allocated to a qualified technician who possesses the required skill (Equation 4.3). This defines a major constraint in FWS. Recall that each technician can have one or more skills. The more skills a technician has, the more tasks, in principle, (s)he can perform, and consequently the less constrained the problem instance will be. Assuming that $\tau_i$ is the number of technicians who have the required skill to do task $t_i$, we measure the tightness of skill constraints ($\tau$) as follows:

$$\tau = \frac{\sum_{i \in T} \tau_i}{|T|} \tag{4.10}$$

3. **Resource utility ($\gamma$).** Another constraint relationship is that a technician cannot perform two different tasks at the same time (Equation 4.8). The impact of this constraint on the nature (or difficulty) of problem instances can be described in terms of resource utility. For FWS, the utility of a resource is defined as the

total time required by all tasks divided by the total available times of resources. When the resource utility increases (i.e. longer tasks' durations and travelling times), the number of tasks a technician can carry out in a day shift reduces, and thus the tightness of the constraint grows. The resource utility can then be measured as follows:

$$\gamma = \frac{\sum_{i \in T} (dur_i + avgTrvl)}{\sum_{j \in R} shift_r} \tag{4.11}$$

Where $avgTrvl$ is the average travel time a resource spends on travelling to a task location, and $shift$ the technician's shift length, i.e. 520 minutes as defined by the problem generator.

### 4.4.2 Experimental Design

The following experiments examine the performance of the two local search algorithms under different conditions. The hope is that the identified problem characteristics would distinguish between dissimilar sets of instances for the FWS problem, in which the efficiency of an algorithm might vary, and thus explain the performance difference between the solution techniques.

In general, it could be anticipated that the $LS_a$ approach outperforms $LS_r$ in instances that have similar characteristics to assignment problems such as tight skill matching constraints. As such characteristics change and instances involve more characteristics from the routing problem, the significance of the out-performance of the $LS_a$ approach decreases.

Our primary evaluation criterion in these experiments is $\Delta$, the ratio of the best cost found by the $LS_r$ approach ($\delta_r$) to that of the $LS_a$ approach ($\delta_a$):

$$\Delta = \delta_r / \delta_a \tag{4.12}$$

This ratio $\Delta$ will be equal to 1 if both approaches obtain the same solution quality. As $\Delta$ diminishes, the $LS_a$ significantly outperforms the $LS_r$. In contrast, when $\Delta$ exceeds 1, $LS_a$ is outperformed by the $LS_r$.

There is no resource limitation imposed upon both techniques, that is each local search stops when it reaches a local optimum.

In order to investigate the influence of the defined problem characteristics on both solution approaches, we use the problem generator described in section 4.2 while varying the following parameters:

- The problem size is controlled by a combination of two parameters, the workforce size ($|R|$) and the technician-to-task ratio $\sigma$. The values of these parameters are: $|R| \in \{30, 50, 70\}$, and $\sigma \in \{3, 4, 5\}$. For instance, when $|R| = 30$ and $\sigma = 3$, the number of tasks will be 90.

- The tightness of skill constraints, which is represented by the percentage of technicians who are qualified to do a particular task, is controlled by varying the parameter $\tau$ within the following domain: $\tau \in \{0.2, 0.4, 0.6, 0.8\}$. The tightest condition is represented by $\tau = 0.2$ where each task can be served by only 20% of the workforce (i.e. each technician has, on average, 2 out of 10 skills); whereas the loosest condition occurs when $\tau = 0.8$, in which 80% of the workforce have the required skill of a task.

- The resource utility describes the amount of resources (expressed in terms of task durations plus travelling distances) required by all tasks. This is defined by the parameter $\gamma$, such that $\gamma \in \{0.8, 1.0, 1.2, 1.3\}$. For instance, when $\gamma = 0.8$, the total amount of time required by all tasks is equal to 80% of the total available time for technicians, and thus serving each task consumes about $(0.8 * shift/\sigma)$ minutes.

Based on these characteristics, a set of instances is generated that contains each

possible combination of $|R|$, $\sigma$, $\tau$, and $\gamma$. First, an instance is generated for each combination of $|R|$ and $\sigma$. These two parameters determine the number of technicians and tasks per instance. From this instance, four instances are derived with different skill constraint levels ($\tau$). Finally, from each derived instance, four instances are obtained by varying the resource utility level ($\gamma$). These 144 instances represent a set of instances with different characteristics. In total, 10 sets are generated. For each instance, the $LS_r$ and $LS_a$ approaches are executed five times, and the mean of these runs are reported.

### 4.4.3 Experimental Results

#### 4.4.3.1 Problem size

Figure 4.2 presents the performance difference $\Delta$ with respect to the workforce size ($|R|$) and the technician-to-task ratio ($\sigma$). Overall, the figure shows that $\Delta$ grows as $|R|$ and $\sigma$ increase, which indicates the impact of increasing the problem size on the performance of $LS_a$, though it performs better than $LS_r$ (i.e. $\Delta < 1$) on all cases. As shown in Figure 4.2(a), $LS_a$ can produce 2.7% better results than $LS_r$ when $|R|$ is 30 (i.e. $\Delta = 0.973$). The difference diminishes to 1.4 % ($\Delta = 0.986$) as the $|R|$ grows to 70. On the other hand, Figure 4.2(b) shows that when $\sigma$ equals 3 (i.e. the technician-to-task ratio is 1:3), the value of $\Delta$ is 0.965 (i.e. $LS_a$ is 3.5% better than $LS_r$). When $\sigma$ grows to 5, the difference becomes less significant (i.e. $\Delta = 0.995$).

A reason for these results is that the solution space in $LS_a$ is defined as a function of $|T|$ which is expressed by $\sigma$. Increasing the number of tasks while maintaining a simple neighbourhood operator (i.e. single swap) is expected to have a negative impact on the performance of such a heuristic. On the other hand, $LS_r$ maintains multiple efficient neighbourhood operators that consider all resources (i.e. tours). Therefore, the influence of these operators becomes more significant as the the number of resources ($|R|$) grows.

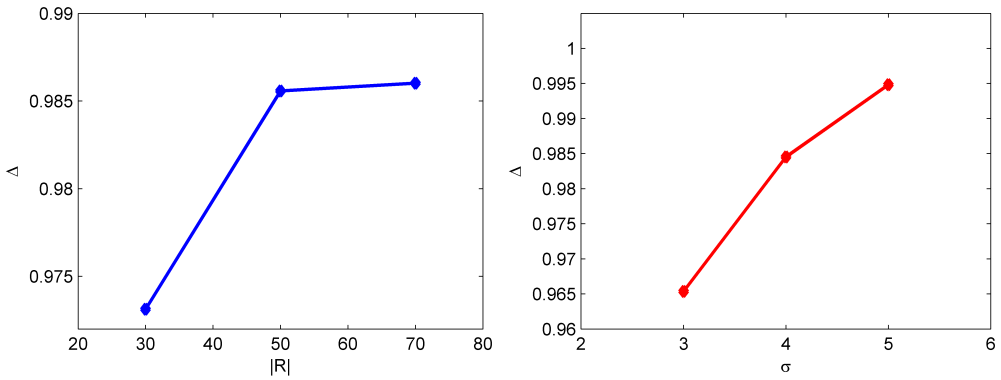



Figure 4.2: The relative performance difference ($\Delta$) between the two approaches with respect to: (a) $|R|$ and (b) $\sigma$.

#### 4.4.3.2 Constraint tightness

The impact of the skill constraint on the relative performance of both techniques is given in Figure 4.3. It depicts the sensitivity of $\Delta$ to the level of the skill constraint expressed in terms of $\tau$. The performance of the $LS_r$ approach declines (i.e. $\Delta$ decreases) as the percentage of technicians who are qualified to do a particular task shrinks. For instance, under relatively under-constrained conditions (i.e. $\tau = 0.8$), the performance difference between the two approaches is only 0.9% (i.e. $\Delta = 0.991$). As the constraint level increases (i.e. $\tau$ decreases), $LS_a$ becomes more effective. When $\tau$ reaches 0.2, $LS_a$ obtains results better by 3.5% (i.e. $\Delta = 0.965$).

This is attributed to the constraint-handling technique applied by both algorithms. While $LS_r$ rejects any infeasible solution, $LS_a$ applies a deterministic scheduling procedure that transforms any permutation of tasks into a *feasible* solution using a set of rules. The usefulness of this procedure increases on tightly constrained problems.

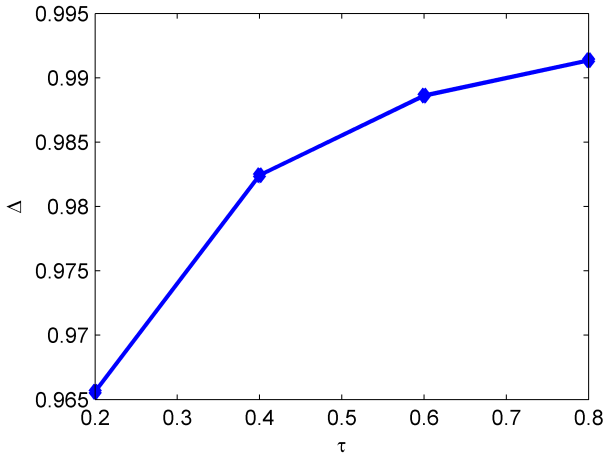



Figure 4.3: The relative performance difference ($\Delta$) as a function of $\tau$

#### 4.4.3.3 Resource utility

The influence of resource utilization ($\gamma$) on the relative performance difference $\Delta$ is shown in Figure 4.4. The figure shows that the performance of $LS_a$ is positively correlated (i.e. $\Delta$ is negatively correlated) with $\gamma$. Recall that as $\gamma$ increases, the utilization of technicians and, therefore, the constraint level of resource utility grows. This confirms the outstanding performance of the $LS_a$ on highly constrained instances of the FWS problem. When $\gamma$ is equal to 1.3, $LS_a$ can produce 2.5% better results than $LS_r$. The difference decreases down to about 1% on under-constrained instances (i.e. $\gamma = 0.8$).

### 4.4.4 Discussion

The experimental results suggest several remarks. They confirm that varying problem characteristics influences the performance of the applied scheduling techniques whose efficiency cannot be retained on all problems of all characteristics. In particular, the results suggest that the defined problem characteristics (i.e. $|R|$, $\sigma$, $\tau$, and $\gamma$) provide possible explanations of the performance difference between the assignment solution

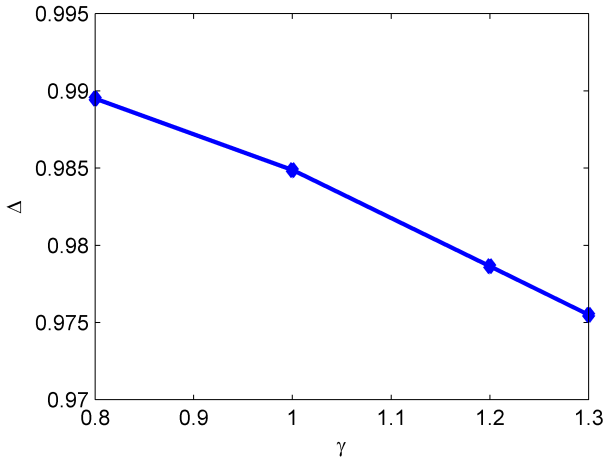



Figure 4.4: The relative performance difference ($\Delta$) as a function of $\gamma$

approach ($LS_a$) and the routing approach ($LS_r$).

Overall, the results indicate the sensitivity of the routing approach to the constraint level of the problem. On relatively over-constrained problems (in terms of skill constraint and resource utilization), the assignment approach significantly outperforms the routing approach. On the other hand, the routing approach shows very competitive performance to the assignment approach on large size problems.

The performance difference between the two approaches can be attributed to two factors. First, the level of effectiveness of the neighbourhood (i.e. move) operator employed by these methods. The assignment approach applies a simple neighbourhood function, that is the single-swap operator between any two tasks; whereas the routing approach maintains multiple efficient neighbourhood operators within each rout and between different routes. This factor enhances the performance of the routing approach (compared to the assignment one) on problems of large size.

The second factor concerns the constraint-handling technique used by each algorithm to contain infeasible solutions. In this study, a deterministic rule-based procedure is used by the assignment approach to ensure it generates only feasible solutions. On the other hand, any infeasible solution obtained by local changes to the current solution

is rejected by the routing approach defined in this study. This factor contributes to the outstanding performance of the assignment approach (compared to the routing one) on problems with tight constraints. It is worth pointing out that the results encourage the investigation of the application of routing approaches with more advanced constraint-handling techniques. This direction is not considered in the present study, due to the scope of the research.

With relation to our research, the development of a flexible model for FWS requires the consideration of employees' requests and preferences (i.e. their constraints on the tasks assigned to them). This implies the increase in the constraint level of the FWS problem. Since the present study suggests the effectiveness of the assignment approach on highly constrained FWS problems, we will opt for the assignment approach when solving the new scheduling model that is to be elaborated later.

Although the results of the characterization process carried out here are revealing, it offers a study that can be further extended, and thus the results be more generalized. For example, the correlations of the defined characteristics with hardness are still to be studied. The hardness of a problem instance for a particular solution procedure can typically be described by the CPU-time an exact solution procedure needs to solve a particular problem instance to optimality. Furthermore, there are other possible problem characteristics such as time-related constraints, and alternative models for the defined characteristics (e.g. see the characterization of nurse scheduling in [122]).

## 4.5 Conclusions

The FWS problem is a complex combinatorial optimization problem. The process of developing an efficient algorithm to tackle such an optimization problem requires a proper understanding of the problem's properties. Problem formulation and characterization are very important steps which are often underestimated in this process. An

attempt in this direction is made in this chapter which gives a formal mathematical definition of the problem, and describes a problem instance generator. It also identifies problem's key characteristics, namely the problem size and the level of constraint, which are found to impact the performance of scheduling techniques.

Problem characterization helps in relating the performance of different algorithms to the properties of the problem. This was demonstrated empirically by analysing, under various characteristics, the performance of a representative local search algorithm from the literature of routing problems, and another one from that of assignment problems. The experimental results confirm that varying problem characteristics influences the performance of the applied scheduling techniques whose efficiency cannot be retained on all problems of all characteristics.

Overall, the results reveal the outstanding performance of the assignment approach over the routing approach on over-constrained problems, both in terms of skill-matching and resource utility, as well as in relatively small instances. As the constraint level reduces, and the problem size grows, the performance of the routing approach enhances.

Based on the results of this study, we will use the assignment approach throughout the rest of our research, which concerns incorporating empowerment into FWS. This is due to the tight constraints imposed by involving employees in the scheduling process, as well as the effectiveness of the assignment approach on highly constrained FWS problems.

# Chapter 5

# Empowerment Scheduling of Field Workforce

This chapter proposes a new field workforce scheduling model that implements empowerment as an alternative, flexible management approach, with the hope of improving employee morale, and productivity. We begin by formalizing empowerment in workforce scheduling and establishing the term "Empowerment Scheduling", section 5.1. Sections 5.2 and 5.3 propose and discuss a new Empowerment Scheduling model (EmS) for designing workforce scheduling systems. Several computational experiments are conducted to examine various aspects of EmS applied to the FWS problem. These experiments and their results are explained in section 5.4. Finally, we conclude with section 5.6.

## 5.1 Empowerment Scheduling

*Empowerment Scheduling* is a term we introduce to formalize the employee empowerment in workforce scheduling, and therefore it describes scheduling models that incorporate the empowerment concept. In section 2.2, empowerment was discussed from a

management perspective. Based on that discussion, an attempt is made here to answer vital questions about empowerment scheduling models. Such questions include: What constitutes empowerment in workforce scheduling? What are the features of an empowerment scheduling model? How should the efficiency and effectiveness of an empowerment scheduling model be measured? Addressing these issues will help to develop an effective empowerment scheduling model and to evaluate the existing flexible scheduling models.

### 5.1.1 Constitution of Empowerment in FWS

A key issue that needs to be established is the definition of what constitutes empowerment in workforce scheduling systems. Recall that there are different conceptions of empowerment reflected in the management literature [51, 29, 37, 1]. The most general conception defines empowerment as a combination of a management relationship (i.e. enhancing employee power over decision making) and a psychological value (i.e. enhancing their self-efficacy). This describes the concept of empowerment that we consider here, which views empowerment as a combination of enhancing both employee decision power and self-efficacy. Therefore, an empowerment scheduling model should first recognize an individual's self-interest by providing employees with an empowerment practice through which they have control over the allocation decision. At the same time, employees need to be convinced that they have at least some control over the decision-making task; thus, the model should reflect employees' power in the final allocation decision.

### 5.1.2 The Essential Features

Several features should exist in any efficient empowerment scheduling model, which describe the requirements of an empowerment scheduling model. Deterioration in any one of these features would impact the feasibility of implementing this management

style. These are described as follows:

1. **Simplicity**. The model should be simple enough to facilitate the utilization of the potential of empowerment for all employees, whatever their capability levels are. This feature is motivated by the argument that employee perception and commitment varies according to factors like education, experience, and skilfulness [37, 58].

2. **Flexibility**. The model should be flexible to increase employee power and control over the scheduling decision. This is due to the crucial importance of an employee's feeling of power to the success of empowerment [29].

3. **Fairness**. Employees should trust the system and feel it is fair and transparent. This is arguably the most critical feature that should be considered carefully. The absence of fairness would dramatically weaken employee trust in the system, and thus the potential benefit of empowerment would be lost.

4. **Providing a win-win approach**. The model should provide a convincing beneficial solution for both the employees and the organization, presenting an all-win scenario. The feasibility of the model in terms of the impact of empowerment on the organizational interests, and the retaining of the organizational control over the scheduling process (as in *traditional* scheduling models), would motivate organizations to apply an empowerment scheduling model.

### 5.1.3 The Measurement Process

Another key issue that needs to be addressed is the measurement process of any empowerment scheduling model. For empowerment management practices in general, this process requires three different measures for evaluating three aspects of empowerment: model efficiency, organization benefits, and employee benefits. In the empowerment

scheduling context, model efficiency can be evaluated according to the first three properties of the aforementioned features of an efficient empowerment scheduling model, namely simplicity, flexibility and transparency/fairness. On the other hand, measuring the benefits for organizations and employees is a difficult process. An ideal empowerment scheduling model would be able to provide employees with full control over work allocation without impact on the optimality of the organizational objective. If this cannot be achieved, the model should otherwise enhance employee power in decision making and employee contribution to the organization, while minimizing the impact on the organizational objective.

### 5.1.4 Current Implementations of Empowerment Scheduling

This section compares and contrasts empowerment scheduling with other modelling approaches, namely self-scheduling and preference scheduling. A summary of these comparisons is depicted in Figure 5.1.

#### 5.1.4.1 Empowerment Scheduling vs. Self-scheduling

Self-scheduling (section 2.2.2.1), which is about enabling employees to construct their own schedule whilst respecting some requirements determined by a manager, is a flexible empowerment scheduling model. It satisfies the constitution of empowerment (as defined in section 5.1.1) since it identifies employees' individual interests which are reflected in the final schedule. From the essential features of an empowerment scheduling model as listed in section 5.1.2, self-scheduling incorporates flexibility and all-win approach, but lacks simplicity and fairness. The model is not simple enough: it is a manual scheduling model, it is only applicable to some scenarios where the scheduling problem is known well in advance, and it requires a conflict resolution policy. Transparency and fairness are also critical issues which are normally underestimated in this model. The absence of these features would greatly impact employees' self-efficacy.

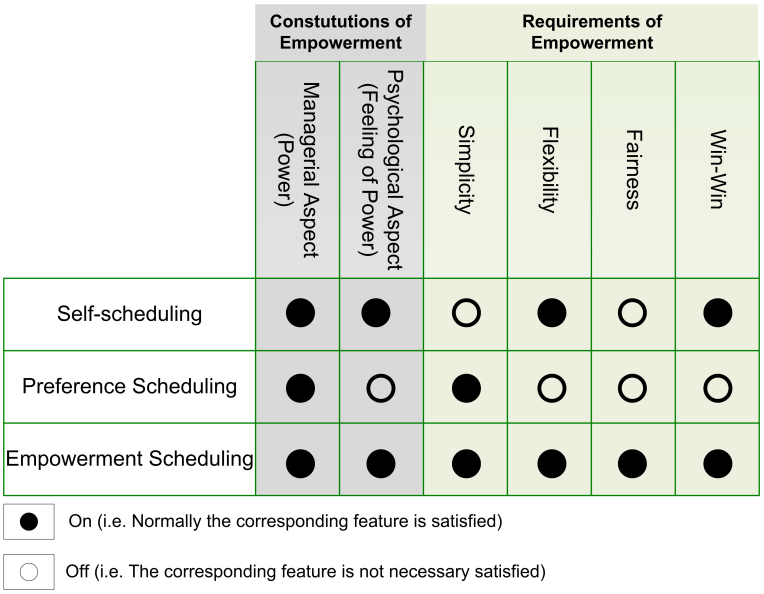



Figure 5.1: Feature Comparison: flexible scheduling models

#### 5.1.4.2 Empowerment Scheduling vs. Preference Scheduling

Preference scheduling (section 2.2.2.2) is another well-known flexible management scheduling model, which attempts to accommodate individual preferences when creating schedules. As reviewed in section 2.2.2.2, we distinguished between four preference scheduling approaches. In the first approach, the employer defines an employees' satisfaction measure without the employee's explicit involvement. The employees' individual interests are not recognized in this approach, and thus employees have no explicit authority over the allocation decision. As a result, this approach does not satisfy the constitution of empowerment scheduling as defined in section 5.1.1.

The second approach empowers employees by providing a simple, limited involvement practice that enables employees to express their preferences about the scheduling item. The simplicity of this model impacts its flexibility, as the scope of employee involvement is very limited and employees cannot suggest their specific requirements. Furthermore, fairness is not considered in this approach, and thus the psychological

aspect of empowerment is underestimated. Therefore, this approach does not attain the essence of empowerment scheduling since the essential features, namely flexibility, fairness, and consequently all-win, are not incorporated into such scheduling models.

The third preference scheduling approach enhances the flexibility of the second approach by allowing employees to express their individual requests. Although models in this approach incorporate flexibility as a feature, fairness and transparency are still absent, which would seriously impact employees' trust of the system, and thus the benefit of empowerment is not guaranteed.

The fourth approach includes models that improve the fairness of the third approach, presenting models that satisfy, to various extents, the constitution of empowerment scheduling, and maintain its critical features. These methods include those of Eveborn and Rönnqvist [42], Bard and Purnomo [11] and De Grano et al. [35]. The latter is an auction-based model, which is more complex when compared to the two former models. As we mentioned earlier, the concern with this auction-based approach is its high susceptibility to game playing, which means the system is not equally utilized by all employees whatever their education, skilfulness and personal characteristics. In such a situation, it is likely to depress some employees and thus weaken their self-efficacy. On the other hand, the fairness mechanisms implemented in the model of Eveborn and Rönnqvist's, and in that of Bard and Purnomo [11], assume that each employee has a set of preferences and requests, and thus they can be fully or partially satisfied. Therefore, such mechanisms are impractical in short-term task assignment problems, such as the FWS problem. In such problems, the scheduling horizon is very short, and employees' requests would vary from day to day. Hence, each employee may have one request per schedule.

## 5.2 EmS: An Empowerment Scheduling Model

Our aim is to develop a general empowerment scheduling model, and demonstrate its application to the FWS problem. We focus on formulating the problem in such a way that it provides employees with a simple, flexible, yet explicit control over decision making. Transparency, fairness, and providing a win-win approach are the main concerns of our model. This section introduces this model in relation to the FWS problem (as defined in chapter 4) and extends the formulation of the FWS problem accordingly.

### 5.2.1 Work Plan: The Empowerment Vehicle

In order to deploy the empowerment management concept in the workforce scheduling problem, employees should be involved in the scheduling process and given some power to influence their own schedule. This can be attained by enabling them to express their preferences and submit (daily) requests for consideration in the allocation process.

A key element in any empowerment scheduling model is the representation of employee involvement. In workforce scheduling, employees would like to be given a decision power that would allow them to influence the allocation process. In FWS, for instance, an employee would like to be able to plan for a particular day, e.g. "I want to finish my shift at a particular area/location", or "today I want to do only jobs that are of a particular skill(s)".

This suggests that the representation of employee involvement in the decision-making process can be expressed using constraint language [113]. An employee can instantiate a work plan which is a constraint relationship that can express most possible scenarios, including requests and preferences. The work plan is the vehicle of empowerment in our model, by which employees can express their own specific requests. These constraints will then be considered during the scheduling process.

Workforce scheduling problems are very complex, and thus the promise of satisfying

all plans cannot be guaranteed. One direct way to handle employees' work plans is to treat them as soft constraints such that each plan is assigned a violation cost (i.e. penalty). In this case the task is to find a schedule that satisfies the majority of constraints, rather than all constraints. More precisely, the scheduling system will try to minimize the sum of penalties associated with the violated constraints.

### 5.2.2 The Updating Mechanism

The extreme difficulties in satisfying all plans initiates the need to define an efficient strategy that determines the incurred cost of violating each plan. The higher the violation cost of a plan, the higher the incentive for the scheduling system to satisfy this plan. A naïve strategy is to assign an equal cost (i.e. power) to all plans. However, this strategy lacks transparency and fairness. When all plans are equal, there is always a chance that plans of a particular employee are satisfied more often than those of another employee. To overcome this problem, we propose another strategy based on the argument that employees should vary in the power they have in decision making, reflecting the employees' history of their plan satisfaction. In our proposed strategy, every employee has an *employee power* ($EP$) that describes the amount of power (s)he has. The more power (i.e. the higher $EP$ value) an employee has, the more chance that his/her plan could influence the decision-making process.

The values of $EP$s are continually managed by EmS via an *updating mechanism*. The aim of this mechanism is to ensure the effectiveness and fairness of the empowerment model by compensating employees whose plans were not satisfied. The basic idea is that each time an employee is satisfied, his/her $EP$ value is reduced. Thus, employee power can be seen as an internal currency. An employee whose proposed plan is accepted will pay with a certain amount of $EP$.

On day $d$ (assuming that the scheduling system works on a daily basis), the $EP$ value of an employee $r$ with a total number of satisfied plans in previous days ($sat_r$),

is calculated as follows:

$$EP_r^d = (d - sat_r)/d \tag{5.1}$$

On the first day, the value of each $EP$ is initialized by one, i.e. all employees have equal opportunities. Afterwards, the values of $EP$s are updated based on the output of the previous schedule(s).

From an implementation point of view, this equation requires keeping track of two variables per employee: $EP$ and $sat$. This can be reduced to only $EP$ by deriving the value of $sat$ as a function of $EP$ and the current day $d$ (i.e. $sat_r = d - d * EP_r^d$). Thus, the new $EP$ for employee $r$ can be calculated as follows:

$$EP_r^{d+1} = \frac{1 + EP_r^d * d - y_r}{d + 1} \tag{5.2}$$

Where $y_r$ is a satisfaction indicator for the current plan of day $d$, such that $y_r \in \{0, 1\}$.

### 5.2.3 A Multi-objective Optimization Approach

From an optimization perspective, employees' plans add more soft constraints to the problem, and thus impact on the optimality of the productivity objective. Therefore, the organization may need to pay for satisfying employees' plans out of the productivity rate. The amount of optimality the organization will give up defines the *cost* of empowerment. This cost is hoped to be offset by the improvements in service quality and productivity which result from the higher employee morale and motivation as a consequence of employee empowerment. These improvements describe the *profit* of empowerment. If the profit of empowerment (in terms of improvement in the productivity rate) is measured and its relationship to the cost (in terms of decline in the productivity rate) is defined, then the empowerment objective and the productivity rate can be modelled in a single objective. However, the benefits of empowerment depend on employees (i.e. human) who are complex and difficult to be modelled, and consequently it

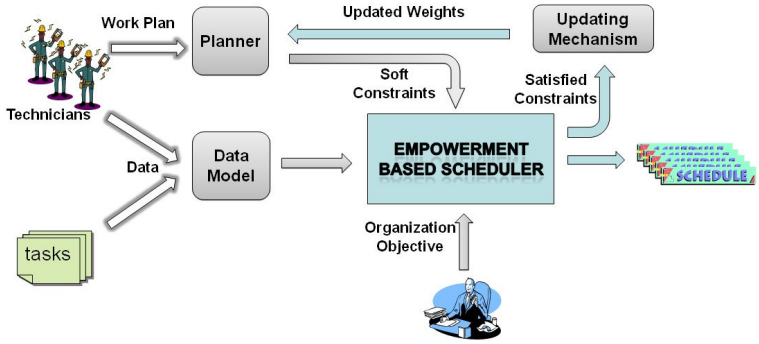
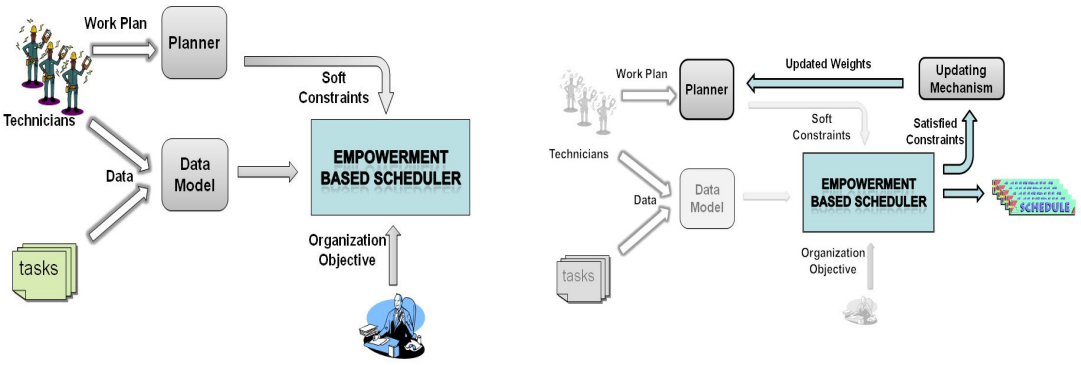



Figure 5.2: The framework of the proposed empowerment scheduling model (EmS)

is difficult to formalize the relationship between the empowerment cost and profit. As a result, the organizational objective and empowerment objective can be considered as conflicting objectives, at least at the optimization stage. This implies that the problem can be formulated as a multi-objective optimization problem, where the target is to find a set of solutions that represents the optimal set of trade-offs between the two objectives.

### 5.2.4 EmS-FWS: The Implementation of EmS for FWS

Figure 5.2 summarizes the application of EmS to FWS (referred to as EmS-FWS). The model can be divided into two stages. In the first stage (Figure 5.3(a)), the technicians' and tasks' details are fed into the scheduling engine to construct an optimal complete schedule. The quality of a schedule (i.e. the optimization criterion) is determined by both the organizational objective and plans satisfaction. The latter is measured by the cost of violated employees' work plans. On the first run, the violation costs (determined by the $EP$ value of each employee) of all work plans are set to one.




(a) EmS: the first stage - generating a schedule

(b) EmS: the second stage - the updating procedure

Figure 5.3: The two stages of EmS

The second stage (Figure 5.3(b)) starts once the scheduler produces the final schedule. The purpose of this stage is to ensure the fairness of the model by updating the violation cost for each employee's work plan in the next schedule. This is maintained by the updating mechanism which recalculates the $EP$ value of each employee as a function of his/her satisfaction history (Equation 5.1).

The formulation of the FWS problem needs to be extended as follows, in order to include the new elements introduced by EmS. Every technician $r \in R$ is given an opportunity to provide a work plan ($wp_r$) everyday. A plan is basically a constraint ($\rho$) by which a technician can describe the jobs (s)he wants to undertake. For instance, a technician $r$ might want to be allocated to jobs that require a particular set of skills, or jobs in a particular region. Each plan is associated with a weight, in our case this is equal to the $EP$ value of $r$. An indicator $y_r$ ($\forall r \in R$) is defined, which is equal to 1 if the plan $\rho_r$ is satisfied and 0 otherwise.

The empowerment practice introduced by EmS changes the nature of FWS. The underlying optimization problem is reformulated to not only maximize productivity in

terms of the number of served jobs, but also to satisfy the maximum number of work plans. Similar to the productivity objective, the plan satisfaction objective will be normalized by dividing by the sum of all weights associated to plans.

The work plans augment the FWS problem (section 4.1) with the following additional objective:

$$wp_r \quad = \quad < \rho_r, EP_r > \quad \forall r \in R, \quad 0 \leqslant EP_r \leqslant 1 \tag{5.3}$$

$$max \qquad \frac{\sum_{r \in R} y_r * EP_r}{\sum_{r \in R} EP_r} \tag{5.4}$$

At this stage (i.e. in this chapter), the EmS-FWS problem is dealt with as a single-objective optimization problem by combining the two optimization functions defined in Equation 4.1 and Equation 5.4 using the weighted-sum approach:

$$f(s) \quad = \quad w_1 * \frac{\sum_{r \in R} \sum_{t \in T} c_t x_{rt}}{\sum_{t \in T} c_t} + w_2 * \frac{\sum_{r \in R} y_r * EP_r}{\sum_{r \in R} EP_r} \tag{5.5}$$

In this function, $w_1$ and $w_2$ are predefined constants that determine the importance given by the organization to each objective, such that $w_1 + w_2 = 1$.

The reason for aggregating the objectives is that controlling the weight setting given to the two objectives would help in understanding the correlation between both objectives, and in evaluating various aspects of EmS under different weight settings.

## 5.3 Discussion

There are three factors that contribute to the effectiveness and practicality of EmS as a flexible scheduling model. These factors are discussed here, and linked to the essential features of an empowerment scheduling model (section 5.1.2), namely simplicity, flexibility, fairness, and providing an all-win scenario.

The first factor is the employment of constraints as the communication means between employees and the system. Simplicity and flexibility as essential features are incorporated in EmS by using *constraint*, which provides a simple, yet flexible approach to describe most scenarios. Using constraints also enables the benefit from the field of Constraint Satisfaction Problem (CSP) [113] which is well-established research. From the literature of CSP, EmS applies the concept of soft constraint which allows the system to violate a constraint at an incurred penalty.

The second factor of EmS is the incorporation of an updating mechanism. It provides an automatic strategy to control the violation cost of each employee's work plan, i.e. $EP$. This considerably enhances the simplicity, transparency and fairness of the model. Employees' involvement is limited to the actual planning only. They do not have to think or reason about determining the weights associated to their work plans. This improves the usability of the model for all employees whatever their skilfulness and capability levels are. Moreover, using a deterministic procedure to update the $EP$ makes the model transparent, addressing the important challenge of enhancing employees' trust of the system. The proposed updating procedure calculates the new $EP$ value of each employee as a function of his/her satisfaction history. This helps the model to maintain fairness, which is a critical challenge.

It is important to highlight that the updating procedure can be considered as an input to EmS. Therefore, the proposed updating mechanism above can be adapted or replaced if the scheduling context changes or the scope of $EP$ is extended. We actually

believe on the potential of the $EP$ parameter to be more than a reflection of employee's satisfaction history. Its scope can be extended to reflect the amount of an employee's contribution to organizational success e.g. productivity rate, failure rate, or customer satisfaction. This ensures the delivery of a new rewarding system, which supports other systems that rely on bonus-based rewarding mechanisms.

The equation used to model the proposed updating strategy (Equation 5.1) suggests two main indications that need to be highlighted. First, the updating mechanism considers only the current day and the satisfaction history of an employee to calculate the new $EP$ value. This implies that there will be no difference in the $EP$ value of an employee whether (s)he has not proposed a plan for a day or his/her plan for that day was unsatisfied. The motive for such an equal treatment is to prevent gaming by employees. To illustrate this, assume that an employee whose plan was unsatisfied will be compensated by an increase in his/her $EP$ value. A savvy employee may instantiate plans which are impossible (or hard) to satisfy, in order to increase his/her $EP$ value the next day. Such a case is captured by this updating strategy.

The second indication is that an employee's plan is considered to be either satisfied or unsatisfied, regardless of how hard/easy the satisfaction of that constraint is. This is justified by the principle that an employee should pay for 'securing' a situation (s)he wants, even if that situation is very probable to happen anyhow. On the other hand, employees will neither pay for submitting hard or impossible non-granted plans, nor will they be recompensed. However, such scenarios can sometimes be easy to detect, which stimulates the addition of a preprocessing component outside the system, which tries to catch impossible or hard plans and allows for replanning. The practicality and feasibility of such extensions are issues that need to be investigated after developing the basic system, which is the concern of the research in this thesis.

The third factor of EmS is the multi-objective formulation of the underlying optimization problem. This is considered as a feature for two reasons. First, multi-objective

optimization is a well-studied topic and its literature is rich with useful concepts and efficient techniques that can be adopted for solving the present problem effectively. For example, multi-objective optimization algorithms return an optimal set of trade-offs between objectives. In our case this set helps the organization to assess the impact of empowerment on the organizational interests. The second reason is that using the multi-objective approach ensures that the organization is still in charge of the scheduling process as in *traditional* scheduling. Thus, the organization can avoid undesirable outcomes by controlling the amount of optimality they are willing to give up in order to consider employees' work plans. For example, tasks undesirable to any of the employees will still be done by controlling their importance (compared to other tasks) to the organizational objective and/or the importance of the organizational objective compared to the plan satisfaction objective.

## 5.4 Computational Experiments

EmS applied to the FWS problem, as outlined above, is implemented. In order to examine the effectiveness of the model, particularly the feasibility of the empowerment concept and the model's fairness, four sets of experiments were conducted on benchmarks produced by the problem generator (section 4.2) with its default values. Below, we detail the problem instances, the applied algorithm and the experimental results.

### 5.4.1 Work Plan Generator

A major element in our experimental design is to model employees' plans. Recall that a plan is a constraint that limits the possible values a variable can take, and employees want to be able to describe jobs that they want to do. There are three main properties that can be used to describe a job in FWS, namely the job's id, type (i.e. required skill) or location.

```
Plans Generator(R, T, SkillSet, Areas, σ)
  for each r ∈ |R| do
    P ← Random("id", "skill", "location")
    if P = "id" then
      size ← Random(σ ··· 2 * σ)
      D_r ← Random(d|d ⊆ T, |d| = size)
    else if P = "skill" then
      size ← Random(1 ··· min(|skills_r|, |SkillSet|/2))
      D_r ← Random(d|d ⊆ skills_r, |d| = size)
    else if P = "location" then
      size ← Random(1 ··· |Areas|/2)
      D_r ← Random(d|d ⊆ Areas, |d| = size)
    end if
  end for
```

**Algorithm 5.1:** The outline of the generator of employees' work plans

Our model is based on these properties, such that each plan uses a property to describe the tasks that the corresponding employee wants to do. When an employee wants to do particular tasks or tasks that require particular skill(s), (s)he just needs to determine that tasks' id or their skill codes, respectively. In order to facilitate the use of task location as a descriptor, the whole service region of the FWS problem is clustered into several areas which can be used by employees to limit the jobs that will be assigned to them to particular area(s).

Given the sets of technicians ($R$), tasks ($T$), skills ($SkillSet$) and areas ($Areas$), and the number of tasks, on average, that a technician can perform on a daily shift ($\sigma$), the plan generator is outlined in Algorithm 5.1. The plan of an employee is generated randomly. First, a property will be chosen randomly, which can take three possible values: *id*, *skill* or *location*, which makes the plan describe tasks by their identifier, skill requirement or location, respectively. Accordingly, the size of the domain value is drawn randomly within a specific range. This range is calculated as a function of the number of tasks, on average, a technician can perform on a daily shift for the *id* property; the size of the employee's set of skills for *skill*; or the number of areas that

construct the whole service region for *location*. Finally, a subset of the relevant set to the chosen property is randomly selected as the domain value.

Table 5.1 shows three examples of work plans which illustrate how the constraints are generated and interpreted. These plans are taken from an instance of the benchmark generated for a study in this thesis. In the first example, the work plan of the employee with Id 25 restricts the type of jobs that can be assigned to the employee to those that require either skill 2 or 5, eliminating other skills the employee possesses (i.e. skills 3, 7, 8 and 9). Thus, during the optimization process, a soft constraint is instantiated to check whether all assigned jobs to this particular employee are of skills 2 or 5.

Table 5.1: Examples of an employee's work plan

| Employee ID | Work Plan | Constraint Variable | Domain Values |
|---|---|---|---|
| 25 | "I want to take jobs only of type 2 and 5" | Skill | 2,5 |
| 40 | "I want to be assigned to jobs in areas 1 and 2" | Location | 1,2 |
| 51 | "I want to do only jobs from this particular set: 7, 15, 18, 55, 79, 82, 118, 128, 143 " | Job ID | 7, 15, 18, 55, 79, 82, 118, 128, 143 |

### 5.4.2 Algorithm: Guided Local Search

As reviewed in section 2.1, various approaches have been applied to FWS problems. Among these approaches, GLS obtained the best results on a relevant real benchmark problem [114]. The proposed GLS in [114] employs an assignment-based local search that is defined in section 4.3.1. The performance of this local search method was examined in comparisons with a representative routing-based local search, as detailed

in chapter 4. The computational results suggested the superiority of this local search over the routing-based one, particularly on problems of high constraint levels. In EmS, employees' work plans are basically constraints that increase the constraint level of the problem. This suggests the potential of the assignment-based local search for solving EmS.

The outstanding performance of GLS on FWS problems, and the potential of the underlying local search to be an effective method for EmS, motivates the study of the application of GLS to the EmS-FWS problem. The implementation of the basic components of GLS to the EmS-FWS problem is described below.

#### 5.4.2.1 The Local Search

The local search defined in section 4.3.1 is applied to the EmS-FWS problem here. This requires the adaptation of only the following two basic components:

- Solution Representation: A candidate solution is represented by a permutation of tasks, determining the order of tasks to be considered by a defined scheduling procedure. The applied scheduling procedure can be summarized as "allocate this task to the nearest technician with respect to all constraints and technicians' plans". At the beginning, a list of technicians is associated to each task, which involves those who are qualified to perform the associated task. The list will be sorted before the searching starts, and it remains static during the search. The sorting criteria are: (1) the technician's plan, those who prefer (in accordance with their work plans) the associated task come first and (2) travelling cost, the nearest technicians come first.

- Cost Function: For the EmS-FWS problem, the cost function ($f(s)$) will be the weighted-sum function defined in Equation 5.5.

#### 5.4.2.2 Solution Features and their Costs

In EmS-FWS, $f(s)$ is an aggregation of two functions with different weights, and thus two sets of features will be defined. Since minimizing the total unallocated tasks is a key objective, each feature in $F_1$ represents failure in serving a job (i.e. the exhibition indicator $I_i^1$ equals 1 if task $i$ is unallocated, 0 otherwise). Thus, $F_1$ has $|T|$ features, where $|T|$ is the total number of tasks. The second objective concerns minimizing unsatisfied employees' plans, and therefore each feature in $F_2$ represents the failure in satisfying an employee's plan (i.e. the exhibition indicator $I_j^2$ equals 1 if technician $j$ is not satisfied, 0 otherwise). There are at maximum $|R|$ features in $F_2$, where $|R|$ is the total number of technicians.

Each feature is associated with a cost to help GLS in choosing features that have more influence on the cost function in order to penalize them. We consider the task priority as the cost of features in $F_1$, in order to give higher priority to tasks of higher importance, whereas the weights associated with work plans ($EP$) are the cost of the features in $F_2$.

As a result, the augmented objective function $h(s)$ is expressed as follows:

$$h(s) = f(s) + w_1 * \lambda_1 \sum_{i \in F_1} p_i * I_i^1 + w_2 * \lambda_2 \sum_{j \in F_2} p_j * I_j^2 \tag{5.6}$$

The weighted-sum method, which is applied to transfer the two objective functions into a single function, is also propagated to the corresponding set of features. Moreover, we defined two parameters of GLS $\lambda_1$ and $\lambda_2$; however, empirical results suggest that EmS-FWS was relatively insensitive to those parameters.

### 5.4.3 Experiment 1: Feasibility - Trade-off Analysis

The aim of the first experiment is to assess the immediate impact of adopting employee empowerment on the main organizational objective, i.e. the feasibility of empowerment.

This is attained by examining the underlying optimization problem of the new model, in comparison to the *traditional* model. Although the results are dependent on the applied algorithm (i.e. GLS) the concluding remarks will still be significant, given that the proposed local search and GLS are well-known and representative methods for such optimization problems.

For the purpose of this experiment, only the optimization problem of one day (i.e. without the updating mechanism) is considered, and therefore we assume that all employees are of an equal power (i.e. all plans are of the same weight). We intend to examine the tightest scenario, where every employee has a plan to be considered.

We ran the GLS algorithm on 20 problem instances. For each problem, different sets of weights $(w_1, w_2)$ were tested, such that $(w_1, w_2) \in \{$ (1,0), (0.9,0.1), (0.7,0.3), (0.5,0.5), (0.3,0.7), (0.1,0.9), (0,1)$\}$. In one extreme, when $w_1 = 1$, the plan satisfaction objective (Equation 5.4) is ignored in the optimization process, representing the *traditional* model for FWS. As $w_1$ decreases, the influence of the plan satisfaction objective on the optimization process should increase.

Since the plans generator is based on a stochastic model, five different instances of employees' plans were tested for each weight setting.

In this experiment (and the following experiments), we always present the actual cost of each objective rather than the aggregation of both objectives. For each weight setting, the mean of five runs with different employee plans is calculated for each instance, and then the mean of 20 instances is measured.

The results are given in Table 5.2 which describes the mean cost of each objective for each weight setting. For simplicity, the "Productivity Loss" is included, which refers to the reduction in the productivity rate as a result of satisfying a subset of employees' work plans, e.g. the Productivity Loss for the weight setting $(0.5, 0.5) = 100 - (87.8/94.1)$. The correlation between the productivity loss and the empowerment cost is plotted in Figure 5.4, which shows the cost of satisfying employees' plans on the

Table 5.2: The mean cost of each objective for each weight setting

| Weight Setting $(w_1, w_2)$ | Productivity (%) | Productivity Loss (%) | Satisfaction (%) |
|---|---|---|---|
| (1,0) | 94.1 | 0 | 14.7 |
| (0.9,0.1) | 92.2 | 2 | 42 |
| (0.7,0.3) | 89 | 5.5 | 55.1 |
| (0.5,0.5) | 87.8 | 6.7 | 57.5 |
| (0.3,0.7) | 87.6 | 7 | 57.4 |
| (0.1,0.9) | 87.4 | 7.1 | 58.4 |
| (0,1) | 80 | 14.6 | 56.1 |

organizational main objective.

Interestingly, the results suggest that the plans of 58.4% of employees can be satisfied at the expense of losing 7.1% of the productivity objective. This is achieved when the weight given to the plan satisfaction objective is high (i.e. $w_2 = 0.9$). The results encouragingly show that even if the plan satisfaction objective is ignored (i.e. the *traditional* FWS), about 15% of all plans can be satisfied. The inability to satisfy all plans is expected and this is due to either the existence of conflict between multiple plans, in which only some of which can be satisfied, or to the inability of the optimization algorithm to obtain the global optimum solution.

### 5.4.4 Experiment 2: Feasibility - Organizational Benefit

One of the potential benefits of empowerment is improved productivity. The aim of this experiment is to contrast the benefit of productivity gain against the costs. This can be achieved by assuming a best-case scenario where the employee's productivity rate is improved. This improvement is modelled by a reduction in the task duration by 5%. For example, when the duration of a task is estimated to take 60 minutes, an empowered technician will finish it within 57 minutes. Measuring the organizational benefit from this improvement helps examine the potential of empowerment and the

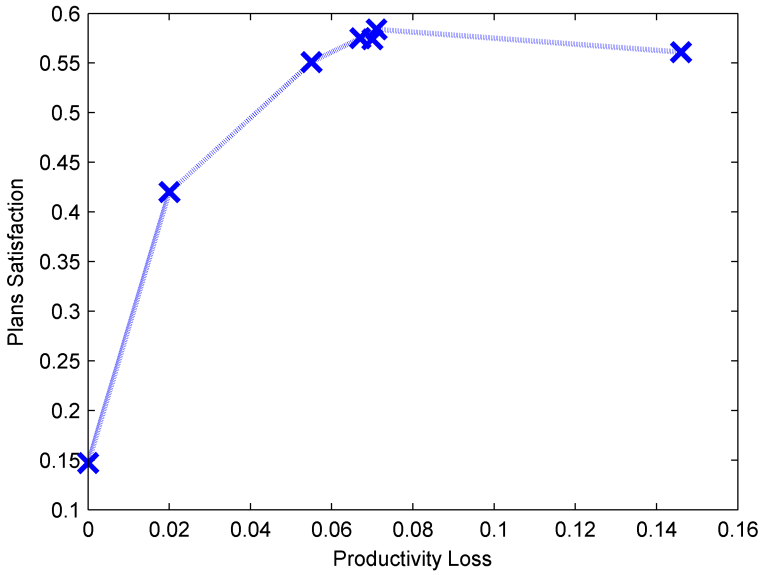



Figure 5.4: The correlation between the plan satisfaction objective and the productivity objective

feasibility of the model in providing an all-win scenario.

To obtain this measurement, Experiment 1 is repeated while reducing the duration of each task by 5%. The results, then, are compared to those obtained in Experiment 1 (i.e. with the original tasks durations). The idea is to analyse the contribution of a 5% improvement in task durations to the organizational objective.

The results are presented in Table 5.3 which shows, for each weight setting, the change in the productivity objective when task durations are reduced by 5%. Assuming that this reduction comes as a benefit of empowerment, the results reveal that this model will eventually improve the productivity (i.e. the organizational interest) by 2.7%-3.8% depending on the applied weight setting. Similarly, the plans satisfaction is enhanced by 20%-33%.

The promise of empowerment includes improvements in morale, job satisfaction, retention and productivity. This experiment shows that an improvement in the productivity will lead to a significant benefit to the organization. Therefore, the concept

Table 5.3: The mean cost of each objective for each weight setting when task durations are reduced by 5%

| $(w_1, w_2)$ | Productivity | | | Satisfaction | | |
|---|---|---|---|---|---|---|
| | before | after | ±% | before | after | ±% |
| (1,0) | 94.1 | 97.7 | +3.7 | 14.7 | 19.6 | +33.4 |
| (0.9,0.1) | 92.2 | 95.7 | +3.8 | 42 | 51.1 | +21.7 |
| (0.7,0.3) | 89 | 92.1 | +3.6 | 55.1 | 65.8 | +19.4 |
| (0.5,0.5) | 87.8 | 90.8 | +3.4 | 57.5 | 68.6 | +19.2 |
| (0.3,0.7) | 87.6 | 90.4 | +3.3 | 57.4 | 69.2 | +20.5 |
| (0.1,0.9) | 87.4 | 90.4 | +3.5 | 58.4 | 69.5 | +19.1 |
| (0,1) | 80 | 82.5 | +2.7 | 56.1 | 67.7 | +20.7 |

and the model are capable of providing a convincing win-win approach, and eventually a feasible empowerment scheduling model.

### 5.4.5 Experiment 3: Fairness Analysis

An essential feature of an empowerment scheduling model is fairness. This feature is handled in EmS by the proposed updating mechanism (section 5.2.2). The aim of this experiment is to check whether fairness is achieved by the updating mechanism. The success of EmS in providing a transparent and fair scheduling process via this novel updating mechanism is achieved when the variance between employees, in their satisfaction levels, is significantly reduced. Attaining fairness may have a negative impact on the overall average of employees' satisfaction. However, a successful updating mechanism should not negatively impact the organizational interests.

The effectiveness and usefulness of the updating mechanism is evaluated by running GLS on a month-scenario, i.e. a sample of technicians and 20 samples of tasks are generated, each sample of tasks represents a day (excluding weekends) in that month.

For each day: (1) a work plan is generated for each technician, (2) five different sets of weights, $(w_1, w_2) \in \{$ (0.9,0.1), (0.7,0.3), (0.5,0.5), (0.3,0.7), (0.1,0.9)$\}$, are applied, and then (3) for each weight setting, the GLS algorithm is run twice: once with the

updating mechanism that continually updates the EP values, and the other without updating, i.e. the cost of violating any plan is equal to one.

By the end of each month-scenario, for each weight setting and each updating approach, the following measures are calculated:

1. Productivity rate, which is represented by the task completion rate as defined in Equation 4.1. Here, the mean of the productivity rates of the 20 days is calculated, in order to measures the quality of the schedules from the organizational angle.

2. Employees' satisfaction rate ($SatRate$), which is the mean of employees' individual satisfaction level ($satLvl$). The satisfaction level ($satLvl$) of an employee $r$ is calculated in this experiment as follows:

$$satLvl_r = sat_r/period \qquad \forall r \in R \tag{5.7}$$

   where $sat_r$ is the total number of satisfied plans in the examined time horizon and $period$ is the total number of produced schedules (i.e. time horizon), which is equal to 20 in our case. The satisfaction rate is used here instead of the plan satisfaction objective (Equation 5.4) in order to measure the quality of the schedules from each employee's individual perspective. Equation 5.4 considers a single schedule/day and involves the weights associated to plans for that schedule/day, and thus it does not necessarily reflect employees' perception and satisfaction.

3. Variance ($var$), which measures the variability of the satisfaction level of an employee from the average satisfaction levels ($SatRate$). The most common measures are the Variance and the Standard Deviation; the Variance measure is used here. The closer this value is to 0, the less variable the employees' satisfaction rates and the fairer the schedule.

The process is repeated 20 times with a new sample of technicians generated every

time. Finally, the mean of each of the three measures over the 20 month-scenarios is calculated.

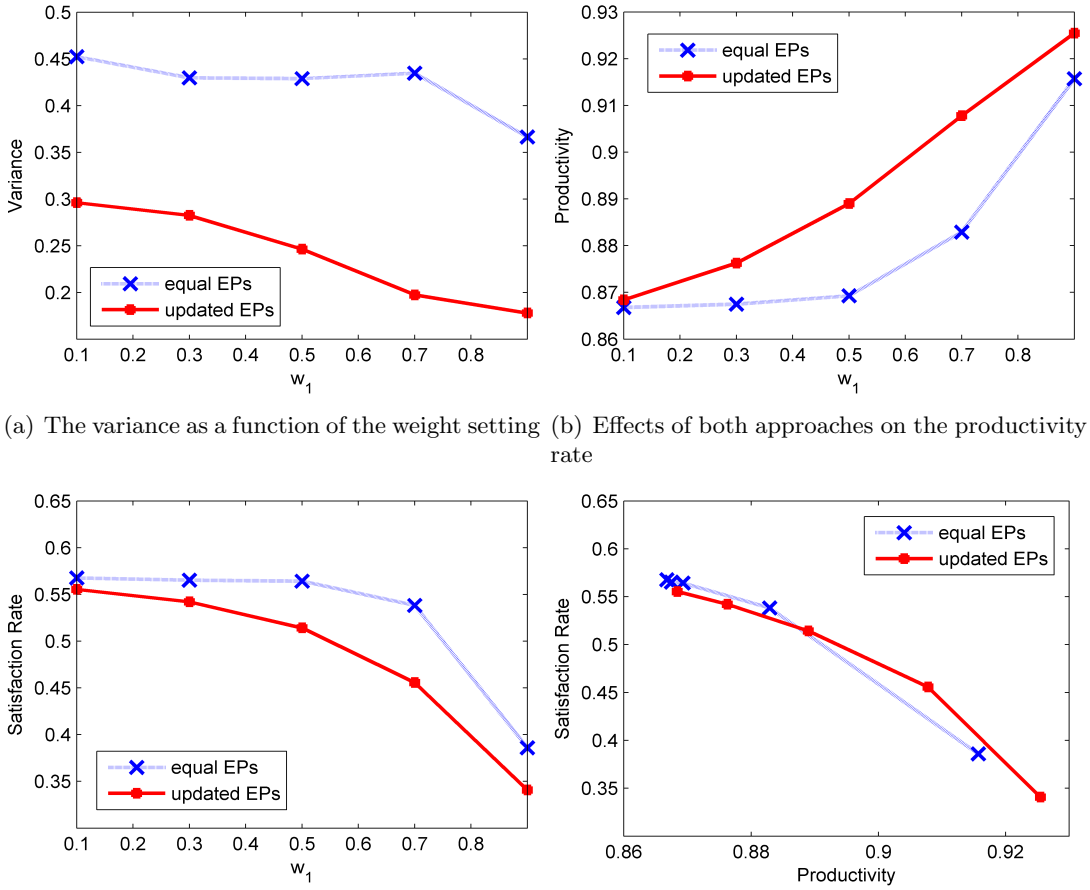
Table 5.4: The productivity rate, satisfaction rate, and variance of the model *with* and *without* the updating mechanism, at different weight settings, $(w_1, w_2)$

| Updating | Weight Setting | Productivity | $SatRate$ | Variance |
|---|---|---|---|---|
| No | (0.1,0.9) | 0.866 | **0.568** | 0.452 |
| | (0.3,0.7) | 0.867 | **0.565** | 0.429 |
| | (0.5,0.5) | 0.869 | **0.564** | 0.429 |
| | (0.7,0.3) | 0.883 | **0.538** | 0.435 |
| | (0.9,0.1) | 0.916 | **0.386** | 0.366 |
| Yes | (0.1,0.9) | **0.868** | 0.555 | **0.296** |
| | (0.3,0.7) | **0.876** | 0.542 | **0.282** |
| | (0.5,0.5) | **0.889** | 0.514 | **0.246** |
| | (0.7,0.3) | **0.908** | 0.456 | **0.197** |
| | (0.9,0.1) | **0.925** | 0.341 | **0.178** |

The results of this set of experiments are given in Table 5.4. The third and fourth columns represent the productivity rate and the employees' satisfaction rate, respectively. The fifth column indicates the variance of an employee's satisfaction level ($satLvl$) from the mean ($SatRate$). These values give the following indications:

- The updating mechanism enhances fairness by significantly decreasing the variance value under all weights, Figure 5.5(a). The updating mechanism reduces the variance by about 34% at high values of weight for the plan satisfaction objective ($w_2$), and as the weight decreases the difference increases. At low values of weight given to the plan satisfaction objective, the difference in variances can reach up to about 54%.

- Using the updating mechanism to change the penalty of violating each plan has no negative impact on the productivity objective, as plotted in Figure 5.5(b). Interestingly, the productivity objective attained better results with the updating mechanism.




(a) The variance as a function of the weight setting

(b) Effects of both approaches on the productivity rate

(c) Effects of both approaches on the employees' satisfaction rate

(d) The trade-off between the two objectives for each approach

Figure 5.5: Comparing EmS *with* and *without* the updating mechanism.

- To obtain fairness, the employees' satisfaction rate ($SatRate$) is sacrificed, Figure 5.5(c). The lower the weight given to the plan satisfaction objective (i.e. the higher the setting of $w_1$), the greater the difference in the satisfaction rate between the two approaches will be. At high values of weight for the plan satisfaction objective ($w_2$), the difference is between 2%-4%, which can reach up to 15% at very low values of $w_2$.

- Since the updating mechanism improves the productivity rate at the expense of

decreasing the overall satisfaction rate, this trade-off is plotted in Figure 5.5(d), and compared to that of the *traditional* approach (i.e. without updating mechanism). It shows that most of the solutions obtained by using the updating mechanism are non-dominated by those generated without updating. Thus, in addition to improving fairness, the updating mechanism produces competitive results to those obtained without updating.

These indications suggest, as a conclusion, the effectiveness of the updating mechanism in EmS, which provides a simple, yet efficient mechanism to improve the fairness and usefulness of the model. Whatever the weights given to the two objectives, the mechanism is able to significantly diminish the variance of an employee's satisfaction level from the mean satisfaction rate ($SatRate$). This is achieved without harming the productivity rate which, in contrast, is improved with this mechanism. This indication suggests that the conflict between the productivity objective and the plan satisfaction objective is slightly reduced by using the updating mechanism. Alternative interpretations would need further investigation in such a complex system.

On the other hand, minimizing the variance at the expense of the overall satisfaction rate was expected. The reason for this is that the freedom given to the greedy approach (i.e. without updating) to maximize the number of satisfied plans is heuristically controlled by the updating mechanism, which varies the cost coefficient associated with violating each work plan. Two main reasons make such a loss acceptable: (1) the aim of any empowerment scheduling model is to enhance employees' individual perception and feeling of power, and (2) without providing a transparent, fair model, the whole empowerment practice could fail to attain its benefits, as discussed earlier. Thus, the employees' satisfaction rate, which considers the employees as a group, is not more important than an employee's individual perception and trust in the system.

In addition, the results show a negative correlation between the weight given to the plan satisfaction objective ($w_2$), and the differences between the two approaches in the

variance, as well as the overall satisfaction rate. At one extreme, when the importance weight of satisfying plans is very low, the chance of violating many plans per day is very high. Consequently, the influence of the updating mechanism on deciding which plans are to be satisfied, with the aim to minimize the variance, grows. On the other extreme, when the importance weight is very high, this influence decreases due to the high importance of satisfying each plan to the aggregated objective function (Equation 5.5), even though a plan may have a low cost coefficient.

### 5.4.6 Experiment 4: Various Plan Demand Levels

The previous experiments considered the tightest scenario where every employee has a plan every day. By assuming that employees are rational beings, and given that an employee has to pay from his/her power ($EP$) when his/her plan is satisfied, employees would like to reserve their power in order to have influential plans (i.e. plans associated with high weights) for particular important days. Therefore, the number of employees who would plan for a particular day would fluctuate in real-life scenarios.

The goal of this set of experiments is to examine the performance of this model under various plan demand levels from employees. Theoretically, two main expectations may be anticipated: (1) there is a negative correlation between the plan demand level (as a measure of how many plans, on average, per day) and the optimality of both the productivity and plan satisfaction, and (2) there is a positive correlation between the plan demand level and the difference in the variance (as a measure of fairness) between the model *with* and *without* the proposed updating mechanism.

Testing these hypotheses is attained by repeating Experiment 3, while varying the number of work plans that are to be considered on each day. Three different values of the number of work plans per day are considered: 30%, 50%, and 70% of the number of employees, representing various plan demand levels ($stress$), where $stress \in \{0.3, 0.5, 0.7\}$ respectively. This means that when $stress = 0.3$, for instance, 30% of employees will

have a plan per day, and those employees are uniformly sampled per day. In terms of measurements, the only change to those used in Experiment 3 is the definition of the satisfaction level of each employee ($satLvl$), Equation 5.7. The definition of $satLvl$ is generalized as follows:

$$satLvl_r = sat_r/(stress * period) \qquad \forall r \in R \tag{5.8}$$

This expression also covers the situation in Experiment 3, where $stress = 1$. Thus, the results of Experiment 3 can be used to represent the case when $stress$=1.

The results are given in Table 5.5, which shows for each value of $stress$ and each weight setting, the results of the productivity rate, the employees' satisfaction rate, and the variance of an employee's satisfaction level from the mean. As a summary, the mean of the five different weight settings grouped by the $stress$ column is shown in Table 5.6. Major indications of these results are as follows:

- The effectiveness and usefulness of the model significantly increases as the plan demand level decreases, whether the updating mechanism is used or not. This is demonstrated in Figure 5.6(a), which gives the trade-off between the productivity rate and the employees' satisfaction rate at each demand level. It shows that the dominance level of the set of solutions obtained at each value of $stress$, grows as the plan demand level shrinks. As given in Table 5.6, with updating mechanism, the employees' satisfaction rate is 0.537 and the productivity rate is about 0.902 when $stress = 0.7$. These rates increase up to 0.682 and 0.918 for the satisfaction and productivity rates respectively, when $stress = 0.3$.

- Figure 5.6(a) also suggests that the correlations between using the updating mechanism and both the productivity objective and the satisfaction rate are not sensitive to changes in the $stress$ value. Whatever the value of $stress$, obtaining fairness by using the updating mechanism has no negative impact on the produc-

Table 5.5: The productivity rate, satisfaction rate and variance of the model *with* and *without* updating mechanism, at different plan demand levels (*stress*)

| *stress* | Updating | Weight | Productivity | *SatRate* | Variance |
|---|---|---|---|---|---|
| 0.3 | No | (0.1,0.9) | **0.911** | **0.742** | 0.602 |
| | | (0.3,0.7) | **0.911** | **0.753** | 0.615 |
| | | (0.5,0.5) | 0.912 | **0.744** | 0.604 |
| | | (0.7,0.3) | 0.917 | **0.717** | 0.592 |
| | | (0.9,0.1) | 0.931 | **0.544** | 0.491 |
| | Yes | (0.1,0.9) | **0.911** | 0.731 | **0.569** |
| | | (0.3,0.7) | **0.911** | 0.748 | **0.569** |
| | | (0.5,0.5) | **0.914** | 0.729 | **0.557** |
| | | (0.7,0.3) | **0.92** | 0.683 | **0.496** |
| | | (0.9,0.1) | **0.933** | 0.52 | **0.38** |
| 0.5 | No | (0.1,0.9) | **0.898** | **0.660** | 0.489 |
| | | (0.3,0.7) | 0.897 | **0.67** | 0.48 |
| | | (0.5,0.5) | 0.897 | **0.673** | 0.488 |
| | | (0.7,0.3) | 0.905 | **0.639** | 0.498 |
| | | (0.9,0.1) | 0.926 | **0.48** | 0.408 |
| | Yes | (0.1,0.9) | **0.898** | 0.638 | **0.394** |
| | | (0.3,0.7) | **0.898** | 0.662 | **0.396** |
| | | (0.5,0.5) | **0.903** | 0.642 | **0.4** |
| | | (0.7,0.3) | **0.913** | 0.588 | **0.344** |
| | | (0.9,0.1) | **0.93** | 0.439 | **0.285** |
| 0.7 | No | (0.1,0.9) | 0.885 | **0.611** | 0.460 |
| | | (0.3,0.7) | 0.884 | **0.618** | 0.453 |
| | | (0.5,0.5) | 0.886 | **0.611** | 0.454 |
| | | (0.7,0.3) | 0.896 | **0.586** | 0.438 |
| | | (0.9,0.1) | 0.922 | **0.432** | 0.377 |
| | Yes | (0.1,0.9) | **0.887** | 0.593 | **0.324** |
| | | (0.3,0.7) | **0.888** | 0.6 | **0.342** |
| | | (0.5,0.5) | **0.896** | 0.575 | **0.324** |
| | | (0.7,0.3) | **0.91** | 0.52 | **0.261** |
| | | (0.9,0.1) | **0.927** | 0.397 | **0.229** |

Table 5.6: A summary of Table 5.5, showing the mean of the five different weight settings, grouped by the *stress* column

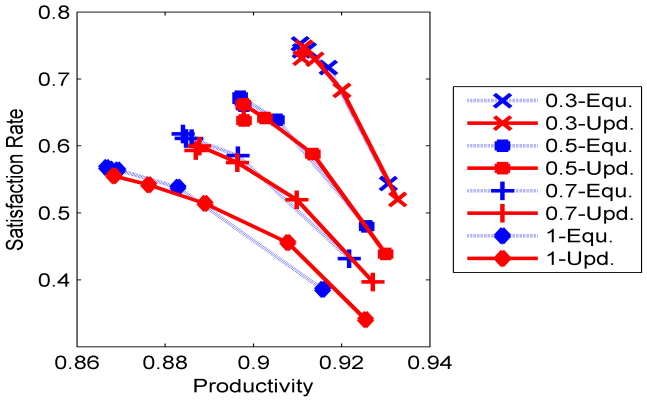
| Updating | *stress* | Productivity | *SatRate* | Variance |
|---|---|---|---|---|
| No | 0.3 | 0.916 | **0.7** | 0.581 |
| | 0.5 | 0.905 | **0.624** | 0.473 |
| | 0.7 | 0.895 | **0.571** | 0.436 |
| | 1.0 | 0.88 | **0.524** | 0.422 |
| Yes | 0.3 | **0.918** | 0.682 | **0.514** |
| | 0.5 | **0.908** | 0.594 | **0.364** |
| | 0.7 | **0.902** | 0.537 | **0.296** |
| | 1.0 | **0.893** | 0.482 | **0.240** |

tivity objective, and the overall satisfaction rate is sacrificed.

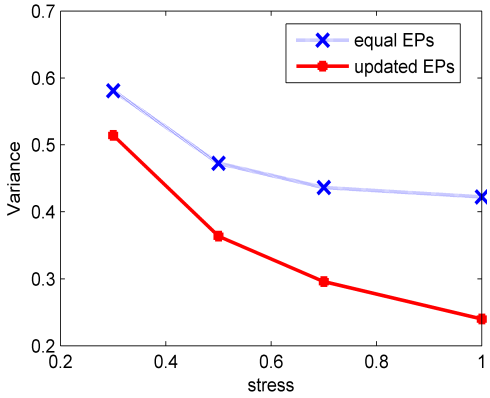
- The updating mechanism is able to significantly reduce the variance in employees' satisfaction levels whatever the value of *stress* is, Figure 5.6(b). However, the difference in the variance between the model *with* and *without* the updating mechanism increases as the plan demand level grows. The updating mechanism reduces the variance of the model by 11% when $stress$=0.3. When $stress$=0.7, the minimization becomes 32%.

- The ratio of the variance of the model with updating to that of without updating is calculated. Figure 5.6(c) shows for each plan demand level (including $stress = 1$), the variance ratio as a function of the weight given to the productivity objective ($w_1$). It suggests that the difference between the ratios exists under different weight settings. It also confirms that the difference in variance decreases (i.e. the ratio approaches 1) as the plan demand level shrinks.

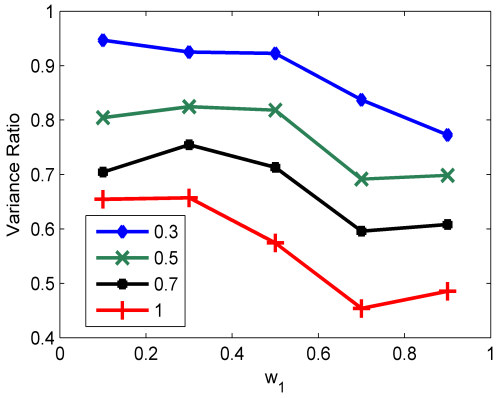
As a conclusion, the results reveal the sensitivity of the model to the plan demand level (*stress*) variable. There is a negative correlation between the plan demand level and the optimality of both the productivity and employees' satisfaction (*SatRate*) rates. The reason for this is that the decrease in the plan demand level would, on




(a) The trade-off between the two objectives for each updating approach and stress level




(b) The variance as a function of the stress




(c) The variance ratio for each stress level, as a function of the weight setting

Figure 5.6: Comparing EmS *with* and *without* the updating mechanism, under different plan demand levels.

one hand, increase the chance of satisfying more employees and, consequently, the employees' satisfaction rate. On the other hand, the smaller the number of plans, the lower the pressure on optimizing the productivity objective will be, and thus the lower the impact of satisfying employees on the productivity objective. As shown in Figure 5.6(a), reducing the *stress* would generate a better (in terms of dominance) trade-off between the productivity and the employees' satisfaction rates. This is an appealing feature, since in real scenarios, relatively low values of *stress* are expected more often.

In addition, the results prove the effectiveness of the updating mechanism which significantly improves fairness by minimizing the variance between employees' satisfaction levels, under any plan demand level. The significance of this improvement grows as the plan demand level increases. This is because the larger the plan demand level, the stronger the competition between employees' plans will be, and then the more influence the updating mechanism will have over the scheduling decision. These findings remain for the examined weight settings, revealing the low sensitivity of the difference in the variance between the model *with* and *without* the updating mechanism to the weights given to the productivity and plan satisfaction objectives.

## 5.5 An Evaluation Summary of EmS

Evaluating an empowerment scheduling model is a three-stage process. The first stage is to ensure that the model satisfies the constitution of empowerment in workforce scheduling (section 5.1.1). Then, the second stage assesses the feasibility of empowerment in terms of its cost impact on the organizational interests. After that, the final stage examines the incorporation of the model to the essential features of an empowerment scheduling model (section 5.1.2). According to this process, the evaluation of EmS is as follows:

1. EmS provides employees with a convincing empowerment practice. From a managerial perspective, it gives each individual the power to express his/her own specific preference on the scheduling decision. From a psychological perspective, employees can feel the power they have since EmS will either explicitly satisfy an employee's preference or acknowledge failure using a transparent compensation policy. Since EmS maintains the managerial and psychological aspects of empowerment, it satisfies the constitution of empowerment scheduling.

2. As stated in section 5.1.3, a feasible empowerment scheduling model should en-

hance employee power while minimizing the impact on the organizational objective. To examine the feasibility of EmS Experiment 1 has been conducted, which assesses the cost of the empowerment practice provided by EmS on the organizational objective. The results show that, in the worst case, the organization will sacrifice a small percentage (about 7%) of its interest to satisfy a significant number of their employees (i.e. about 58% of plans instantiated by all employees). This remark suggests the potential of EmS as a feasible empowerment scheduling model.

3. As an empowerment scheduling model, EmS should incorporate the following four essential features:

   - Simplicity. In EmS employees express their plans using *constraints*, which is a simple language. Furthermore, the weights associated to work plans are controlled by the updating mechanism in an automatic manner, without interference and reasoning from employees. Consequently, EmS presents a simple model that can be utilized by all employees whatever their capabilities or limitations.
   - Flexibility. This feature is maintained by EmS via using the language of *constraints*, which is flexible enough to capture most possible work plans. Examples of such plans are given in Table 5.1.
   - Fairness. EmS employs an automatic updating mechanism that controls the weights associated to work plans. These weights describe the penalties of violating each plan, and thus the priorities of these plans. The effectiveness of this mechanism has been examined in Experiment 3 and Experiment 4, which evaluate the variance between employees in their satisfaction levels. The results reveal the success of this mechanism in reducing the variance between employees' satisfaction levels without impacting the optimality of

the organizational objective. This reduction is significant whatever the plan demand pattern (i.e. the number of employees who have a plan).

- Providing an all-win scenario. In order to evaluate whether enhancing employees' control over the scheduling decision could benefit the organizational objective, Experiment 2 has been conducted to examine one major organizational benefit of empowerment, that being improved productivity. The results show that an improved productivity by 5% reduction in task duration results in an up to 3.8% improvement in the organizational main objective. This appealing outcome represents only one of the several benefits promised by empowerment. Therefore, EmS has the potential to provide an effective win-win model.

As a conclusion of this evaluation process, EmS is an efficient and effective empowerment scheduling model for field workforce scheduling.

## 5.6 Conclusions

Empowerment Scheduling is a term that we introduce to represent our formalization of empowerment in workforce scheduling. It defines an empowerment scheduling model as the one that recognizes employees' individual interests and enhances their feeling of power by reflecting their interests in the final schedule. An efficient model should have four essential features, namely simplicity, flexibility, fairness, and applying a win-win approach.

Following this formalization, we propose an empowerment scheduling model and apply it to FWS. The model enables employees to express their own schedule via constraints as a simple and flexible language. The model reformulates the FWS problem as a biobjective optimization problem, in which the task is not only to look at the organizational objective, but also at employees' interests. An important feature of this

model is how each employee's power over the scheduling decision is defined and managed to maintain fairness within the model. A special updating mechanism is adopted for this purpose.

Several experiments were conducted to evaluate the new model using an adapted GLS algorithm. The results suggest that EmS is a simple, flexible, fair and transparent empowerment scheduling model. They also confirm the feasibility and practicality of the model, providing an all-win scenario for both the employees and the organization.

In this chapter, the underlying multi-objective optimization problem was reduced to a single-objective optimization problem by using a weighted-sum approach, and solved using a single-objective metaheuristic (i.e. GLS). The alternative is to apply a multi-objective optimization approach to this problem, in order to present the trade-offs between the two objectives in a single run. This is pursued in the next chapters in the thesis. In the next chapter, we start by extending GLS to solve problems with multiple objectives. Then, the application of the new algorithm to the EmS-FWS problem is described and examined in chapter 7.

# Chapter 6

# Guided Pareto Local Search

This chapter proposes Guided Pareto Local Search (GPLS), which is an adaptation of GLS for solving multiple objective problems. The motivation of GPLS is to extend the capabilities of GLS to handle senarios with multiple conflicting objectives. In addition, simple GPLS-based frameworks are proposed, which confirm the potential of GPLS to be a central part of advanced multi-phase frameworks.

The organisation of this chapter is as follows. GPLS is described in section 6.1 and its complexity is analysed in section 6.2. Section 6.3 discusses the design of the computational experiments. The experimental results for GPLS are given in section 6.4. Then, GPLS-based frameworks are described in section 6.5, and their experimental results are given in section 6.6. Several parametric analysis studies are reported in section 6.7. Finally, we conclude with section 6.8.

## 6.1 Guided Pareto Local Search

As elaborated in section 3.1, GLS employs a penalty-based approach that guides other single-objective heuristics (including local search algorithms) to escape locally optimum solutions. When local search settles in a local optimum, GLS penalizes some selected

features of the candidate solution, and then restarts the local search method from this solution using an augmented objective function. GLS is comprised of three basic components, namely the guided heuristic, solution features and the penalization scheme. These components should be considered in order to adapt GLS to tackle multi-objective optimization scenarios.

GPLS, that we propose here, extends the guidance approach in GLS to guide PLS to escape Pareto local optimum sets. GPLS modifies the basic components of GLS as follows. First, it applies PLS as the guided multi-objective heuristic. Second, the definition of features has to be derived from all objectives. Thus, it is possible in GPLS to have one feature set if it represents all objectives, or a multiple set of features, each of which is derived from one or more objectives. In GPLS, all Pareto local optima in the maintained *archive* share the same feature set. This leads to a slight modification in the penalization scheme applied by GPLS. Beside the two factors (feature's cost and penalty) that determine the utility of penalizing a particular feature, a third factor is incorporated which employs knowledge extracted from the Pareto local optimum set (i.e. the *archive*). The idea is that the more solutions, in the obtained Pareto local optimum set, that exhibit a particular feature, the greater the utility of penalizing this feature. The details of adapting these components are given in this section.

### 6.1.1 Pareto Local Search (PLS)

GPLS requires a PLS that returns a Pareto local optimum set. As reviewed in section 3.2.4, there have been several variants of PLS in the literature, a representative of which is SteepPLS that is proposed in [95]. As outlined in Algorithm 3.1, a main characteristic of SteepPLS is the applied neighbourhood exploration strategy. It fully explores the neighbourhood of each candidate solution. However, the application of SteepPLS to the biobjective TSP [95, 83] reveals that a major weakness of SteepPLS is its excessive demand for computational time, particularly when the search starts from a

```
Pareto Local Search([g_1, .., g_k],archive)
  if archive = ∅ then
    s_0 ← InitialSolution()
    archive ← s_0
  end if
  fails ← 0
  while ∃s ∈ archive such that Visited(s) = false do
    for each s' ∈ Neighbourhood(s) do
      Evaluate(s', [g_1, .., g_k])
      if s' dominates s then
        s ← s'
        fails ← 0
      else if s does not dominate s' then
        if s' is non-dominated by any solution in archive then
          UpdateArchive(s')
        end if
      else
        fails ← fails + 1
        if fails > maxFails then
          return archive
        end if
      end if
    end for
    Visited(s) = true
  end while
  return archive
```

**Algorithm 6.1:** Pseudo-code for Greedy Pareto Local Search (GreedyPLS)

random solution, and/or the underlying neighbourhood operator generates large neighbourhoods, such as the 3-Opt [95] for TSP. One reason for this is the full exploration of the neighbourhood of a solution, which is much more timely compared to other visiting strategies. Thus, in cases where the time budget is limited, other greedy visiting strategies would be very competitive, particularly for many-objective (i.e. more than two) optimization problems.

Alternatively, a variant of PLS is introduced here, referred to as GreedyPLS, that applies a first improvement strategy and partial neighbourhood exploration. GreedyPLS is depicted in Algorithm 6.1. It works, from a high level perspective, as follows:

1. It starts with a randomly or heuristically generated solution that is added to the *archive*.

2. A candidate solution is randomly chosen from the *archive*, and its neighbourhood is explored while applying a first-improvement strategy (i.e. any better solution found is accepted immediately).

3. The *archive* is updated with any non-dominant neighbour.

4. The exploration of the neighbourhood of a candidate solution stops when the whole neighbours are visited, or the maximum number of fails $maxFails$ is exceeded. The number of fails ($fails$) is incremented every time a neighbour is dominated by the current solution, and it is reset to zero every time a move occurs. When the exploration stops, the current solution is considered as a Pareto local optimum.

5. The current solution is marked as being visited.

6. The procedure continues at step 2, while there is a solution in the *archive* that is not visited.

GreedyPLS is similar to SteepPLS with two differences: (1) GreedyPLS applies a greedy visiting strategy, in which it accepts and moves to the first better neighbour, and (2) a solution in GreedyPLS becomes a Pareto local optimum either when all its neighbours are considered and none of which dominates it, or the maximum number of fails $maxFails$ is exceeded. Without using this parameter, GreedyPLS might be unable to reach a Pareto local optimum set while being given a limited amount of time, and maintaining a large or unbounded size of archive. Therefore, such a parameter defines an earlier stopping criterion, and then a *virtual* Pareto local optimum set. In this case, GPLS can be utilized even within a limited amount of computational time.

The only change that GPLS makes to the underlying PLS is the replacement of each objective function $g_i$ with an augmented objective function $h_i$ during the evaluation of neighbour solutions. The augmented functions are not used in the updating procedure of the *archive*, which always depends only on the original objective functions.

### 6.1.2 Features and their Cost

GPLS defines a set (or multiple sets) of features, which will be shared by all solutions. In multi-objective optimization problems, one should take into consideration two different scenarios:

1. All objectives have the same structure (e.g. putting an item in all knapsacks in the Knapsack problem, or an edge in the multi-objective TSP), and therefore they share one defined feature set. However, the cost of a feature varies according to a particular objective. In order to define the cost of a feature in this case, the influence of the feature on each objective has to be considered and modelled into a single cost function. Such models include, for example, (weighted) aggregating approaches and other general functions such as min, max or mean. Modelling feature's cost is problem-dependant, and thus it is hard to develop a general model. Algorithm 6.2 considers this scenario since the benchmarks used in this chapter are of this class.

2. A distinct feature set needs to be defined for each objective, and a cost is associated with each feature to describe its influence on the corresponding objective. In this case, features from all feature sets should be considered at the penalization phase to pick one or more features to penalize. An example of a problem in this category is the EmS-FWS problem, which is elaborated in chapter 7.

### 6.1.3 Penalization Scheme

The guidance strategy that GLS employs relies on penalizing some features exhibited by the recent local optimum. As described in section 3.1, the novelty of GLS is mainly in the way it selects features to penalize. The target is to penalize "bad features" when the local search settles in a local optimum. Two factors affect the utility of a feature, namely its cost and the frequency of penalizing this feature in previous penalizations.

GPLS deals with a Pareto local optimum set instead of a local optimum. A straightforward penalization strategy is to evaluate the utility of all features exhibited by *any* non-dominant solution in the *archive*, and penalize features with maximum utility (Equation 3.3). However, this simple strategy does not incorporate any information from the *archive* (i.e. the Pareto local optimum set). An example of important information that can be extracted from the *archive* is the number of non-dominant solutions that exhibit a particular feature. This is another factor that can be incorporated into the utility function, such that the more Pareto local optima that exhibit a feature, the greater its utility of penalization. Recall that when a feature is penalized, only those Pareto local optima that exhibit the penalized feature will have the chance to escape. Therefore, increasing the utility of penalizing a feature that occurs in many Pareto local optima would enhance the chance of escaping a Pareto local optimum set by restarting from more solutions. It would also help to prevent the search from directing all its efforts towards any particular region of the PF, which therefore leads to a better spread over the PF. Bad features, in terms of their cost, are hoped to be removed either during the next calls of PLS or by future penalization.

The utility function, as stated in Equation 3.3, is redefined to incorporate the number of solutions in the Pareto local optimum set that exhibit this feature ($\gamma_i$):

$$util_i(archive) = I_i(archive) \times \frac{c_i \times (\gamma_i / |archive|)}{1 + p_i} \tag{6.1}$$

where *archive* is a Pareto local optimum set, $I_i(archive)$ indicates whether "at least" one solution in the *archive* exhibits feature $i$, $c_i$ the feature's cost, $p_i$ the penalty and $|archive|$ the size of the Pareto local optimum set. The feature with the greatest *util* value will be penalized. When a feature is penalized, its penalty value is always incremented by 1.

This utility function redefines the term "bad feature". If a feature is not exhibited in the Pareto local optimum set (indicated by $I_i$), then the utility of penalizing it is 0. The higher the cost of this feature (the greater $c_i$) and the more non-dominant solutions exhibiting it (the greater $\gamma_i$), the greater the utility of penalizing it. Furthermore, the more times that it has been penalized (the greater $p_i$), the lower the utility of penalizing it again.

Having penalized a feature, all solutions in the *archive* that exhibit this feature need to be marked as 'non-visited', so as to be considered by the PLS method in the next iteration.

### 6.1.4 $\lambda$-parameter

The lambda parameter $\lambda$ is the only parameter to GLS which determines the scaling of the penalty. In multi-objective scenarios each objective ideally has its own lambda, which is calculated as a function of a local optimum with respect to the correspondent objective. Thus, GPLS requires a set of lambda parameters: $[\lambda_1, \cdots, \lambda_k]$, where $\lambda_i$ is a parameter for the objective $h_i$. Recall that lambda can be dynamically computed after the first local optimum and before penalties are applied to features for the first time (Equation 3.4).

### 6.1.5 Algorithm Overview

Having discussed its basic components, the GPLS algorithm works as follows (Algorithm 6.2). As inputs, GPLS requires a set of objective functions, a set of $\lambda$-parameters,

a set of features' exhibition indicators and features' costs. In the initialization phase, the penalty of each feature is set to 0, and an augmented objective function $h_j$ is defined for each objective function $g_j$. After that, the underlying PLS algorithm is applied, which uses $h_j$ instead of $g_j$ in the function $Evaluate(s, [g_1, .., g_k])$ as given in Algorithm 6.1, while the *archive* is being updated with respect to the value of $g_j$. Every time PLS settles at Pareto local optimum set, the penalization phase calculates the utility of penalizing each feature. The feature with the maximum utility is penalized by incrementing its associated penalty value $p$. Since all solutions in the *archive* are marked as being *visited* by PLS, those solutions that exhibit the most recent penalized feature are unmarked, so that PLS can restart from those solutions. GPLS repeatedly penalizes and then apply PLS, until a stopping criterion is met. The output of GPLS is a set of non-dominant solutions, maintained by the *archive*.

**Input**: Objective function set: $[g_1, .., g_M]$, lambda parameter set: $\lambda$, Feature's indicator set: $[I_1, \ldots, I_K]$ and Features' cost: $[c_1, \ldots, c_K]$
**Output**: An approximation of the PF: *archive*
$archive = \emptyset$;
**foreach** $i \in [1 \cdots K]$ **do**
  $p_i = 0$;
**end**
**foreach** $j \in [1 \cdots M]$ **do**
  $h_j = g_j + \lambda_j * \sum_{i \in F} p_i * I_i$;
**end**
**repeat**
  $PLS([h_1, .., h_M], archive)$;
  **foreach** $i \in [1 \cdots K]$ **do**
    $util_i = I_i(archive) * (c_i * (\gamma_i / |archive|))/(1 + p_i)$;
  **end**
  **foreach** *i such that* $util_i$ *is maximum* **do**
    $p_i = p_i + 1$;
    **foreach** $s \in archive$ *such that* $I_i(s) = 1$ **do**
      $Visited(s) = false$;
    **end**
  **end**
**until** *StoppingCriterion*;
**return** *archive*;

**Algorithm 6.2:** Pseudo-code for Guided Pareto Local Search

## 6.2 Complexity Analysis

The computational complexity of GPLS is analysed here in comparison with PLS, which is a very simple algorithm compared to other state-of-the-art multi-objective optimization methods, in particular MOEAs. We assume here that PLS is the same as the one employed by GPLS, and thus PLS encompasses GreedyPLS and SteepPLS. We focus on the extra computational complexity added to PLS by GPLS.

GPLS can be divided into two phases: the local search phase and the penalization phase. The local search phase comprises the following three basic operations:

1. The initialization phase. The *archive* is initialized in both PLS and GPLS by adding a solution ($O(1)$). GPLS requires an additional initialisation for the features' penalties where the size of the feature set(s) is $K$ ($O(K)$).

2. Solution acceptance/rejection. PLS requires $1 * M$ comparisons to decide whether to accept or reject the neighbour solution, where $M$ is the number of objectives ($O(M)$). GPLS needs $2 * M$ comparisons ($O(2 * M)$), one uses $h(x)$ and the other uses $g(x)$. A comparison of the $h(x)$ value is applied in the movement decision (i.e. whether to accept, and then move to a neighbour solution), while a comparison of the $g(x)$ value is applied in the *archive* updating decision (i.e. whether to update the *archive* with the new neighbour).

3. Updating the *archive*. GPLS applies the same procedure as PLS for updating the *archive*. This is because GPLS uses only $g(x)$ to compare the new solution with those in the *archive* ($O(M * |archive|)$).

The second phase of GPLS, which PLS does not have, chooses and penalizes selected features from the solutions in the *archive*. This phase includes two basic operations:

1. Calculating features' utilities. GPLS needs to evaluate the utility of penalizing each feature. In the worst case, $K$ evaluations are required ($O(K)$).

2. Marking exhibiting solutions as 'non-visited'. Having penalized a particular feature, each solution in the *archive* that exhibits this feature is unmarked, so as to be considered in the next local search phase. In the worst case, we would have to unmark all solutions in the *archive* ($O(|archive|)$).

In the worst case, a total complexity of GPLS per iteration becomes $O(2 * K + 2 + 2 * M + M * |archive|)$ or $O(M * |archive|)$, where that of PLS is $O(M * |archive|)$.

## 6.3 Experimental Design

In this study, two multi-objective optimization problems are used as benchmarks for examining and analysing the performance of GPLS. The first problem is the 0/1 multi-objective knapsack problem, which is a typical benchmark for a wide range of multi-objective optimization algorithms including MOEAs. The second benchmark is the biobjective TSP, which is a typical benchmark for PLS and its frameworks and hybrids. The programming language used to implement the proposed algorithms is Java. The experiments have been performed on a PC with 3.0 GHz, 3.0 GB Intel Core 2 Duo processor.

### 6.3.1 The Multi-objective 0/1 Knapsack Problem (MOKP)

A multi-objective 0/1 knapsack problem can be described by a set of $n$ items and a set of $m$ knapsacks. Given the capacity of knapsack $j$ ($\zeta_j$), the profit from including item $i$ in knapsack $j$ ($\rho_{ij}$), and the weight of item $i$ according to knapsack $j$ ($\omega_{ij}$), the task in the MOKP is to find a vector $x = (x_1, ..., x_n) \in \{0, 1\}^n$, where $x_i$ describes whether item $i$ is put in all the knapsacks, that aims to:

$$maximize \quad f_j(x) = \sum_{i=1,..,n} \rho_{ij} x_i, \qquad \forall j = 1, ..., m \tag{6.2}$$

$$subject \quad to \sum_{i=1,..,n} \omega_{ij} x_i \leq \zeta_j, \qquad \forall j = 1, ..., m \tag{6.3}$$

This problem was first formulated and solved by Zitzler & Thiele [138]. Since then, the problem has become a standard benchmark for testing multi-objective metaheuristics. They produced nine problem instances with 250, 500, and 750 items, each of which had various numbers of objectives (2, 3 and 4 objectives). These are available (at the time of writing) from an Internet web-site[1].

### 6.3.2 The Biobjective Travelling Salesperson Problem (biTSP)

Given a set of $N$ cities: $v_1, v_2, .., v_N$, and two different cost factors $c^1$ and $c^2$ (which correspond to distance, travel time, cost .. etc.) defined for each pair of cities, the biTSP concerns finding a tour (i.e. an order of the cities) $\pi$ of a minimum costs:

$$minimize \quad f_j(\pi) = \sum_{i=1,..,N-1} c^j_{\pi_i \pi_{i+1}} + c^j_{\pi_N \pi_1}, \qquad \forall j \in 1, 2 \tag{6.4}$$

We use the same set of eight instances of TSP as in [95, 83, 66]. They are constructed from two different single-objective TSP instances having the same number of cities. These instances are obtained from the public single-objective instances available from the TSPLib[2], namely KroA100-KroD100, KroA150-KroB150, and KroA200-KroB200 with 100, 150, and 200 cities.

### 6.3.3 The Implementation of GPLS for the MOKP

The only attempt to apply GLS to knapsack problems, to our knowledge, is reported in [57], where GLS has been applied to a single-objective variant of the multidimensional knapsack problem. In order to apply GPLS to the MOKP, we have defined the following components.

[1] http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/
[2] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

#### 6.3.3.1 Repair Method

The MOKP involves constraints that are to be handled. This can be achieved by defining a heuristic for repairing infeasible solutions. Several repair approaches have been proposed for this purpose [138, 65]. In [138], a straightforward extension to a repair method applied to single-objective knapsack problems was proposed. The repair procedure removes items from the solution step by step until all capacity constraints are fulfilled. Items are ordered increasingly using the maximum profit-to-weight ratio over all objectives per item as the sorting criterion; This describes the order in which the items are deleted.

However, this heuristic favours items with a maximum ratio on a single objective, despite its impact on other objectives. An alternative, fairer heuristic that considers the profit-to-weight ratios with respect to all objectives is proposed here. Instead of using the maximum profit-to-weight ratio per item, we use the summation of the profits divided by the summation of weights for each item over all knapsacks:

$$\frac{\sum_{j=1,..,m} \rho_{ij}}{\sum_{j=1,..,m} \omega_{ij}} \tag{6.5}$$

This repair method works very well with PLS, and yields a slightly better performance over the former one.

#### 6.3.3.2 PLS

Three main components need to be defined in order to apply PLS to the MOKP, the first of which is solution representation. We represent each solution in the search space by a vector $x = (x_1, ..., x_n) \in \{0, 1\}^n$, where $x_i$ describes whether item $i$ is put in all the knapsacks.

The second component is the neighbourhood function that maps a solution to a set of candidates. We use a simple neighbourhood function, by which a new neighbour is

obtained by performing a single flip of any value in the current vector. Since flipping an item from 1 to 0 (i.e. the item is removed) will not yield a better solution, a random non-included item is included (i.e. flipping from 0 to 1) when an item is removed.

The last component of PLS is the initial solution, which can be generated randomly or heuristically. In this study, we use a heuristic initialization by including all items ($x_i = 1$), and then applying the proposed repair method (Equation 6.5).

#### 6.3.3.3 Features and their Costs

A solution to the MOKP decides which items to be put in the knapsacks. Thus, a possible feature to consider for this problem is whether an item $i$ is included. The importance of putting item $i$ in the knapsacks varies between knapsacks (i.e. objectives). In order to define a single value that determines the cost of a feature (i.e. picking an item) on all objectives, the average of the profit-to-weight ratio of an item over all objectives is applied. Therefore, a "bad feature" can be defined as including an item with a overall *low* profit-to-weight ratio (i.e. high weight-to-profit ratio), and then the cost of feature $i$ is defined as follows:

$$c_i = \frac{\sum_{j=1,..,m} (\omega_{ij}/\rho_{ij})}{m} \tag{6.6}$$

The lower the average of the profit-to-weight ratio for the $m$ objectives of feature $i$, the higher the cost of this feature, and thus the more chance this feature is to be penalized.

### 6.3.4 The Implementation of GPLS for the biTSP

The implementation of GLS for the single-objective TSP was extensively investigated by Voudouris and Tsang [125]. We follow the same approach to implement GPLS for the biTSP, as described in the following components:

#### 6.3.4.1 PLS

A solution is represented by a permutation (i.e. vector) which contains the order of the cities in the tour. The solution space of the problem is made of all the possible permutations of the cities. The well-known 2-Opt improvement heuristic is applied. The 2-Opt employs the 2-exchange neighbourhood operator, where a neighbour is obtained from the current tour by deleting two edges and reconnecting the two resulting paths with the only possible way that yields a new tour. Starting from a randomly generated permutation, PLS iteratively searches the neighbourhood of the current solution in a randomly chosen order, and moves to the first better (in terms of Pareto optimality) 2-exchange neighbour.

#### 6.3.4.2 Features and their Costs

A tour in the biTSP includes a number of edges, each edge is associated with two cost factors. Therefore, the set of all edges defines the set of features for the biTSP. Each tour either includes (i.e. exhibits) an edge (i.e. feature) or not. In the biTSP, the cost of each feature is modelled by using the average of the cost of the correspondent edge over all objectives. Therefore, the cost of the feature associated with edge $e_{ij}$ is defined as follows:

$$c_{ij} = \frac{c_{ij}^1 + c_{ij}^2}{2} \tag{6.7}$$

The higher the average of the cost of the $m$ objectives for a feature, the higher the cost of this feature, and thus the more chance this feature has to be penalized.

#### 6.3.4.3 Fast Local Search

As a speed up technique, GLS is always coupled with Fast Local Search (FLS) [128] instead of local search. FLS is a general method used to improve the efficiency of the

local search by guiding the search to consider only those neighbours who are likely to lead to fruitful moves. In order to apply FLS, one should define a way of dividing a neighbourhood set into sub-groups, each of which is associated with an activation bit. Only those sub-groups which are active are examined during the search process. In our case, all 2-exchange operations that involve an index (city) in the permutation, defines a subgroup. Thus, an activation bit is associated with every city. This bit has two possible values: $\{0, 1\}$. FLS works first by initializing all bits to 1 to activate all sub-groups. Then, if all permutations obtained from an index $j$ do not provide a better solution, the activation bit of $j$ is flipped to 0 (i.e. deactivated). This bit can be changed back to 1 (i.e. reactivated) only if $j$ has been involved in an accepted move. At a (Pareto) local optimum, all activation bits in FLS are inactive. In order for PLS to restart the search from a Pareto local optimum after penalizing one of its features, the associated sub-neighbourhoods to the two ends of the penalized edge are reactivated.

## 6.4 Experimental Results for GPLS

### 6.4.1 Comparisons with PLS and Pareto-based MOEAs on the MOKP

The aim of this experiment is three-fold. The first is to compare the performance of GreedyPLS and SteepPLS. The second is to examine the performance of GPLS in comparison with PLS variants. The third is to evaluate the performance of GPLS in comparison with representative Pareto-based MOEAs, namely SPEA, SPEA2 and NSGA2. SPEA was tested on all nine MOKP instances in [138], whereas SPEA2 (which dominates its predecessor) and NSGA2 were tested on the three instances of 750 items in [140]. Their results are available from the same website that the problem instances are obtained from.

Similar to [138, 140], the running lengths allowed for those algorithms are expressed in terms of the total number of evaluations ($maxEval$), and the size of the *archive* is

Table 6.1: Parameter settings for the examined algorithms on the MOKP

| Instance | | $maxEval$ | $\|archive\|$ | $maxFails$ | $\lambda$ |
|---|---|---|---|---|---|
| $n$ | $m$ | | | | |
| 250 | 2 | 75000 | 150 | 15 | 20 |
| | 3 | 100000 | 200 | | |
| | 4 | 125000 | 250 | | |
| 500 | 2 | 100000 | 200 | 15 | 30 |
| | 3 | 125000 | 250 | | |
| | 4 | 150000 | 300 | | |
| 750 | 2 | 125000 | 250 | 20 | 40 |
| | 3 | 150000 | 300 | | |
| | 4 | 175000 | 350 | | |

limited. These are given in Table 6.1. Accordingly, we have adjusted the stopping condition of both PLS and GPLS to be equal to that of other algorithms. For PLS, the algorithm stops when the number of evaluations exceeds $maxEval$. Initial experiments showed that PLS does not settle at a real Pareto local optimum set within the allowed time. Therefore, in order to test GPLS, a virtual Pareto local optimum set is applied by using the $maxFails$ parameter. This works well only for MOKP instances with two objectives, but not those with three and four. This is due to the very restricted time limit. To resolve this, another stopping parameter is defined for the underlying PLS method in GPLS, that is $maxRestarts$ which limits the number of solutions in the *archive* that can be considered and have their neighbourhood explored. In order to maintain an *archive* with a limited size, the crowding distance (Definition 12) is employed by the proposed algorithms as a clustering procedure. To avoid excessive calls of the clustering procedure, the clustering is applied only when the *archive* size reaches a soft limit that is set to 130% of the *archive* size. For the MOKP test instances, we found empirically that all augmented objectives in GPLS can have a similar $\lambda$ parameter. The settings of $maxFails$ and $\lambda$ parameters are given in Table 6.1. The $maxRestarts$ is set to 30% of the *archive* size of each instance. Finally, 30 runs are

performed independently for each algorithm on each test instance.

Due to the nature of MOOPs, multiple performance indices should be used for comparing the performances of different algorithms [67]. The following performance indices, which we use here, are commonly used to compare metaheuristics on the MOKP: Set Coverage (C-metric; Definition 8) and Distance from Representatives in the PF (D-metric; Definition 9). The latter requires a set of uniformly distributed points along the PF, or an upper-approximation of the PF. In this study, the upper-approximation to each MOKP test instance that was proposed in [65] is used here. By using such an upper-approximation, the D-metric could measure both the convergence and the diversity of the algorithm, such that obtaining a lower value requires an approximation that covers the whole PF.

Table 6.2 presents the means of the C-metric values of the final approximations obtained by both variants of PLS (SteepPLS and GreedyPLS) and GPLS, compared to SPEA on instances of size 250 and 500. The means of the C-metric values for the PLS variants and GPLS, compared to SPEA2 and NSGA2 on instances of size 750 are given in Table 6.3. Table 6.4 gives the means and standard deviations of the D-metric values for SteepPLS, GreedyPLS and GPLS. The average CPU time used by each algorithm is given in Table 6.5. From these tables, the following remarks can be made:

Table 6.2: Means of the C-metric values between the proposed algorithms (PLS variants and GPLS) (A) and SPEA (S)

| Instance | | SteepPLS | | GreedyPLS | | GPLS | |
|---|---|---|---|---|---|---|---|
| $n$ | $m$ | C(A,S) | C(S,A) | C(A,S) | C(S,A) | C(A,S) | C(S,A) |
| 250 | 2 | 0.87 | 0.0 | 0.88 | 0.0 | **0.92** | 0.0 |
| | 3 | 0.77 | 0.0 | 0.76 | 0.0 | **0.96** | 0.0 |
| | 4 | 0.84 | 0.0 | **0.85** | 0.0 | 0.82 | 0.0 |
| 500 | 2 | 0.98 | 0.0 | 0.98 | 0.0 | **1.0** | 0.0 |
| | 3 | 0.85 | 0.0 | 0.85 | 0.0 | **0.98** | 0.0 |
| | 4 | 0.98 | 0.0 | 0.98 | 0.0 | **1.0** | 0.0 |

Table 6.3: Means of the C-metric values between the proposed algorithms (PLS variants and GPLS) (A) and SPEA2 (S) and NSGA2 (N)

| Instance | | SteepPLS | | GreedyPLS | | GPLS | |
|---|---|---|---|---|---|---|---|
| $n$ | $m$ | C(A,S) | C(S,A) | C(A,S) | C(S,A) | C(A,S) | C(S,A) |
| | 2 | 0.49 | 0.0 | 0.49 | 0.0 | **0.78** | 0.0 |
| 750 | 3 | 0.73 | 0.0 | 0.73 | 0.0 | **0.93** | 0.0 |
| | 4 | 0.79 | 0.0 | 0.80 | 0.0 | **0.96** | 0.0 |
| $n$ | $m$ | C(A,N) | C(N,A) | C(A,N) | C(N,A) | C(A,N) | C(N,A) |
| | 2 | 0.69 | 0.0 | 0.69 | 0.0 | **0.94** | 0.0 |
| 750 | 3 | 0.72 | 0.0 | 0.71 | 0.0 | **0.92** | 0.0 |
| | 4 | 0.76 | 0.0 | 0.77 | 0.0 | **0.93** | 0.0 |

- Both variants of PLS are superior to the other MOEA algorithms in terms of the C-metric values. They were capable of producing an approximation of the PF with a very good quality. Table 6.2 and Table 6.3 show that the final sets of solutions produced by SteepPLS or GreedyPLS are better in all instances than those of other MOEA algorithms in terms of the C-metric. Actually, none of the MOEA algorithms were able to generate a solution that dominates any solution obtained by PLS variants. The difference between SteepPLS and GreedyPLS is very low, as shown in their D-metric values (Table 6.4). Thus, the statistical significance of the difference of the D-metric values is tested with the t-test at 0.05 level of confidence. This analysis reveals that the differences are insignificant in all instances except instances of four objectives, where SteepPLS is statistically better than GreedyPLS.

- Within a limited amount of time, using the concept of a *virtual* Pareto local optimum set, and then applying GPLS, significantly enhances the performance of PLS in terms of solution quality and diversity. This is revealed in Table 6.2, Table 6.3 and Table 6.4 which show that, with the guidance approach of GPLS, PLS produces final sets of solutions that dominate most of the solutions obtained

Table 6.4: Means (standard deviations) of the D-metric values of SteepPLS, GreedyPLS and GPLS

| Instance | | SteepPLS | GreedyPLS | GPLS |
|---|---|---|---|---|
| $n$ | $m$ | | | |
| 250 | 2 | 233.6(28.5) | 229.31(24.7) | **185.19**(23.9) |
| | 3 | 836.69(26.0) | 833.96(27.5) | **618.57**(31.2) |
| | 4 | 1072.04(20.1) | 1094.59(23.4) | **793.34**(20.1) |
| 500 | 2 | 621.25(39.4) | 611(48) | **435.26**(52.3) |
| | 3 | 2050(28.2) | 2063.24(34.9) | **1664.02**(46.6) |
| | 4 | 2655.49(22.1) | 2670.17(24.9) | **2156.5**(34.4) |
| 750 | 2 | 1282.2(81.5) | 1286.44(63) | **917.2**(65.6) |
| | 3 | 3036.07(28.5) | 3032.31(35) | **2607.02**(43.1) |
| | 4 | 4297.86(31.4) | 4318.03(23.7) | **3740.1**(43.2) |

Table 6.5: Means of the CPU times (in seconds) used by PLS variants and GPLS

| Instance | | SteepPLS | GreedyPLS | GPLS |
|---|---|---|---|---|
| $n$ | $m$ | | | |
| 250 | 2 | 3.71 | 3.69 | 4.25 |
| | 3 | 5.92 | 5.88 | 7.14 |
| | 4 | 10.00 | 9.74 | 11.67 |
| 500 | 2 | 9.98 | 9.96 | 11.51 |
| | 3 | 15.35 | 15.24 | 18.79 |
| | 4 | 23.39 | 22.83 | 29.53 |
| 750 | 2 | 19.84 | 19.33 | 22.22 |
| | 3 | 27.91 | 27.65 | 34.19 |
| | 4 | 43.11 | 39.46 | 55.39 |

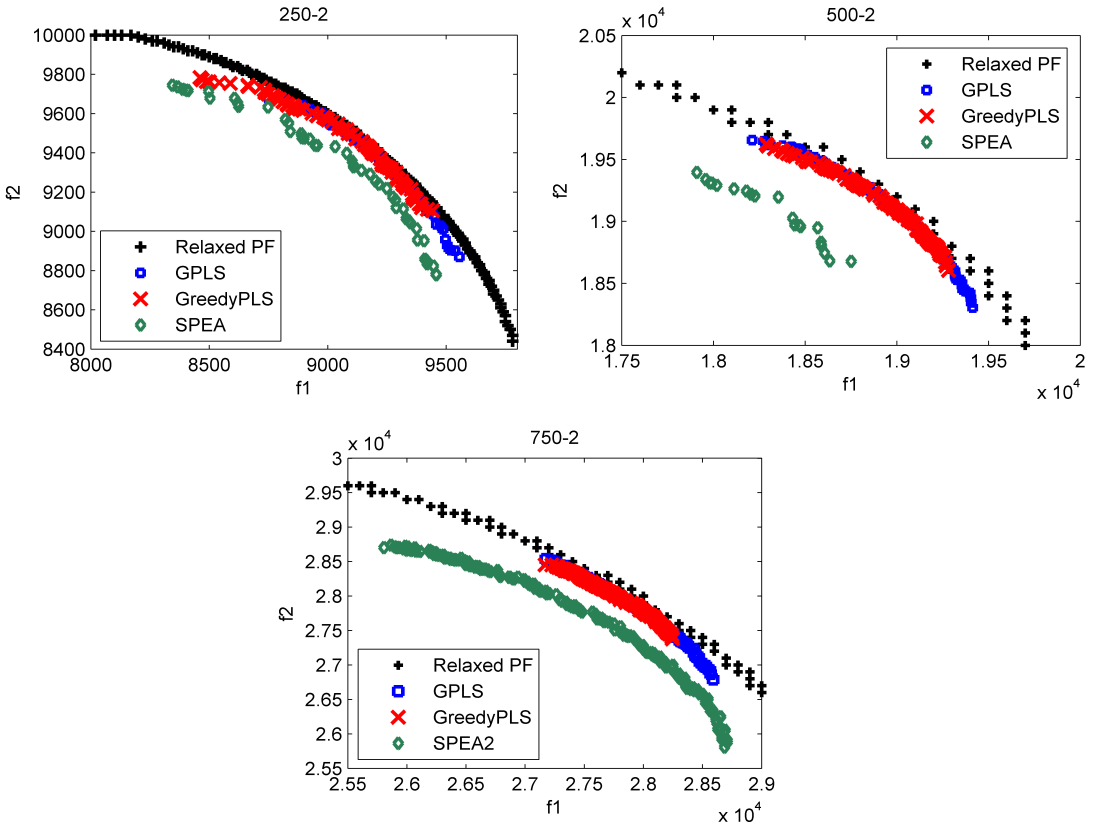



Figure 6.1: Plots of a non-dominant solution set of PLS variants, GPLS and an MOEA algorithm for all the 2-objective MOKP test instances.

by the MOEA algorithms on all instances in terms of the C-metric. Excluding instances 250-4 and 750-2, the solutions of GPLS dominate about 92%-100% of those produced by the MOEA algorithms. Such an obvious domination of GPLS is enough to claim the superiority of GPLS to the MOEA algorithms, without the need to look at other (distance-based) metrics such as the D-metric. Furthermore, GPLS shows its capability of improving the D-metric values of PLS on all instances. The amount of improvement made by GPLS is between 18% and 30%. The significance of such improvements are confirmed using the t-test at 0.05 probability level. The difference between the approximations obtained by GreedyPLS, GPLS and an MOEA algorithm on instances 250-2, 500-2 and 750-2

can be visually detected from Figure 6.1.

- The computational time required by GPLS is about 19% higher than that of PLS variants, given the same number of evaluations. As explained in section 6.2, this is due to the evaluation of the augmented objective functions and the penalization process that GPLS employs. However, the evolution of the D-metric value with the number of evaluations for the MOKP instance 750-2 (as shown in Figure 6.2) importantly indicates that GPLS needs a lower number of evaluations than PLS variants for minimizing the D-metric value. This appealing feature of GPLS clearly suggests that GPLS is more efficient and effective than PLS.

Overall, these remarks suggest that PLS is a very good technique even within a time limit constraint and on many objectives. It is better than well-known MOEA algorithms such as SPEA2 and NSGA2 on the MOKP. We can also claim that GPLS is better overall than PLS in terms of convergence speed and solution quality in this set of test instances.

### 6.4.2 Comparisons with PLS on the biTSP

The aim of this experiment is to evaluate the performance difference between PLS and GPLS on a major benchmark of PLS where it obtains (probably) the most significant results. Paquete et al. [95] studied the application of SteepPLS with different improvement heuristics on the biTSP. They showed that SteepPLS with 3-Opt (i.e. an extending version of 2-Opt where three edges deleted instead of two) can achieve very competitive results in comparison with a representative MOEA, that is the multi-objective genetic local search. Since GPLS employs 2-Opt, a comparison is made only with SteepPLS with 2-Opt (PLS-2Opt). The speed-up technique (i.e. FLS) applied by GPLS allows the (Pareto) 2-Opt to reach a Pareto local optimum set very quickly. Thus, there is no need to define an earlier stopping condition (i.e. $maxFails$) for the

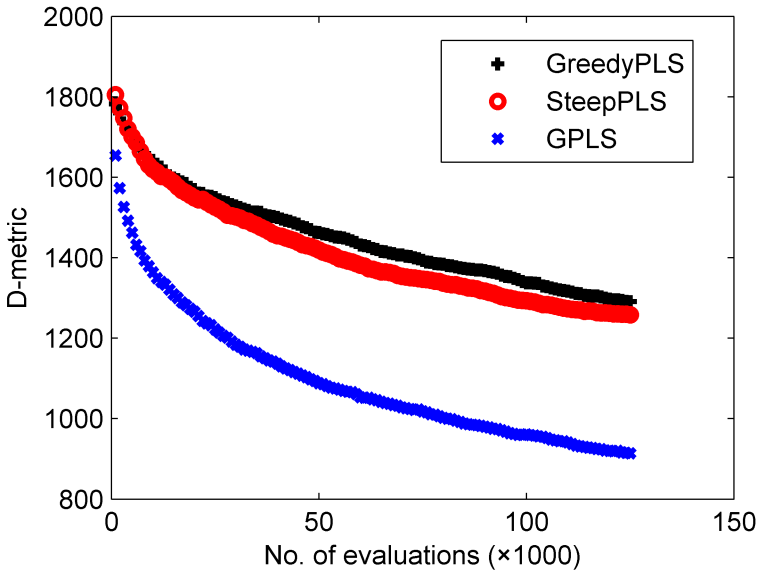



Figure 6.2: An example of the evolution of the D-metric for PLS variants and GPLS on the MOKP

underlying PLS method. The $\lambda$ parameter of GPLS is automatically set as a function of the cost of the first (Pareto) local optimum and a tuning parameter $\alpha$ (Equation 3.4). $\alpha$ is set to 0.3 for all instances.

Similar to [95], we use the same set of seven biobjective instances of TSP. They are KroA100-KroD100 and KroA150-KroB150 with 100 and 150 cities, respectively. In [95], the $R$ values (Definition 10) of PLS-2Opt in these instances are given, and the non-dominant solution sets for PLS-2Opt in all instances except KroAB150 are available on the author's website[1]. These sets are used here to calculate other performance indices, namely the set-coverage (C-metric) and the Hypervolume ($H$; Definition 11). For KroAB150, only the $R$ values are used in the comparisons. As stated in [95], PLS-2Opt requires approximately 60 seconds to find a Pareto local optimum set in instances of 100 cities and 200 seconds in those of 150 cities. Similarly, the computational time of GPLS is limited to these figures by controlling the total number of penalizations.

[1] http://eden.dei.uc.pt/ paquete/tsp/

The results are the mean of 20 executions.

Table 6.6: Means of the values of C-metric ($C$), $R$, $H$ and CPU time (in seconds) values for GPLS and PLS-2Opt; the $C$ value of PLS-2Opt stands for C(PLS-2Opt,GPLS), and vice versa for that of GPLS

| Instance | PLS-2Opt | | | | GPLS | | | |
|---|---|---|---|---|---|---|---|---|
| | $C$ | $R$ | $H(\times 10^8)$ | time | $C$ | $R$ | $H(\times 10^8)$ | time |
| KroAB100 | 0.29 | 0.9339 | 224.5 | 60 | **0.70** | **0.9345** | **224.8** | 54.2 |
| KroAC100 | 0.25 | 0.9312 | 224.3 | 60 | **0.72** | **0.9317** | **225.1** | 56.9 |
| KroAD100 | 0.23 | 0.9334 | 225.9 | 60 | **0.75** | **0.9339** | **226.5** | 53.7 |
| KroBC100 | 0.29 | 0.9351 | 225.9 | 60 | **0.69** | **0.9356** | **226.4** | 58.7 |
| KroBD100 | 0.23 | 0.9335 | 224.5 | 60 | **0.74** | **0.9340** | **224.8** | 54.3 |
| KroCD100 | 0.26 | 0.9378 | 229.2 | 60 | **0.73** | **0.9384** | **229.8** | 49.3 |
| KroAB150 | - | 0.9407 | - | 200 | - | **0.9411** | - | 96.4 |

The results of GPLS compared to those of PLS-2Opt are presented in Table 6.6. The results suggest the superiority of GPLS, demonstrated by the significant improvements in the quality of the obtained Pareto local optimum sets measured by all three metrics in all instances. Solutions obtained by GPLS dominate between 69% and 75% of those obtained by PLS-2Opt, and between 23% and 29% vice versa. In the KroAB100 instance, for example, the $R(H)$ value of the approximation obtained by GPLS is $0.9345(224.8 \times 10^8)$, whereas that of PLS-2Opt is $0.9339(224.5 \times 10^8)$. An interesting feature of GPLS is its ability to obtain a better result compared to PLS-2Opt in the larger instance (KroAB150) within considerably less computational time. The $R$ value of GPLS is 0.9411 within 96.4 seconds, whereas that of PLS-2Opt is 0.9407 within 200 seconds. The t-test (at 0.05 level of significance) confirms that differences in the results of all measures are statistically significant.

In order to evaluate the contribution of GPLS's guidance approach to these results, the evolution of the $R$ value for GPLS is plotted in Figure 6.3(a). The $R$ values are given as a function of the number of penalizations performed by GPLS (i.e. PLS calls). The first $R$ value represents the quality of the first Pareto local optimum set (i.e. the approximation obtained by the standard PLS without penalization). The evolution of

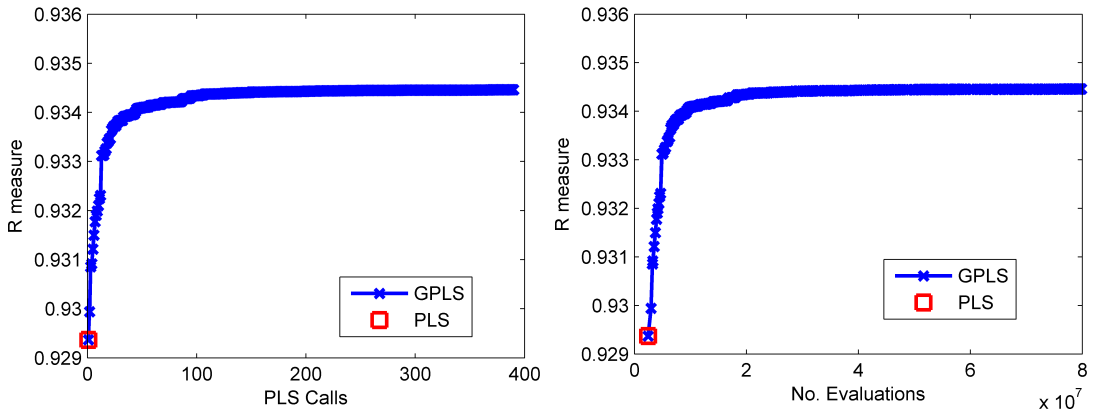



Figure 6.3: An example of the evolution of the $R$ metric for GPLS as a function of: (a) the PLS calls and (b) the number of evaluations

the $R$ metric, as a function of the computational time expressed in terms of the total number of objectives evaluations (i.e. visited solutions), is shown in Figure 6.3(b). Both figures clearly show the important role of the penalization phase that GPLS applies to guide PLS to escape Pareto local optimum sets in an effective way. This guidance procedure improves the performance of PLS in terms of convergence to the PF and diversity in the objective space.

## 6.5 GPLS-based Frameworks

The computational results of applying GPLS to the MOKP proves its effectiveness to converge quite quickly, however, to the middle (i.e. the compromise) area of the PF (for illustration, see Figure 6.1). This feature has a negative impact on the quality of the obtained approximation in terms of diversity over the objective space. On the other hand, several studies suggest the potential of PLS as a central component of a two-phase framework [83, 66] or a hybrid algorithm [12, 73]. Given that GPLS improves the performance of PLS (as shown in the previous section) and that it is a general algorithm

with few parameters to tune, GPLS can also have the potential to be a central part of other frameworks. Such frameworks would further enhance the performance of GPLS, particularly the spread of solutions over the PF.

We aim here to present simple GPLS-based frameworks with two purposes: (1) to improve the performance of GPLS in terms of solution diversity to cover the whole PF, and (2) to confirm the potential of GPLS to be an effective Pareto search technique which can be a basic component in a multi-phase framework. Simple GPLS-based frameworks are proposed here, all of which rely on GPLS as a central searching technique, and none of which requires the modification of GPLS. The idea is to enhance the diversity in the *archive* by heuristically generating an initial diverse set of solutions that approximates the whole PF. This set of solutions is utilized by GPLS in two ways. First, the initial solutions are added to the *archive*, and then GPLS starts normally. Second, multiple runs of GPLS are performed independently, each of which starts from a different solution(s) in this initial set. Thus, these frameworks depend on the generator of the initial solution set. The following describes such a generator, as well as the proposed frameworks.

### 6.5.1 Solution Set Generator ($InitialSetGenerator()$)

One way to obtain a diverse set of initial solutions is to decompose the PF into a number of scalar objective optimization sub-fronts. Then, a local improvement or a heuristic solution generator is applied for each scalar objective. A simple method to perform such a decomposition is to use the weighted-sum approach. In order to cover the PF, uniformly distributed normalized weight vectors are generated (following the approach proposed in [56]) by using all weight vectors in which each individual weight takes one value from: $\{l/k : l \in \{0, .., k\}\}$, where $k$ is a parameter that defines the number of weight levels. For example, when $k = 3$ in a biobjective problem, the set of weights will be: $\{(0/3), (1/3), (2/3), (3/3)\}$. Thus, the sub-fronts are represented by the following

weight settings for the two objectives: $\{(0,1),(1/3,2/3),(2/3,1/3),(1,0)\}$.

### 6.5.2 GPLS with an Initial Diverse Solution Set (iGPLS)

In this framework (Figure 6.4(b)), the *archive* of GPLS is first filled with the initial solution set generated by $InitialSetGenerator()$. Then, GPLS as given in Algorithm 6.2 starts with this filled *archive*. Thus, the difference between GPLS and iGPLS is whether the algorithm starts with an empty or (partially) full *archive*, respectively.

Adding such diverse solutions to the *archive* would influence the search in iGPLS in two ways. First, these initial solutions are marked as *non-visited*, and thus they are candidates for local improvements to approximate the correspondent sub-fronts. Second, since the penalization scheme incorporates knowledge from the *archive* (Equation 6.1), such a diverse set of solutions will have an influence on the utility of penalizing a particular feature. This is attributed to the argument that solutions from different sub-fronts are likely to vary in their features.

The only parameter that this framework adds to GPLS is the size of the initial solution set, controlled by $k$. This depends on the size of the *archive* and the computational time required to generate the initial solution set.

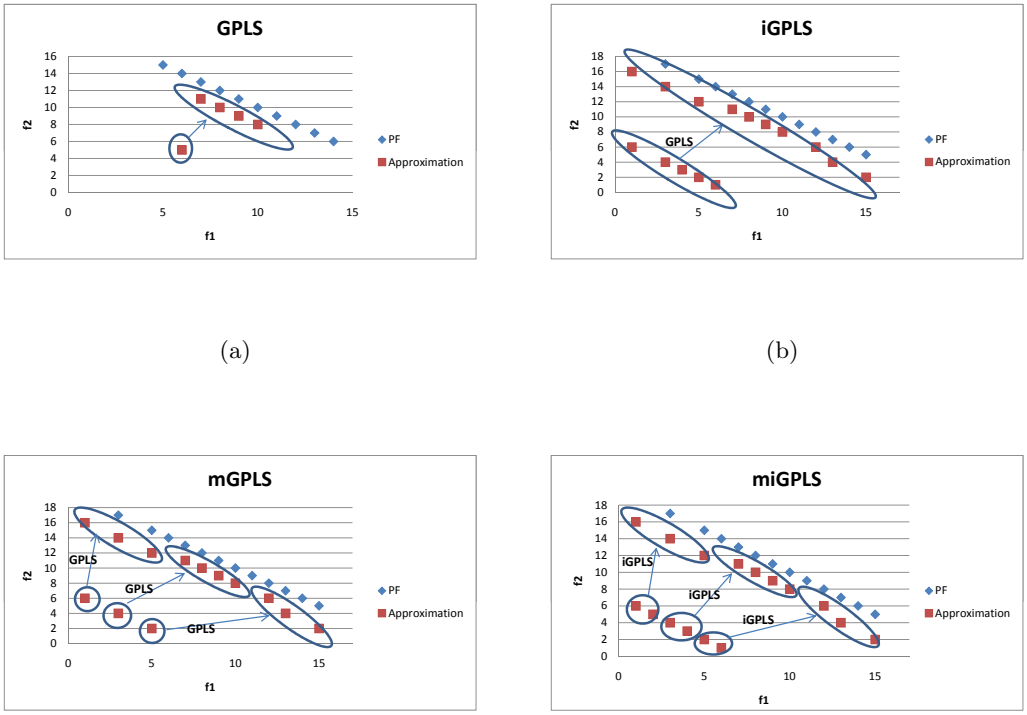
The framework of iGPLS, then, can be simply described as follows:

1. Generate the initial solution set: $InitialSetGenerator()$, mark each generated solution as *non-visited*, and then add them to the *archive*.

2. Apply the GPLS procedure (Algorithm 6.2) using this filled *archive*.

### 6.5.3 Multiple, Parallel GPLS (mGPLS)

Another use of the initial solution set is to perform multiple independent GPLS runs, each GPLS takes a different solution from the initial set as a starting point (Figure 6.4(c)). This is motivated by the speedy convergence of GPLS, as shown earlier (Figure




(a) (b)

(c) (d)

Figure 6.4: Illustrations of the differences between GPLS and its frameworks

6.2). Since the GPLS runs are independent, mGPLS is seen as a parallel version of GPLS. Every GPLS maintains its local *archive*. These local *archive*s are merged into one global *archive* that is to be returned as the final set.

Similar to iGPLS, mGPLS requires one additional parameter to be defined, which is the number of independent runs (i.e. the size of the initial set). This depends on the computational time budget, and requires estimating the time GPLS needs to converge to the PF.

The framework of mGPLS can be described as follows:

1. Generate the initial solution set: *InitialSetGenerator*(), mark each generated

solution as *non-visited*, and then add them to the *archive*.

2. For each solution in the *archive*: apply an independent run of GPLS (Algorithm 6.2) while initializing its local *archive* with this solution.

3. The returned set of Pareto local optima from each GPLS run is added to the global *archive*.

### 6.5.4 miGPLS: A Combination of iGPLS and mGPLS

In mGPLS, each GPLS run starts from a single point that represents a sub-front. Due to its parallel nature, the number of independent runs is expected to be limited. Another (intuitive) approach is to generate multiple solutions that approximate the whole sub-front, and then apply iGPLS instead of GPLS. This approach is called miGPLS (Figure 6.4(d)) which is similar to mGPLS with the only difference being that miGPLS replaces GPLS with iGPLS. Each independent iGPLS starts from a sub-set of the initial solution set, approximating a sub-front of the PF. In this case, miGPLS requires two parameters to be set, namely the size of the initial solution set (controlled by $k$) and the number of independent agents ($|agents|$). Thus, the size of the initial sub-set for each iGPLS run will be equal to $|archive|/|agents|$.

The framework of miGPLS can be described as follows:

1. Generate an initial solution set: $InitialSetGenerator()$, mark each generated solution as *non-visited*, and then add them to the *archive*.

2. Divide the *archive* into $|agents|$ sub-sets.

3. For each sub-set: apply an independent run of iGPLS while initializing its local *archive* with this sub-set.

4. The returned set of Pareto local optima from each iGPLS run is added to the global *archive*.

## 6.6 Experimental Results for GPLS-based Frameworks

The effectiveness of the proposed frameworks is examined here by comparing their performance to that of GPLS on the MOKP and biTSP. Comparisons to state-of-the-art algorithms on both benchmarks are also made. The following describes the conducted experiments and reports on their results.

### 6.6.1 Comparisons on the MOKP

Using the same MOKP instances and performance indicators as in section 6.4.1, this experiment evaluates the performance of GPLS-based frameworks, and compares them to that of GPLS and a state-of-the-art algorithm on the MOKP, namely MOEA/D[1] [136] which is an aggregation-based MOEA. In order to facilitate the comparison with MOEA/D, the experimental settings follow those in [136]. The running lengths allowed for those algorithms to be expressed in terms of the number of calls to the repair method (*maxRepair*), and the size of the *archive* is limited. Table 6.7 gives the setting of these two parameters, together with those related to GPLS as used in the proposed frameworks. It also gives the settings of $k$ for iGPLS and miGPLS which both are similar to that of the decomposition approach employed by MOEA/D. For miGPLS, the number of independent iGPLS runs is set to 10 on all instances. Thus, the initial *archive* is divided into 10 sub-sets, each iGPLS run starts from a different sub-set. For mGPLS, $k$ is set to lower values ($k$ equals 9, 3 and 3 for instances of 2, 3 and 4 objectives respectively) due to the limited amount of time, and hence to the few number of GPLS runs that should be defined. The number of independent GPLS runs in mGPLS is 10 for instances of 2 and 3 objectives, and 20 for that of 4 objectives. The allowed time budget is divided equally among the GPLS runs in mGPLS, and among the 10 iGPLS runs in miGPLS. All reported results are the average of 30 executions.

Table 6.8 presents the means of the C-metric values of the final approximations

[1]MOEA/D's results are available at: http://dces.essex.ac.uk/staff/qzhang/mypublication.htm

Table 6.7: Parameter settings for GPLS and its frameworks on the MOKP

| Instance | | $maxRepair$ | $k(\|archive\|)$ | $maxFails$ | $\lambda$ |
|---|---|---|---|---|---|
| $n$ | $m$ | | | | |
| 250 | 2 | 75000 | 149 (150) | 15 | 20 |
| | 3 | 100000 | 25 (351) | | |
| | 4 | 125000 | 12 (455) | | |
| 500 | 2 | 100000 | 199 (200) | 15 | 30 |
| | 3 | 125000 | 25 (351) | | |
| | 4 | 150000 | 12 (455) | | |
| 750 | 2 | 125000 | 249 (250) | 20 | 40 |
| | 3 | 150000 | 25 (351) | | |
| | 4 | 175000 | 12 (455) | | |

obtained by GPLS, mGPLS, iGPLS and miGPLS, compared to that of MOEA/D in all instances. Table 6.9 compares the means and standard deviations of the D-metric values for all examined algorithms (the D-metric results for GPLS are provided in Table 6.4). The average CPU time used by each framework for each instance is given in Table 6.10. It also shows the CPU time used by MOEA/D as provided in [136]. Although MOEA/D was implemented using a different programming language and tested on a PC with different specifications to that of the proposed algorithms, we find this (unfair) comparison still helpful to give an idea of the order of magnitude[1] of the computational times of these algorithms.

The results confirm the following remarks:

- GPLS-based frameworks significantly enhance the performance of GPLS, in terms of solution quality and diversity. For all instances, mGPLS, iGPLS and miGPLS produce approximations of the PF which have much better values of both the C-metric and D-metric than that of GPLS. Taking the D-metric value for the MOKP instance 750-2 as an example, GPLS obtained on average 917.2, while

[1]The order of magnitude of a number can be defined as the smallest power of ten required to represent that number. It is commonly used to make approximate comparisons between different values.

Table 6.8: Means of the C-metric values of the proposed GPLS-based frameworks, compared to MOEA/D

| Instance | | Algorithm ($A$) | C($A$,MOEA/D) | C(MOEA/D,$A$) |
|---|---|---|---|---|
| $m$ | $n$ | | | |
| 2 | 250 | GPLS | 0.22 | 0.39 |
| | | mGPLS | 0.37 | 0.51 |
| | | iGPLS | 0.58 | 0.32 |
| | | miGPLS | 0.52 | 0.36 |
| | 500 | GPLS | 0.25 | 0.19 |
| | | mGPLS | 0.54 | 0.31 |
| | | iGPLS | 0.78 | 0.11 |
| | | miGPLS | 0.74 | 0.16 |
| | 750 | GPLS | 0.24 | 0.06 |
| | | mGPLS | 0.62 | 0.29 |
| | | iGPLS | 0.75 | 0.17 |
| | | miGPLS | 0.70 | 0.22 |
| 3 | 250 | GPLS | 0.03 | 0.49 |
| | | mGPLS | 0.01 | 0.87 |
| | | iGPLS | 0.02 | 0.8 |
| | | miGPLS | 0.03 | 0.68 |
| | 500 | GPLS | 0.06 | 0.08 |
| | | mGPLS | 0.03 | 0.75 |
| | | iGPLS | 0.03 | 0.68 |
| | | miGPLS | 0.07 | 0.53 |
| | 750 | GPLS | 0.07 | 0.00 |
| | | mGPLS | 0.06 | 0.51 |
| | | iGPLS | 0.24 | 0.25 |
| | | miGPLS | 0.36 | 0.18 |
| 4 | 250 | GPLS | 0.0 | 0.56 |
| | | mGPLS | 0.0 | 0.84 |
| | | iGPLS | 0.0 | 0.85 |
| | | miGPLS | 0.01 | 0.72 |
| | 500 | GPLS | 0.02 | 0.03 |
| | | mGPLS | 0.01 | 0.60 |
| | | iGPLS | 0.03 | 0.48 |
| | | miGPLS | 0.04 | 0.40 |
| | 750 | GPLS | 0.04 | 0.00 |
| | | mGPLS | 0.01 | 0.34 |
| | | iGPLS | 0.10 | 0.22 |
| | | miGPLS | 0.12 | 0.17 |

Table 6.9: Means (standard deviations) of the D-metric values of GPLS-based frameworks and MOEA/D

| Instance | | mGPLS | iGPLS | miGPLS | MOEA/D |
|---|---|---|---|---|---|
| $n$ | $m$ | | | | |
| 250 | 2 | 40.57 (2.8) | **31.4** (1.9) | 33.4 (1.8) | 37.17 (3) |
| | 3 | 261.48 (14) | 210.64 (7.7) | 193.98 (12.8) | **97.75** (7.2) |
| | 4 | 404.33 (8.6) | 372.2 (9.4) | 341.92 (10) | **176.52** (7.3) |
| 500 | 2 | 65.7 (4.8) | **44.47** (2.2) | 46.69 (2) | 79.07 (5.4) |
| | 3 | 585.53 (22.4) | 433.31 (20.7) | 376.69 (14.1) | **270.31** (11.9) |
| | 4 | 872.24 (15.5) | 650.63 (18.7) | 607.28 (24.8) | **431.94** (12) |
| 750 | 2 | 134.27 (7.2) | **100.96** (5.7) | 109.5 (5.1) | 166.04 (13.6) |
| | 3 | 862.05 (20.4) | 466.43 (15.8) | **407.01** (17.3) | 446.12 (19.1) |
| | 4 | 1360.93 (15.7) | 931.66 (25.3) | 887.58 (24.7) | **761.57** (17.3) |

mGPLS, iGPLS and miGPLS obtained 134.3, 100.9 and 109.5 respectively.

- These simple GPLS-based frameworks are very competitive to other state-of-the-art techniques on the MOKP. The results suggest classifying problem instances in two groups:

  1. **Biobjective instances**. As shown in Table 6.4 and Table 6.9, MOEA/D clearly outperforms GPLS in terms of the D-metric on instances of two objectives, whereas GPLS is globally better in terms of the C-metric as given in Table 6.8. These results reveal that GPLS is better than MOEA/D in terms of convergence, however, only to a part of the PF, while MOEA/D is superior in approximating the whole PF. This remark is illustrated in Figure 6.5(a). In comparison with GPLS-based frameworks, MOEA/D is inferior to iGPLS and miGPLS in terms of both metrics, for all biobjective instances. Moreover, apart from instance 250-2, mGPLS also obtains better results than that of MOEA/D in both metrics. For example, 75%(70%) of solutions obtained by MOEA/D on the 750-2 problem instance are covered by those generated by iGPLS(miGPLS), while only 17%(22%) vice versa.

For the same problem instance, the D-metric value obtained by MOEA/D is 166.04, while mGPLS, iGPLS and miGPLS obtain on average 134.3, 100.9 and 109.5, respectively. These differences are statistically significant as confirmed by the t-test at 0.05 probability level. The differences between the approximations obtained by these algorithms on instance 750-2 can be visually detected from Figure 6.5, which plots the middle part of the final sets. Last but not least, both the mean and standard deviation values of the GPLS-based frameworks (particularly iGPLS), as given in Table 6.9, prove the efficiency and stableness of these frameworks on the biobjective instances of MOKP.

2. **Many-objective instances**. The outstanding performance of GPLS-based frameworks degrades on instances of 3 or 4 objectives, in comparison with that of MOEA/D. MOEA/D was able to produce (statistically) higher quality approximations in these instances, as given in Table 6.8 and 6.9. For example, the D-metric value obtained by MOEA/D in instance 250-3 is 97.8, while that of mGPLS, iGPLS and miGPLS are 261.5, 210 and 193.9, respectively. Interestingly, the performance differences between MOEA/D and GPLS-based frameworks shrinks as the problem size grows. On instance 750-3, miGPLS produced a better approximation (D-metric = 407 and C-metric = 0.36) than that of MOEA/D (D-metric = 446.1 and C-metric = 0.18).

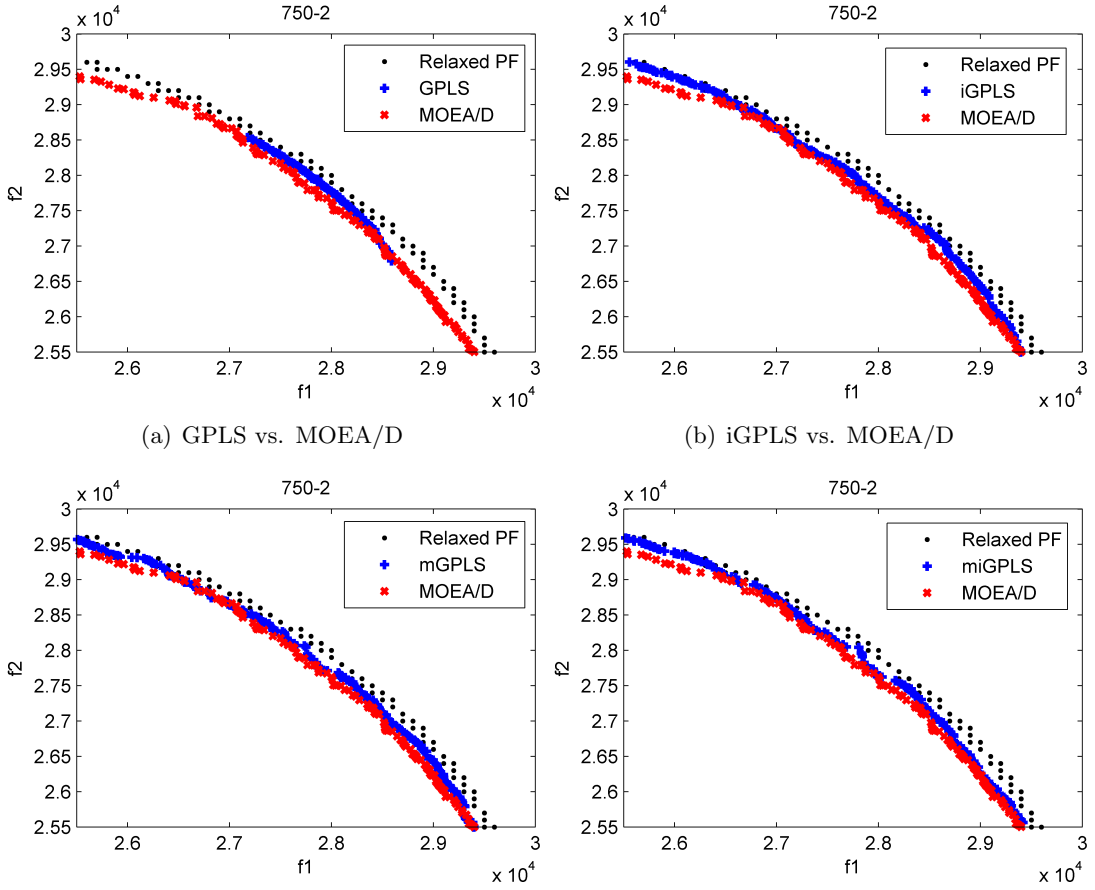
- As given in Table 6.10, the computational times required by miGPLS and iGPLS are almost the same, and both are slightly greater than that of mGPLS. This is attributed to the large size of the initial solution set generated by the *InitialSetGenerator()* in iGPLS and miGPLS. Since solutions are generated using a repair heuristic which is computationally inexpensive, the differences in the computational demand between iGPLS/miGPLS and mGPLS, and between

mGPLS and GPLS are less significant. Given that the different GPLS(iGPLS) runs in mGPLS(miGPLS) are implemented here sequentially, and these runs are independent from each other, the computational times provided in Table 6.10 for mGPLS and miGPLS can be divided by the number of runs (i.e. agents) to have their estimated times when they are implemented in a parallel fashion and executed on a parallel computer. In comparison with MOEA/D, the amount of computational time required by GPLS-based frameworks is close (in terms of the order of magnitude) to that required by MOEA/D even though GPLS is computationally simpler than MOEA/D. This is probably attributed to the complexity of the clustering procedure employed by GPLS and its frameworks in order to maintain a limited archive.

Table 6.10: Means of the CPU time (in seconds) used by GPLS-based frameworks, compared to MOEA/D (as in [136])

| Instance | | mGPLS | iGPLS | miGPLS | MOEA/D |
|---|---|---|---|---|---|
| $m$ | $n$ | | | | |
| 2 | 250 | 4.37 | 4.37 | 4.43 | 3.70 |
| | 500 | 12.10 | 12.08 | 12.20 | 9.40 |
| | 750 | 23.71 | 23.01 | 23.65 | 17.87 |
| 3 | 250 | 7.79 | 7.53 | 7.95 | 6.53 |
| | 500 | 21.60 | 19.46 | 21.36 | 15.30 |
| | 750 | 41.46 | 35.46 | 40.51 | 26.73 |
| 4 | 250 | 12.38 | 11.47 | 12.81 | 19.60 |
| | 500 | 34.23 | 29.49 | 34.17 | 45.17 |
| | 750 | 65.93 | 53.27 | 63.83 | 70.93 |

Overall, these remarks confirm the potential of GPLS to be an effective Pareto search algorithm in multi-phase frameworks. They also suggest the effectiveness of the proposed GPLS-based frameworks, with which GPLS is capable to produce competitive approximations to state-of-the-art algorithms on the biobjective instances of MOKP. Such performance can be attained with relatively low CPU times when mGPLS and




(a) GPLS vs. MOEA/D

(b) iGPLS vs. MOEA/D

(c) mGPLS vs. MOEA/D

(d) miGPLS vs. MOEA/D

Figure 6.5: Plots of a non-dominant solution set obtained by each of GPLS and its frameworks, compared to MOEA/D on the MOKP instance 750-2

miGPLS frameworks are implemented on parallel computers. Furthermore, the results reveal that the outstanding performance of GPLS-based frameworks decreases on MOKP instances with many-objective.

### 6.6.2 Comparisons on the biTSP

The aim of this experiment is to examine the performance of GPLS-based frameworks on the biTSP, in comparison with GPLS. They are also evaluated by making comparisons with two state-of-the-art algorithms proposed for solving the biTSP. The first al-

gorithm is the Two-Phase Pareto Local Search (2PPLS) proposed by Lust and Teghem in [83]. In the first phase of 2PPLS, a good approximation of the PF is generated using one of the best heuristics for the single-objective TSP: the Lin-Kernighan heuristic (LK). Then, the second phase employs a PLS algorithm that starts from the initial approximation generated in the first phase. As shown in [83], 2PPLS produced better results than other state-of-the-art methods and competitive results to the Pareto Memetic Algorithm (PMA) which is proposed by Jaszkiewicz and Zielniewicz in [66]. PMA is a complex hybrid of a memetic algorithm (the first phase) and PLS (the second phase). The complexity of PMA is attributed to its first phase, which employs several techniques such as the LK, path relinking and tabu search. In this study, these two methods are considered and comparisons to them are made. The t-test at 0.05 level of confidence is applied to analyse the significance of the differences between different algorithms. This requires the detailed results of 2PPLS and PMA. These are publicly available only for 2PPLS[1] which are used in the statistical analysis.

The parameter settings of GPLS are similar to that used in section 6.4.2. For GPLS-based frameworks, each scalar function in the $InitialSetGenerator()$ is optimized using an improvement heuristic, that is the GLS algorithm that was proposed in [125] for solving the single-objective TSP. The settings of GLS are given in Table 6.11. In this experiment, there is no explicit time limit defined for the proposed GPLS-based methods. Rather, an absolute stopping condition is defined as a function of a pre-set maximum number of dominants $maxDoms$. The number of dominants ($count$) is incremented every time a neighbour is dominated by a solution in the archive, and it is reset to zero every time a new solution is added to the archive. The settings of $maxDoms$ together with GPLS-based frameworks' parameters are given in Table 6.12.

Similar to [83, 66], we use the same eight biobjective instances of TSP. They are KroA100-KroD100, KroA150-KroB150, and KroA200-KroB200 with 100, 150, and 200

[1] http://sites.google.com/site/thibautlust/research/multiobjective-tsp

Table 6.11: Parameter settings for GLS on the biTSP

| Parameter | Value |
|---|---|
| $\lambda$ | *cost of first local optimum* $\times$ $\alpha$ $\div$ *number of features* |
| $\alpha$ | 0.3 for all instances |
| stopping condition | number of penalizations equals 10000, 15000 and 35000 for instances of size 100, 150 and 200 respectively |

Table 6.12: Parameter settings for GPLS and its frameworks on the biTSP

| Parameter | Value |
|---|---|
| $\lambda$ | 70 for all instances |
| $maxDoms$ | 5M, 10M and 15M for instances of size 100, 150 and 200 respectively |
| $k$ | 9 for mGPLS and 49 for iGPLS and miGPLS |
| $\|agents\|$ | 10 independent runs for mGPLS and miGPLS |

cities, respectively. Two unary performance indices are used in this study: the $R$ measure and hypervolume $H$. While the values of both indicators obtained by 2PPLS are reported in [83], only the $R$ values are available for PMA [66]. All results are the mean of 20 independent runs.

Table 6.13 presents the means of the $R$ values of the final approximations obtained by GPLS, mGPLS, iGPLS and miGPLS, compared to that of 2PPLS and PMA in all instances. Similarly, Table 6.14 presents the means of the $H$ values obtained by these methods, except PMA, in all instances. The average CPU time used by the proposed algorithms to obtain such approximations are reported in Table 6.15. The results indicate the following:

- GPLS produces good quality results, which are significantly improved by mGPLS, iGPLS and miGPLS. For all instances, GPLS-based frameworks were capable of

Table 6.13: $R$ values of GPLS and its frameworks, compared to 2PPLS and PMA

| Instance | GPLS | mGPLS | iGPLS | miGPLS | 2PPLS | PMA |
|---|---|---|---|---|---|---|
| KroAB100 | 0.934613 | 0.935189 | 0.935263 | **0.935280** | 0.935259 | **0.935280** |
| KroAC100 | 0.931891 | 0.932452 | 0.932508 | **0.932526** | 0.932513 | 0.932521 |
| KroAD100 | 0.934057 | 0.934539 | 0.93463 | **0.934641** | 0.934623 | 0.934617 |
| KroBC100 | 0.935738 | 0.936179 | 0.93622 | **0.936229** | 0.936215 | 0.936221 |
| KroBD100 | 0.934129 | 0.934750 | 0.934808 | **0.934811** | 0.9348 | 0.934809 |
| KroCD100 | 0.938511 | 0.939104 | 0.939155 | **0.939174** | 0.939158 | 0.939166 |
| KroAB150 | 0.941439 | 0.941938 | 0.942108 | 0.942108 | **0.942127** | 0.942081 |
| KroAB200 | 0.944363 | 0.944738 | 0.945046 | 0.94504 | **0.945067** | 0.943312 |

Table 6.14: $H$ values ($\times 10^8$) of GPLS and its frameworks, compared to 2PPLS

| Instance | GPLS | mGPLS | iGPLS | miGPLS | 2PPLS |
|---|---|---|---|---|---|
| KroAB100 | 225.16 | 225.98 | 226.1 | **226.11** | **226.11** |
| KroAC100 | 225.48 | 226.23 | 226.31 | **226.32** | **226.32** |
| KroAD100 | 226.8 | 227.35 | 227.41 | **227.42** | 227.41 |
| KroBC100 | 226.66 | 227.32 | 227.37 | **227.38** | **227.38** |
| KroBD100 | 225.17 | 226.07 | 226.12 | **226.13** | 226.12 |
| KroCD100 | 230.03 | 230.83 | 230.88 | **230.89** | **230.89** |
| KroAB150 | 589.99 | 591.88 | 592.47 | 592.46 | **592.51** |
| KroAB200 | 1071.42 | 1073.74 | 1075.99 | 1075.97 | **1076.08** |

improving the performance of GPLS in both performance metrics. In instance KroAB100, for example, the $R$ value of approximations obtained by GPLS is on average 0.934613, where that of mGPLS, iGPLS and miGPLS are 0.935189, 0.935263 and 0.935280 respectively.

- Among GPLS-based frameworks, miGPLS stands out as the best performing algorithm for all instances on both measures. iGPLS is the second best performing framework, producing competitive results to that of miGPLS. Both frameworks are better than mGPLS, which emphasizes the contribution of the size of the initial set of solutions to the performance of the GPLS-based frameworks in the biTSP. The t-test confirms that the differences in the results between these algo-

rithms are statistically significant, except the difference in the $R$ values between iGPLS and miGPLS on KroBD100, where the differences are statistically insignificant.

- The comparison with 2PPLS reveals that miGPLS statistically outperforms 2PPLS in instances of size 100. miGPLS obtains better results in terms of the $R$ measure in all the six instances, and not worse (if not better) on the $H$ measure. In the same instances, iGPLS is statistically inferior to 2PPLS with respect to the $H$ measure, and competitive (i.e. better in 4 instances: KroAB100, KroAD100, KroBC100 and KroBD100, and worse in 2 instances: KroAC100 and KroCD100; the differences in KroAB100 and KroCD100 are statistically insignificant) in terms of the $R$ measure. In the larger instances (i.e. 150 and 200), 2PPLS produces statistically better results in both metrics than both iGPLS and miGPLS.

- When compared with PMA using the $R$ measure, miGPLS shows its superiority to PMA in all instances apart from the KroAB100 where both are equal. iGPLS, on the other hand, was capable of competing with PMA by obtaining competitive results on 3 (i.e. KroAD100, KroBC100 and KroBD100) out of 6 instances, and better values on instances of size 150 and 200. Interestingly, GPLS and mGPLS are both able to outperform PMA in KroAB200.

- In this study, no time restriction has been imposed on GPLS and its frameworks. Table 6.15 shows that iGPLS used the least amount of computational time among the proposed methods in instances of size 100 and 150, and mGPLS for that of 200. Running times of GPLS are higher than those of GPLS-based frameworks. Again, recall that mGPLS and miGPLS are not implemented in a parallel way, and thus their demands for running time are probably 10% of the reported figures when they are executed on parallel computers, as both consist of 10 independent agents. In comparison with other state-of-the-art methods, the computational

times of GPLS and its frameworks are much higher than those of 2PPLS and PMA. This is attributed to the effectiveness of the first phase of both 2PPLS and PMA, which is capable of producing an initial approximation with good quality at less computational time, compared to the multiple runs of single-objective GLS used by $InitialSetGenerator()$ in GPLS-based frameworks.

Table 6.15: Means of the CPU times (in seconds) that are required by GPLS and its frameworks, compared to that of 2PPLS and PMA as reported in [83] and [66], respectively

| Instance | GPLS | mGPLS | iGPLS | miGPLS | 2PPLS | PMA |
|---|---|---|---|---|---|---|
| KroAB100 | 1294 | 688 | 330 | 916 | 36 | 67 |
| KroAC100 | 1004 | 676 | 324 | 885 | 30 | 67 |
| KroAD100 | 869 | 606 | 294 | 802 | 26 | 69 |
| KroBC100 | 948 | 649 | 326 | 896 | 36 | 73 |
| KroBD100 | 1223 | 654 | 319 | 890 | 37 | 74 |
| KroCD100 | 767 | 587 | 302 | 778 | 28 | 73 |
| KroAB150 | 2344 | 839 | 694 | 1296 | 91 | 223 |
| KroAB200 | 2980 | 1299 | 1451 | 2353 | 212 | 567 |

Overall, these remarks confirm the effectiveness of GPLS-based frameworks. This is demonstrated by their ability to enhance the performance of GPLS and compete with state-of-the-art methods on the biTSP. Although a GPLS-based framework requires a high computational time to obtain such outstanding performance, this issue can be resolved by employing more efficient techniques tailored to the TSP in the first phase of these frameworks in order to minimize the times spent finding an initial high quality approximation.

## 6.7 Parametric Analysis

As shown in sections 6.4 and 6.6, experimental results for GPLS and its frameworks, on two standard benchmarks for multi-objective optimization, confirm the ability of

GPLS to be an effective optimization engine which can replace PLS. In order to have more insight into GPLS and its frameworks, further analysis of their components and parameters are required.

### 6.7.1 GPLS

#### 6.7.1.1 Sensitivity to modelling feature cost

As described in section 6.1, a basic component of GLS, that is adapted by GPLS for solving MOOPs, is the definition of feature cost. The costs of a feature on different objectives are expected to vary, and thus a model is required to define a single cost value that considers these different costs. This was modelled as the average of costs over all objectives in both the the MOKP and biTSP. To study the sensitivity of GPLS to this model, an alternative model is defined and compared to the current employed model. The MOKP is considered here as the benchmark. Instead of the average, a new model is defined that chooses the maximum value (i.e. the worst weight-to-profit ratio) among all costs instead of their average. The implementation of GPLS for the MOKP (section 6.3.3) is changed accordingly and tested on instance 750-2. The parameter settings are the same as in section 6.4.1, and the performance is measured using the D-metric. The results show that the D-metric value for the *average* model is 917.2, and that of the *max* model is 905.1. However, the t-test with 95% confidence level reveals that this difference is statistically insignificant. This suggests that GPLS is not sensitive to which model is used to define feature cost, particularly if this model somehow considers the different costs.

#### 6.7.1.2 Contribution of the new penalization scheme

The second major component that is adapted by GPLS is the definition of the utility function of penalizing a feature. The utility function used by GLS is modified in order to consider a Pareto local optimum set instead of a single local optimum, Equation

6.1. To study the influence of this modification to the performance of GPLS, we have tested a GPLS variant that applies the standard utility function of GLS (Equation 3.3) on the MOKP instance 750-2, and compared it to the proposed GPLS that applies the new utility function. The results show that the D-metric value obtained by our proposed utility function is 917.2, which is better than that of the GLS's utility function which obtains on average 949.6. The t-test at 0.05 probability level confirms that the average difference is statistically significant. Thus, the new proposed penalization scheme improves the performance of GPLS.

### 6.7.2 GPLS-based Frameworks

#### 6.7.2.1 Contribution of GPLS to the proposed frameworks

In order to confirm that GPLS is key to the success of the proposed frameworks, the contributions of GPLS to mGPLS, iGPLS and miGPLS are analysed by applying the frameworks to PLS, instead of GPLS. The new frameworks, then, are referred to as mPLS, iPLS and miPLS. The PLS method described in Algorithm 6.1 is used here, however, without the $maxFails$ parameter. The means of the D-metric values on the MOKP instances 250-2, 500-2 and 750-2 for PLS-based frameworks are given in Table 6.16. Comparing these results to that of GPLS-based frameworks (Table 6.9) suggests that GPLS significantly improves the performance of both iGPLS and miGPLS on all instances over that of iPLS and miPLS. For mGPLS, the contribution of GPLS becomes significant in larger instances, i.e. 500-2 and 750-2. For instance, the D-metric values obtained by iGPLS, mGPLS and miGPLS in the 750-2 instance (they are 101, 134 and 109 respectively) are about 30% better than those obtained by iPLS, mPLS and miPLS (they obtained 144, 217 and 156 respectively). These differences between PLS-based frameworks and GPLS-based frameworks are statistically significant, except those between mGPLS and mPLS on 250-2, and miGPLS and miPLS on 250-2.

Table 6.16: Means (standard deviations) of the D-metric values of the PLS-based frameworks

| Instance | iPLS | mPLS | miPLS |
|---|---|---|---|
| 250-2 | 34.32(2.27) | 39.74(2.44) | 34.25(2.04) |
| 500-2 | 54.76(4.75) | 78.85(5.92 | 57.02(3.83)) |
| 750-2 | 144.45(11.83) | 217.34(12.27) | 156.07(13.89) |

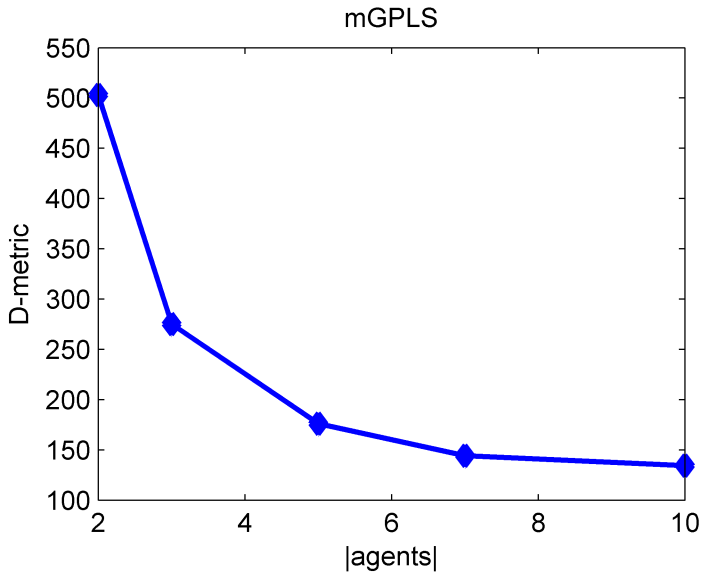



Figure 6.6: The D-metric values as a function of the number of independent agents ($|agents|$) for mGPLS

#### 6.7.2.2 Sensitivity of GPLS-based frameworks to $k$ and the number of agents ($|agents|$)

The major parameter of GPLS-based frameworks is $k$, which reflects the size of the initial approximation of the PF generated by $InitialSetGenerator()$ as described in section 6.5.1. For mGPLS and miGPLS, the number of agents ($|agents|$) is another parameter to be defined. This always equals the size of the initial set of solutions in the case of mGPLS. In this study, the sensitivities of iGPLS and mGPLS to $k$ and miGPLS to $|agents|$ on the MOKP instance 750-2 are tested while varying the values

of these parameters. The results are as follows:

- **mGPLS:** The D-metric values for mGPLS as a function of $k$ (and consequently $|agents|$ which is equal to $k+1$ on biobjective instances) is plotted in Figure 6.6. The considered values of $k$ are: $k \in \{1, 2, 4, 6, 9\}$. The time budget is fixed for all cases, and it is divided equally among the different agents. The figure clearly shows the sensitivity of the performance of mGPLS to the parameter $k$, particularly at low values (i.e. $k \leq 4$). This confirms that, similar to PLS, it is important for GPLS to start with an initial high quality approximation. The better the initial approximation in terms of diversity in the objective space, the better the performance of GPLS. Moreover, the result encouragingly supports the idea of decomposing the PF into multiple sub-fronts. Each sub-front can be considered as a sub-problem and solved independently from the others, with or without cooperation between sub-problems. This is demonstrated by the D-metric value obtained by mGPLS with 10 independent agents, which is 73% better and about five times faster (if it is implemented in a parallel environment) than that of mGPLS with 2 agents.

- **iGPLS:** The D-metric values for iGPLS as a function of $k$ is plotted in Figure 6.7, where $k \in \{1, 5, 10, 20, 50, 100, 249\}$. It shows that iGPLS is sensitive to $k$ only at very low values of $k$, such that $k \leq 10$. At higher values of $k$, the computational overhead of generating more initial solutions is much more than its benefit in terms of the quality of the final approximation obtained by iGPLS. For example, increasing the $k$ by 149% (i.e. from 100 to 249) improves the performance of iGPLS by 5% only. This amount of improvement might be justifiable in the MOKP, where generating initial solutions heuristically is computationally inexpensive. However, in other problems such as the biTSP, generating such huge number of initial solutions heuristically is impractical.

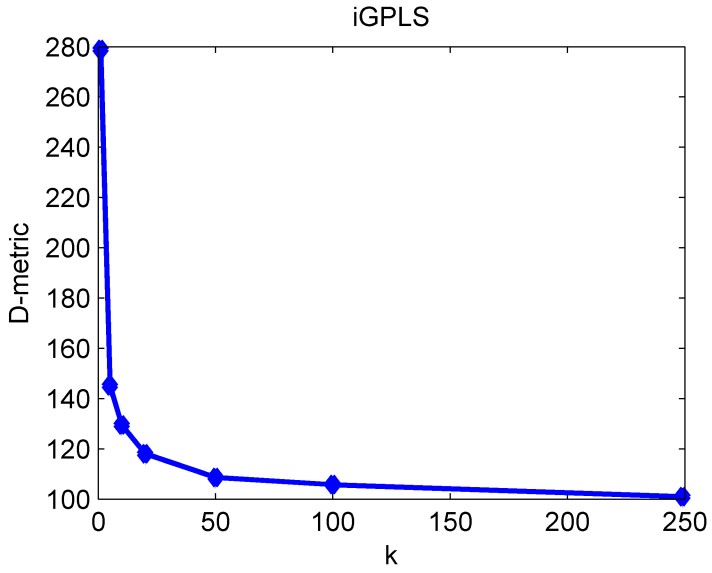



Figure 6.7: The D-metric values as a function of $k$ for iGPLS

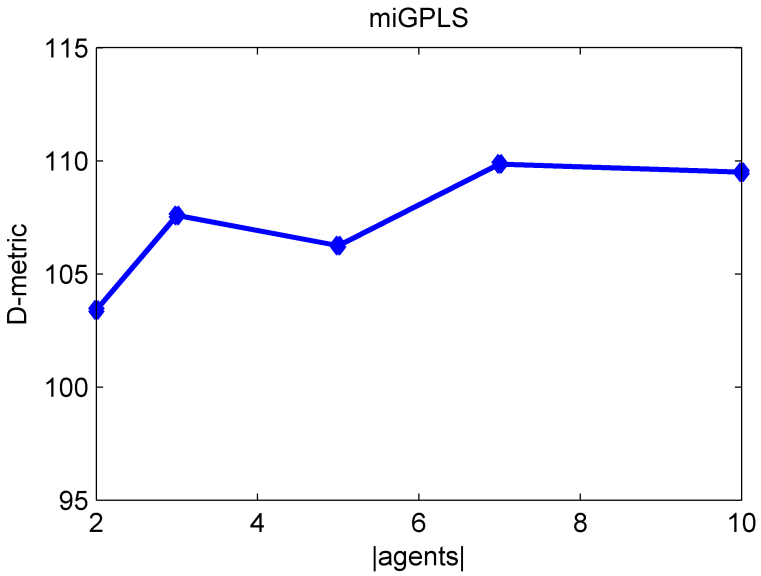



Figure 6.8: The D-metric values as a function of the number of independent agents ($|agents|$) for miGPLS

- **miGPLS:** The means of the D-metric values for miGPLS as a function of $|agents|$ are plotted in Figure 6.8, where $|agents| \in \{2, 3, 5, 7, 10\}$. The value of $k$ was set to 249. The figure shows that increasing the number of iGPLS runs does not necessarily result in a significant impact on the performance of miGPLS. This confirms the potential of decomposing the PF into sub-fronts (i.e. sub-problems). Dividing the initial set of iGPLS into multiple sets, and then applying an independent iGPLS from each sub-set, has no significant impact on the performance of iGPLS. In fact, this would greatly reduce the computational time when miGPLS implemented in a parallel fashion.

## 6.8 Conclusions

A Guided Local Search based multi-objective metaheuristic (GPLS) is proposed here. GPLS confirms the ability of GLS to guide PLS to escape Pareto local optima. GPLS adapts three aspects of the single-objective GLS: (1) the underlying local search which is replaced by PLS that searches for Pareto local optima, (2) the definition of features and their costs in order to consider the influence of features on different objectives, and (3) the penalization scheme which is modified to incorporate knowledge from the archive. Computational experiments show that GPLS is capable of producing better approximations than PLS and representative Pareto-based MOEAs.

As mentioned in section 3.2.4, there has been studies suggesting the potential of PLS to be a major component in a multi-phase framework or in a hybrid algorithm. This has also been proven for GPLS, which is the central method for the three frameworks proposed here, namely iGPLS, mGPLS and miGPLS. iGPLS is a two-phase method which combines GPLS (the second phase) with an initial approximation generator (the first phase). mGPLS is a parallel version of GPLS, where multiple independent GPLS runs are performed, with each run approximating a different sub-front. miGPLS is a

combination of the aforementioned methods. It performs multiple independent iGPLS runs to approximate multiple sub-fronts. Computational results on standard benchmarks demonstrate the effectiveness of these GPLS-based frameworks (particularly iGPLS and miGPLS) in obtaining high quality approximations that yield state-of-the-art results on the MOKP and biTSP.

Several parametric analysis studies for GPLS and its frameworks have been conducted. The major conclusions revealed by these studies are as follows:

1. The new proposed penalization scheme significantly improves the performance of GPLS over the standard one employed by GLS.

2. GPLS is a key element that improves the performance of the proposed frameworks over other state-of-the-art methods.

3. The performance of GPLS-based frameworks are sensitive to the size of the initial approximation (i.e. how well this set represents the whole PF). Interestingly, the significance of this sensitivity appears only at low sizes of the approximation. At higher sizes, generating more initial solutions is not critical to the performance of GPLS-based frameworks.

4. For mGPLS, the more GPLS runs (i.e. the more sub-fronts), the better the quality of the final approximation. For miGPLS, increasing the number of iGPLS runs has no significant impact on the performance of miGPLS. These are appealing features of GPLS-based frameworks when they are implemented in a parallel way and executed on parallel processors. The computational time then will diminish dramatically.

The outstanding performances of GPLS and its frameworks are mainly on biobjective problems. Their performances on many-objective problems are less significant, particularly when compared to other state-of-the-art methods. In addition, the two

standard benchmarks used to test GPLS and its frameworks exhibit a similar feature, that is the convex shape of their PFs. Fortunately, our target multi-objective optimization problem in this thesis, that is the EmS-FWS problem, is a biobjective optimization problem and initial experiments show that the problem instances tend to have a convex PF. These features encourage the application of GPLS to the EmS-FWS problem, which is presented in the following chapter.

# Chapter 7

# The Application of GPLS to EmS

The empowerment scheduling model (EmS), that is proposed in chapter 5 and applied to the FWS problem, was treated as a single-objective optimization problem by aggregating the productivity rate (i.e. organizational objective) and employees' plans satisfaction (i.e. empowerment objective) into one objective function using a weighted-sum approach. In this chapter, the EmS-FWS problem is treated as a multi-objective optimization problem, and solved using GPLS (chapter 6) as a solution technique.

This chapter is organized as follows. Section 7.1 describes the multi-objective nature of EmS. Then, the applications of GPLS and its frameworks to the EmS-FWS problem are discussed in section 7.2. Section 7.3 explains the experimental settings, and section 7.4 discusses the experimental results. The conclusions are given in section 7.5.

## 7.1 EmS-FWS: A Multi-objective Optimization Problem

As described in chapter 5, EmS is a flexible management model for workforce scheduling that involves employees in the allocation decision. It enables employees to plan their own schedules. Employee involvement in EmS is modelled by adding, to the organizational objective, an additional objective that represents employee empowerment. The

latter is measured by the summation of the score given to the satisfied plans. Therefore, the optimization problem of FWS is reformulated to consider the following objectives:

1. Maximizing the productivity rate. For the FWS problem, this is expressed in terms of the number of served jobs multiplied by their costs (Equation 4.1).

2. Maximizing plans satisfaction objective. This is defined by the number of satisfied work plans multiplied by their weights (Equation 5.4).

As discussed in section 5.2.3, the organizational objective and empowerment objective are conflicting objectives, at least at the optimization stage. This implies that the optimization process should target a set of solutions that represent the optimal set of trade-offs between the two objectives.

Treating empowerment scheduling as a multi-objective optimization problem, and hence producing the trade-off between the two objectives, is a key feature of the model. It enables organizations to retain their full control over the scheduling decision, as in *traditional* scheduling. In fact, it helps organizations assessing the immediate impact of empowerment on the overall schedule, and thus avoiding undesirable outcomes effectively. An example of such an outcome would be when an important task is not favoured by employees. This outcome can be resolved by increasing the task's priority. With this increase the impact of unallocating this task on the productivity rate grows, and thus the chance of allocating this task enhances, particularly in sub-fronts (on the trade-off curve) where the productivity rate has more importance than the plans satisfaction objective.

## 7.2 GPLS: A Scheduling Algorithm

This study investigates the application of GPLS, as a multi-objective optimization algorithm, to the empowerment scheduling model for FWS (EmS-FWS). The EmS-FWS problem is formulated in section 5.2.4. The motives of applying GPLS are: (1)

the outstanding performance of GPLS and its frameworks on biobjective optimization problems, and (2) the potential of GLS to be an efficient technique for this scheduling problem, since it holds state-of-the-art results on relevant scheduling problems. The following describes the implementation of the proposed GPLS and its frameworks for the EmS-FWS problem.

### 7.2.1 The Implementation of GPLS for the EmS-FWS problem

In section 5.4.2, the single-objective local search and GLS for the EmS-FWS problem are defined. A similar approach is followed to apply GPLS to the EmS-FWS problem, as follows.

#### 7.2.1.1 PLS

The basic components of PLS are defined similar to that of the single-objective local search as discussed in section 5.4.2. This includes the solution representation, which is an order of tasks; the deterministic scheduling procedure that transforms an order to a complete schedule; and the neighbourhood operator which is a single-swap between any two tasks. The only difference is in the cost function. Instead of aggregating the two objectives into a single cost function, they are considered separately. A solution is evaluated using the notion of Pareto optimality based on the two objective functions: productivity rate (Equation 4.1) and plans satisfaction (Equation 5.4), both of which are to be maximized.

#### 7.2.1.2 Features and their Costs

In the EmS-FWS problem, there are two objectives of different natures, and thus two sets of features need to be defined. As maximizing the total allocated tasks is an objective, each feature in the first set represents the failure in serving a job. The second objective concerns maximizing the total satisfied employees' plans, and therefore each

feature in the second feature set represents the failure of satisfying a work plan. The task priority is considered as the cost of the first feature set (task allocation), whereas the weight associated with a working plan (i.e. $EP$) is considered the cost of the second feature set (plan satisfaction). At the penalization phase of GPLS, a feature from each set with the maximum utility is picked for penalization.

#### 7.2.1.3 GPLS-based Frameworks

In order to apply iGPLS, mGPLS and miGPLS to the EmS-FWS problem, the only component that needs to be defined is the procedure of generating a diverse initial solution set ($InitialSetGenerator$). In section 5.4.2, GLS was proposed to solve a scalar objective function which combines both objectives. The local search only (without penalization) is used here again as the improvement heuristic for each scalar function in the $InitialSetGenerator()$. The reason for this is that running multiple GLS is computationally expensive, while the computational time is limited in the present experiments.

An additional question that we intend to examine here is that since the PF is decomposed into sub-fronts using a representative weight setting for each sub-front, how efficient is to propagate this weight setting to the employed heuristics, if any? For example, in the EmS-FWS problem, a deterministic scheduling procedure is applied to transform each solution (i.e. an order of tasks) into a complete schedule. The basic idea is to assign each task to the first available technician in its associated list of technicians. The list is ordered heuristically using two criteria: the travelling time (as a heuristic to improve the productivity rate) and the technician's preference with accordance to his/her plan. The current implementation gives the second criteria more importance than the first one. Instead, the weight setting applied to each sub-front can be used to combine the two criteria into a single value, and tasks' associated lists are ordered accordingly. The feasibility of such a weight propagation mechanism is examined in

the conducted computational experiments here.

## 7.3 Experimental Design

### 7.3.1 Problem Instances

Since the focus here is on the underlying multi-objective optimization problem, we consider only a one-day scenario which includes a set of tasks, a set of technicians and their plans for that day. The problem generator developed in section 4.2 is used here to generate 50 problem instances using the default values of problem characteristics. For each problem instance, five different instances of work plans are generated as described in section 5.4.1. The only addition is that each plan is associated with a weight value which is uniformly sampled at random, instead of giving all plans the same cost. The tightest scenario is applied here where every technician has a plan in the considered day.

### 7.3.2 Performance Measures

Three performance indices are used in this experiment to measure the quality of an approximation in terms of convergence towards the PF and diversity on the objective space. They are (1) the C-metric (Definition 8), (2) the $R$ measure (Definition 10) that is to be maximized, and (3) the $H$ measure (Definition 11) that is to be maximized as well. The $R$ and $H$ measures are employed for the EmS-FWS problem since the *ideal* (Definition 6) and *nadir* (Definition 7) points in the objective space can be easily defined. The *ideal* point is set to (1,1) (i.e. all tasks are allocated and all plans are satisfied), whereas the *nadir* point is set to (0,0) (i.e. none of the tasks are allocated and none of the plans are satisfied).

### 7.3.3 Benchmarks and Experimental Settings

In order to evaluate the performance of GPLS and its frameworks on the EmS-FWS problem, comparisons are made with four algorithms that act as points of reference. These methods are the following:

1. The standard PLS (SteepPLS) as proposed in [95], in which a solution is picked randomly from the archive, and then its neighbourhood is fully explored while updating the archive with new non-dominant solutions.

2. The underlying PLS used by GPLS (GreedyPLS) which applies a first improvement, partial neighbourhood exploration strategies.

3. A single-objective local search (SO-LS) using weighted-sum aggregation. In this algorithm, the PF is decomposed into $N$ scalar objective optimization sub-fronts, by generating $N$ uniformly distributed normalized weight vectors. Then, the local search method as defined in section 5.4.2 is applied to each scalar objective, and the obtained local optimum is added to the archive. The value of $N$ is set to 100, and the non-dominant solutions among all the 100 solutions generated in the 100 runs form the trade-off curve.

4. A random method (RAND) that randomly generates a certain number of solutions (similar to that of other algorithms) and add them to the maintained set of non-dominant solutions (i.e. the archive).

The running lengths allowed for those algorithms are expressed in terms of a maximum number of evaluations ($maxEval$) which is set to 200,000 evaluations. The size of the *archive* is unlimited. The only parameter to GPLS is $\lambda$ which is set empirically to 0.01. In GPLS-based frameworks, the time is divided equally between the two phases (i.e. the generation of the initial approximation and GPLS). Approximating the PF in the first phase of iGPLS and mGPLS is achieved by decomposing the PF into five

scalar objectives (i.e. $k = 4$), for each of which a local optimum solution is obtained using the defined single-objective local search method. While iGPLS starts a single run of GPLS after filling the *archive* with these local optima, mGPLS executes five independent runs of GPLS, each of which starts from a local optimum. For miGPLS, there will be five independent runs of iGPLS, each of which starts from a sub-set of four local optima (i.e. $k = 19$ and $|agents| = 5$). As mentioned earlier, there will be 50 problem instances, for each of which five different samples of technicians' plans are generated. For each combination, 5 executions are performed, and thus the reported results for each method will be the average of 1250 executions in total.

## 7.4 Experimental Results and Discussion

Table 7.1 presents the means of the C-metric values of the final approximations obtained by GPLS, compared to PLS variants, SO-LS and RAND. Table 7.2 gives the means and standard deviations of the values of $R$ and $H$ measures for these algorithms. It also gives the CPU time required by each algorithm. The experiments have been performed on a PC with 3.0 GHz, 3.0 GB Intel Core 2 Duo processor. The significance of the results are analysed statistically using the t-test with 95% confidence level.

Table 7.1: Means of the C-metric values of GPLS, PLS variants, RAND and SO-LS

| | | $B$ | | | | |
|---|---|---|---|---|---|---|
| | $C(A,B)$ | RAND | SO-LS | SteepPLS | GreedyPLS | GPLS |
| $A$ | RAND | - | 0 | 0 | 0 | 0 |
| | SO-LS | 0.93 | - | 0.1 | 0 | 0 |
| | SteepPLS | 0.91 | 0.7 | - | 0 | 0 |
| | GreedyPLS | 0.97 | 0.96 | 0.92 | - | 0.27 |
| | GPLS | **0.99** | **0.99** | **0.98** | **0.56** | - |

The results clearly reveal the superiority of GPLS over both PLS variants which in turn outperform SO-LS and RAND. In terms of all metrics, GPLS obtains better results

Table 7.2: Means (avg) and standard deviations (stdev) of the $R$ and $H$ values of GPLS, PLS variants, RAND and SO-LS, as well as the CPU time in seconds each algorithm requires

| Algorithm | $R$ | | $H$ | | Time (sec) | |
|---|---|---|---|---|---|---|
| | avg | stdev | avg | stdev | avg | stdev |
| RAND | 0.787 | 0.02 | 0.517 | 0.03 | 109 | 6 |
| SO-LS | 0.814 | 0.02 | 0.577 | 0.04 | 132 | 16 |
| SteepPLS | 0.814 | 0.02 | 0.577 | 0.04 | 133 | 11 |
| GreedyPLS | 0.831 | 0.02 | 0.615 | 0.04 | 58 | 19 |
| GPLS | **0.838** | 0.02 | **0.629** | 0.04 | 125 | 10 |

than the other methods. As shown in Table 7.1, the approximation of the PF obtained by SteepPLS and SO-LS are mostly (about 92%-99%) dominated by that of GreedyPLS and GPLS. In comparison with GreedyPLS, GPLS produces better approximations than that of GreedyPLS in terms of the C-metric. On average, the approximations obtained by GPLS dominates 56% of that obtained by GreedyPLS, and 27% vice versa. The outstanding performance of GPLS is also demonstrated by its ability to obtain better values of the $R$ and $H$ measures than that of the other algorithms, as given in Table 7.2. For instance, the $R$ value of GPLS is 0.838, while GreedyPLS, SteepPLS, SO-LS and RAND obtain 0.831, 0.814, 0.814 and 0.787, respectively. The difference between these algorithms can be visually detected from Figure 7.1. All these differences are statistically significant, except the difference between SteepPLS and SO-LS in their $R$ and $H$ values. In terms of computational time, GreedyPLS reaches a Pareto local optimum set quite quickly, requiring about half the time of other algorithms. GPLS, on the other hand, consumes slightly less computational time than that of SteepPLS and SO-LS.

The results of GPLS, then, are compared to that of GPLS-based frameworks (mGPLS, iGPLS and miGPLS). The means of the C-metric values are given in Table 7.3, where that of $R$ and $H$ measures, together with the required CPU time, are presented

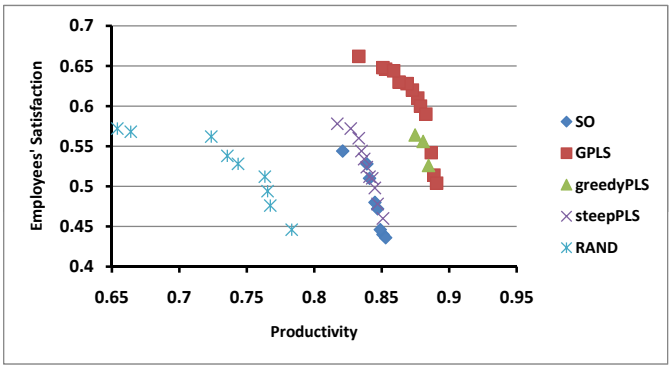



Figure 7.1: An example of the approximations obtained by RAND, SO-LS, PLS variant and GPLS

in Table 7.4.

The results suggest the performance of GPLS is (statistically) significantly enhanced by iGPLS and mGPLS, demonstrated by significant improvements in the values of both $R$ and $H$ measures without considerable demand for additional computational time (Table 7.4). The $H$ values of iGPLS and mGPLS are 0.848 and 0.847 respectively, compared to 0.838 for that of GPLS. In terms of the C-metric, Table 7.3 shows that GPLS is statistically better than mGPLS and iGPLS. A justification of this result is that GPLS invests the total available time to approximate the middle sub-fronts, which allows it to obtain a better convergence towards these sub-fronts at the expense of missing other sub-fronts, i.e. worse diversity in the objective space. To illustrate this, Figure 7.2 plots an example of the approximations obtained by GPLS and its frameworks.

The results also reveal that, statistically, miGPLS is inferior to GPLS in terms of all performance metrics. This suggests the contribution of the quality of the initial approximation (i.e. the first phase) to the overall performance of a GPLS-based frame-

Table 7.3: Means of the C-metric values of GPLS and its frameworks

| $C(A,B)$ | | $A$ | | | |
|---|---|---|---|---|---|
| | | GPLS | iGPLS | mGPLS | miGPLS |
| $B$ | GPLS | - | 0.45 | 0.54 | 0.72 |
| | iGPLS | 0.25 | - | 0.46 | 0.55 |
| | mGPLS | 0.19 | 0.33 | - | 0.53 |
| | miGPLS | 0.13 | 0.21 | 0.28 | - |

Table 7.4: Means (avg) and standard deviations (stdev) of the $R$ and $H$ values of GPLS and its frameworks, as well as the CPU time in seconds each algorithm requires

| Algorithm | $R$ | | $H$ | | Time (sec) | |
|---|---|---|---|---|---|---|
| | avg | stdev | avg | stdev | avg | stdev |
| GPLS | 0.838 | 0.02 | 0.629 | 0.04 | 125 | 10 |
| iGPLS | **0.848** | 0.02 | **0.649** | 0.04 | 127 | 10 |
| mGPLS | 0.847 | 0.02 | 0.646 | 0.04 | 129 | 10 |
| miGPLS | 0.835 | 0.02 | 0.623 | 0.04 | 128 | 10 |

work. In this experiment, a limited amount of time is given to iGPLS and mGPLS to approximate five sub-problems, while the same amount of time is given to miGPLS to approximate 20 sub-problems, which degrades the quality of the initial approximation for miGPLS. Thus, this indicates that increasing the size of the initial approximation at the expense of the convergence property is not always beneficial.

Summarizing, PLS variants (particularly GreedyPLS) demonstrate a very good performance on the EmS-FWS problem, which are significantly improved by GPLS. iGPLS and mGPLS confirm the potential of GPLS-based frameworks to enhance the performance of GPLS, specially in terms of diversity on the objective space. The performance of mGPLS encourages a next step to the parallel implementation of mGPLS whose independent GPLS runs are currently executed sequentially. Such a parallel implementation is expected to dynamically diminish the required computational times.

Finally, as suggested in section 7.2.1.3, the weight setting of each sub-front in

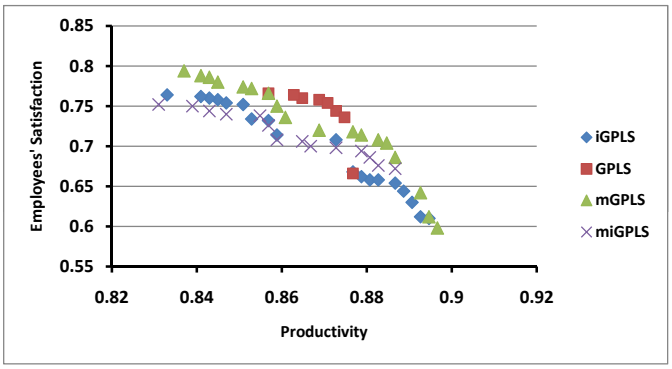



Figure 7.2: An example of the approximations obtained by GPLS and GPLS-based frameworks

GPLS-based frameworks can be propagated to the ordering heuristic applied to the list of technicians associated to each task. This ordering heuristic is employed by the scheduling procedure. In order to examine this propagation, the scheduling procedure in GPLS-based frameworks are adapted accordingly and the algorithms are executed on the same set of instances. The means of the $R$ and $H$ values obtained by the adapted GPLS-based frameworks are given in Table 7.5.

The results clearly reveal the statistical significant contribution of such a propagation to the performance of GPLS-based frameworks, demonstrated by the outstanding performance of these frameworks compared to their performance without propagation. Interestingly, this propagation mechanism dramatically increases the performance of miGPLS, which becomes very competitive to iGPLS and mGPLS. Taking the $R$ measure as an example, mGPLS, iGPLS and miGPLS obtain 0.908, 0.907 and 0.902 respectively, where these frameworks without propagation obtained 0.847, 0.848 and 0.835 respectively. The approximations obtained by the enhanced GPLS-based frameworks on a problem instance are plotted in Figure 7.3.

Table 7.5: Means (avg) and standard deviations (stdev) of the $R$ and $H$ values of GPLS base frameworks with weights propagation

| Algorithm | $R$ | | $H$ | |
|---|---|---|---|---|
| | avg | stdev | avg | stdev |
| iGPLS+ | 0.907 | 0.01 | 0.777 | 0.03 |
| mGPLS+ | **0.908** | 0.01 | **0.781** | 0.03 |
| miGPLS+ | 0.902 | 0.01 | 0.763 | 0.03 |

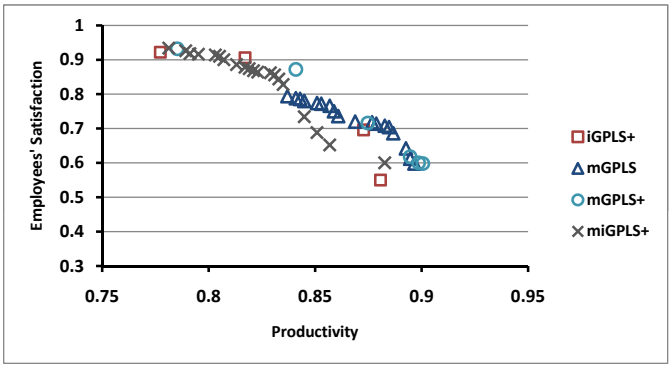



Figure 7.3: An example of the approximations obtained by the enhanced GPLS-based frameworks on the same problem instance used in Figure 7.2; for reference, mGPLS which obtains the best approximation in Figure 7.2, is plotted here

## 7.5 Conclusions

The proposed empowerment scheduling model (EmS) is dealt with as a multi-objective optimization problem. Instead of searching for a single optimum solution, the task is to find the best trade-offs between the empowerment objective and the organizational objective. A major benefit of this approach is the ability of the organization to analyse the cost of empowerment on the organizational interest, and thus any undesirable outcome can be easily detected and avoided since the organization still has a control

over the scheduling process.

As multi-objective optimization techniques, GPLS and its frameworks are applied to the EmS-FWS problem. Computational experiments confirm the effectiveness of GPLS as a stand-alone technique, whose performance is significantly enhanced by its frameworks. In addition, a new idea that is proposed and tested in this chapter is to propagate the weight setting to any employed heuristic that considers the different objectives while optimizing a sub-front (i.e. a scalar function) in GPLS-based frameworks. Interestingly, this approach dramatically improves the performance of GPLS-based frameworks.

On the other hand, unlike the MOKP and biTSP, this problem illustrates the implementation of GPLS on a problem with two objectives of different nature. GPLS defines a feature set for each objective. At the penalization phase of GPLS, a feature from each set with the maximum utility is penalized. This problem supports the potential of GPLS to be a general effective Pareto optimization algorithm.

# Part IV

# Concluding Remarks

# Chapter 8

# Conclusions

This chapter provides a summary of this thesis (section 8.1), identifies its contributions (section 8.2), and sheds the light on some rooms for further research (section 8.3).

## 8.1 Summary

The theme of the thesis can be summarized as developing an empowerment-based model for (field) workforce scheduling using multi-objective optimization as a modelling approach and Guided Local Search (GLS) as a solution technique. This implies that several topics need to be described and their relevant literature reviewed. In chapter 2, the Field Workforce Scheduling (FWS) problem is defined and its related literature is reviewed. It also discusses empowerment from a management perspective in order to understand the essence of empowerment as studied in the management literature. Finally, flexible management models proposed in the literature of workforce scheduling are reviewed. Chapter 3, on the other hand, is dedicated to review solution approaches. This includes describing and reviewing the literature of GLS and multi-objective optimization.

Before proposing a new model for the field workforce scheduling problem, the *tradi-*

*tional* model has to be defined and properly understood. Therefore, the first research step focuses on investigating the FWS problem, chapter 4. This includes formulating the problem and developing a problem instance generator accordingly. Since the FWS problem is a combination of assignment and routing problems, the performance of a representative local search algorithm from the literature of assignment problems and another one from that of routing problems are applied to the FWS problem. Several problem characteristics are identified for FWS, including problem size and constraint levels, in order to understand the problem properties and explain the performance difference between the two different solution approaches. A set of problem instances with various characteristics are generated to test the performance of the two local search methods. Computational results confirm the superiority of the assignment approach, which is positively correlated with the constraint level of the problem instance. On the other hand, the potential of the routing approach is positively correlated with the problem size.

Most flexible management models proposed for workforce scheduling claimed the implementation of empowerment. However, they vary significantly in their conceptions of empowerment in workforce scheduling. In chapter 5, the term *empowerment scheduling* is introduced, which formalizes an empowerment scheduling model as one that recognizes employees' individual interests that will be explicitly reflected in the final allocation decision. Simplicity, flexibility, fairness and providing a win-win approach are critical features to the effectiveness of an empowerment scheduling model. Based on this definition, a new empowerment scheduling model (EmS) is proposed and applied to the FWS problem. Extensive empirical experiments are conducted to examine and study various aspects of the model, including model's feasibility and fairness. The computational results suggest the effectiveness of EmS, providing a simple empowerment scheduling practice while addressing challenges such as maintaining fairness and transparency, and retaining the employer's control over the scheduling decision.

In chapter 6, an adaptation of GLS, which is a single-objective metaheuristic, to contain multiple objective scenarios is proposed. The new algorithm (GPLS) demonstrates the ability of GLS to sit on top of Pareto local search methods. The performance of GPLS is examined on two standard benchmarks for multi-objective optimization, namely the 0/1 multi-objective knapsack problem and the biobjective TSP. Several comparisons to representative multi-objective metaheuristics are made, including NSGA2, MOEA/D and 2PPLS. The experimental results reveal the effectiveness of GPLS to enhance the performance of the underlying Pareto local search, particularly in terms of the convergence towards the Pareto Front. The speedy convergence of GPLS motivates the incorporation of GPLS in a multi-phase framework to help GPLS approximate the whole PF. Several GPLS-based frameworks are proposed and their performances are evaluated. With these frameworks, the performance of GPLS is significantly improved, demonstrated by obtaining state-of-the-art results on the two standard benchmarks.

The outstanding performance of GPLS and its frameworks encourages their application to EmS which is transformed in chapter 5 to a single-objective optimization problem. In chapter 7, the implementations of the basic components of GPLS and its frameworks to EmS are detailed. Computational experiments confirm the effectiveness of GPLS and the outstanding performance of GPLS-based frameworks in comparison with other techniques, including Pareto local search. The performance of GPLS-based frameworks is significantly enhanced by propagating each applied scalar objective to any employed heuristics that consider both objectives, such as scheduling heuristics in FWS.

## 8.2 Contributions

The major contributions we have made in this thesis are as follows:

1. We have proposed an efficient implementation of empowerment for FWS, resulting in a new scheduling model (EmS). To our knowledge, the thesis presents the first conscientious study that examines the feasibility of empowerment in FWS and thoroughly evaluates the efficiency and effectiveness of the proposed model.

2. We have proposed Guided Pareto Local Search (GPLS) which is a new GLS-based multi-objective optimization algorithm. The key feature of GPLS is that it is a simple method with few parameters to tune. This feature enhances the potential of integrating GPLS into advanced multi-phase frameworks. Indeed, we have developed simple GPLS-based frameworks. With these frameworks, GPLS shows state-of-the-art performances on standard benchmarks for multi-objective optimization.

3. We have applied GPLS and its frameworks to the EmS-FWS problem (i.e. EmS applied to the FWS problem) which is formulated as a biobjective optimization problem. The outstanding performances of the proposed methods are empirically confirmed in comparisons with various representative techniques such as PLS.

These major contributions are supported by the following minor contributions:

1. We have carried out a computational study of the traditional (inflexible) model of FWS. This includes examining the performance difference between well-known local search algorithms developed for assignment problems and those for routing problems, on various instances of the FWS problem that share characteristics from both types of problems. This study is a foundational step towards achieving the major contributions of the thesis. It provides a formulated FWS problem, a generator for benchmark datasets, and useful insights on the performance of several solution techniques.

2. We have established a new formalization of empowerment in workforce scheduling. This results in a new term, Empowerment Scheduling, which defines the

constitution of empowerment and its requirements in workforce scheduling. This definition is strongly linked to aspects from the management literature which is the source of empowerment, making Empowerment Scheduling one of the most rigorous formalization of empowerment in the scheduling context.

## 8.3 Future Work

This thesis presents a new empowerment scheduling model which is applied to FWS. The model still requires future research to further enhance the model and adapt it to other workforce scheduling problems. A critical element of the proposed model is *employee power* ($EP$), which describes the amount of power an employee has over the scheduling decision. More precisely, it determines the penalty of violating an employee's plan. The scope of this element can be extended to reflect employees' individual commitment and contribution to the organizational interests. In this case, *employee power* provides an alternative rewarding scheme that can support others such as bonus programs. However, this extension should be done carefully because it could impact the overall transparency and fairness of the model. Another interesting research direction is to introduce new components to the model, with the aim of improving the model's efficiency. An example of such new elements is a *planner* that helps employees at the planning phase to construct compatible (and potentially highly probable) work plans. The planner will establish an interactive procedure with an employee by sharing related information (e.g. knowledge from the current problem instance and from other employees' work plans) and suggesting alternative plans. Advanced techniques such as preference elicitation [18] might be beneficial in this direction.

Whilst we have chosen to examine the model in relation to a particular workforce scheduling problem (i.e. the FWS problem), the model has the potential to be generalized for use with other workforce scheduling problems, particularly nurse rostering

where developing effective flexible scheduling models is still a major concern. Such problems include different scenarios than that of FWS, which require careful adaptation of the various aspects of the proposed model. In nurse scheduling, for example, employees may have multiple plans, and thus an employee can be *fully satisfied*, *partially satisfied*, or *unsatisfied*. In the current model, however, an employee can be either *satisfied* or *unsatisfied*, which is captured by an indicator $y$ with two possible values: 1 and 0 respectively.

On the other hand, this thesis offers a first study of GPLS. The method is still in its infancy and further research is required. Although the present study demonstrates the effectiveness of GPLS on several benchmarks with different structures and objectives, further studies are required to examine the sensitivity of GPLS to the complexity of the shape of the Pareto front which may be, unlike the considered benchmarks, non-convex or discontinuous. In addition, the study confirms the potential of GPLS to be a central element in a multi-phase framework, though the proposed GPLS-based frameworks are very simple. One potentially interesting research direction is to integrate GPLS into more sophisticated frameworks to enhance its performance further. There are several effective frameworks in the literature of multi-objective metaheuristics which can be coupled with GPLS. A recent framework that has attracted the attentions of researchers is MOEA/D [136]. Instead of performing multiple independent GPLS in mGPLS, a similar approach to that of MOEA/D can be applied such that each GPLS (as an agent) can employ some knowledge obtained from its neighbours (i.e. the closest GPLS agents).

# References


[1] Ahearne, M., Mathieu, J. & Rapp, A. (2005). To empower or not to empower your sales force? an empirical examination of the influence of leadership empowerment behaviour on customer satisfaction and performance. *Journal of Applied Psychology*, **90**, 945–955. 24, 25, 79

[2] Alfares, H. (2004). Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, **127**, 145–175. 15, 29

[3] Angel, E., Bampis, E. & Gourves, L. (2004). A dynasearch neighborhood for the bicriteria traveling salesman problem. *Metaheuristics for Multiobjective Optimisation*, **535**, 153–176. 6, 46, 51, 53

[4] Armentano, V. & Claudio, J. (2004). An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem. *Journal of Heuristics*, **10**, 463–481. 48

[5] Arroyo, J.E.C., Vieira, P.S. & Vianna, D.S. (2008). A grasp algorithm for the multi-criteria minimum spanning tree problem. *Annals of Operations Research*, **159**, 125–133. 48

[6] Azaiez, M.N. & Al Sharif, S.S. (2005). A 0-1 goal programming model for nurse scheduling. *Comput. Oper. Res.*, **32**, 491–507. 3, 29

[7] Azarmi, N. & Abdulhameed, W. (1995). Workforce scheduling with constraint logic programming. *BT Technology Journal*, **13**, 81–94. 17, 18, 20, 41

[8] Bailyn, L., Collins, R. & Song, Y. (2007). Self-scheduling for hospital nurses: an attempt and its difficulties. *Journal of Nursing Management*, **15**, 72–77. 27, 28

[9] Baker, S. (1993). Applying simulated annealing to the workforce management problem. Technical report, British Telecom Laboratories. 18

[10] Bandyopadhyay, S., Saha, S., Maulik, U. & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Evolutionary Computation*, **12**, 269–283. 48, 53

[11] Bard, J.F. & Purnomo, H.W. (2005). Preference scheduling for nurses using column generation. *European Journal of Operational Research*, **164**, 510–534. 3, 29, 30, 31, 83, 84

[12] Basseur, M. (2006). Design of cooperative algorithms for multi-objective optimization: application to the flow-shop scheduling problem. *4OR: A Quarterly Journal of Operations Research*, **4**, 255–258. 53, 139

[13] Baykasoglu, A., Owen, S. & Gindy, N. (1999). A taboo search based approach to find the pareto optimal set in multiple objective optimization. *Engineering Optimization*, **31**, 731–748. 47

[14] Ben Abdelaziz, F., Chaouachi, J. & Krichen, S. (1999). A hybrid heuristic for multiobjective knapsack problems. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, 205–212. 47

[15] Bertels, S. & Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Comput. Oper. Res.*, **33**, 2866–2890. 19, 20

[16] Borenstein, Y., Shah, N., Tsang, E.P.K., Dorne, R., Alsheddy, A. & Voudouris, C. (2010). On the partitioning of dynamic workforce scheduling problems. *Journal of Scheduling*, **13**, 411–425. 18, 58, 61

[17] Borges, P. (2000). Chesschanging horizon efficient set search: a simple principle for multiobjective optimization. *Journal of Heuristics*, **6**, 405–418. 46

[18] Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H. & Poole, D. (2004). Preference-based constrained optimization with cp-nets. *Computational Intelligence*, **20**, 137–157. 184

[19] Bradley, D. & Martin, J. (1991). Continuous personnel scheduling algorithms: a literature review. *Journal of the Society for Health Systems*, **2**, 8–23. 14

[20] Bräysy, I. & Gendreau, M. (2005). Vehicle routing problem with time windows, part 1: Route construction and local search algorithms. *Transportation Science*, **39**, 104–118. 64, 66

[21] Bräysy, O. & Gendreau, M. (2005). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, **39**, 119–139. 64, 66

[22] Burke, E.K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G. (2001). Fitness evaluation for nurse scheduling problems. In *The Congress on Evolutionary Computation (CEC2001)*, 1139–1146, IEEE Press, Seoul, Korea. 29

[23] Burke, E.K., De Causmaecker, P., Vanden Berghe, G. & Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, **7**, 441–499. 15, 29

[24] Chang, G.J. & Ho, P.H. (1998). The beta-assignment problems. *European Journal of Operational Research*, **104**, 593–600. 21

[25] Cheeseman, P., Kanefsky, B. & Taylor, W. (1991). Where the really hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91*, Sidney, Australia. 68

[26] Chiu, D., Cheung, S. & Leung, H. (2005). A multi-agent infrastructure for mobile workforce management in a service oriented enterprise. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, 85–94, IEEE Computer Society. 18

[27] Claydon, T. & Doyle, M. (1996). Trusting me, trusting you? the ethics of employee empowerment. *Personnel Review*, **25**, 13–25. 3, 25

[28] Coello, C.A.C. (2006). Evolutionary multi-objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, **1**, 28–36. 49

[29] Conger, J.A. & Kanungo, R.N. (1988). The empowerment process: Integrating theory and practice. *The Academy of Management Review*, **13**, 471–482. 3, 24, 79, 80

[30] Cordeau, J., Laporte, G., Savelsbergh, M. & Vigo, D. (2007). Vehicle routing. *Transportation*, **14**, 367–428. 22

[31] Cordeau, J., Laporte, G., Pasin, F. & Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, **13**, 393–409. 18

[32] Corne, D., Deb, K., Fleming, P. & Knowles, J. (2003). The good of the many outweighs the good of the one: evolutionary multi-objective optimization. *IEEE Connections Newsletter*, **1**, 9–13. 44

[33] Cowling, P., Colledge, N., Dahal, K. & Remde, S. (2006). The trade off between diversity and quality for multi-objective workforce scheduling. In J. Gottlieb & G. Raidl, eds., *Evolutionary Computation in Combinatorial Optimization*, vol. 3906 of *Lecture Notes in Computer Science*, 13–24, Springer Berlin / Heidelberg. 18, 20, 59

[34] Czyzzak, P. & Jaszkiewicz, A. (1998). Pareto simulated annealinga metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, **7**, 34–47. 48

[35] De Grano, M.L., Medeiros, D. & Eitel, D. (2009). Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Management Science*, **12**, 228–242. 3, 29, 30, 31, 83

[36] Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, **6**, 182–197. 45, 49

[37] Dimitriades, Z.S. (2001). Empowerment in total quality: Designing and implementing effective employee decision-making strategies. *Quality Management Journal*, **8**, 19–28. 24, 25, 79, 80

[38] Dondo, R. & Cerda, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, **176**, 1478–1507. 22

[39] Dutot, P., Laugier, A. & Bustos, A. (2006). Technicians and Interventions Scheduling for Telecommunications. Technical report, France Telecom R&D. 18

[40] Ehrgott, M. & Gandibleux, X. (2008). Hybrid metaheuristics for multiobjective combinatorial optimization. *Hybrid metaheuristics: an emerging approach to optimization*, **114**, 221–259. 44, 47

[41] Ernst, A.T., Jiang, H., Krishnamoorthy, M. & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, **153**, 3–27. 13, 14, 15

[42] Eveborn, P. & Rönnqvist, M. (2004). Scheduler - a system for staff planning. *Annals of Operations Research*, **128**, 21–45. 30, 31, 83

[43] Eveborn, P., Flisberg, P. & Rönnqvist, M. (2006). Laps Care–an operational system for staff planning of home care. *European Journal of Operational Research*, **171**, 962–976. 20

[44] Feillet, D., Dejax, P. & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, **39**, 188–205. 22

[45] Gandibleux, X., Mezdaoui, N. & Frville, A. (1997). A tabu search procedure to solve multiobjective combinatorial optimization problem. *Lecture Notes in Economics and Mathematical Systems*, **455**, 291–300. 47

[46] Gans, N., Koole, G. & Mandelbaum, A. (2003). Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, **5**, 79–141. 14, 15

[47] Geiger, M.J. (2007). The interactive pareto iterated local search (ipils) metaheuristic and its application to the biobjective portfolio optimization problem. *IEEE 2007 Symposium on Computational Intelligence in Multi-Criteria Decision Making*, **402**, 193–199. 48

[48] Gendreau, M. & Potvin, J. (2010). *Handbook of Metaheuristics*. Springer. 5, 36, 44

[49] Glover, F. & Laguna, M. (1998). *Tabu search*. Kluwer Academic Pub. 36

[50] Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional. 36

[51] Greasley, K., Bryman, A., Dainty, A., Price, A., Soetanto, R. & King, N. (2005). Employee perceptions of empowerment. *Employee Relations*, **27**, 354–368. 3, 23, 24, 25, 26, 79

[52] Gutin, G. & Punnen, A. (2002). *The traveling salesman problem and its variations*. Kluwer Academic Pub. 38

[53] Hales, D. (2005). How far dare you go? *Stakeholder*, **2**, 31–34. 24, 25

[54] Hani, Y., Amodeo, L., Yalaoui, F. & Chen, H. (2007). Ant colony optimization for solving an industrial layout problem. *European Journal of Operational Research*, **183**, 633–642. 41

[55] Hansen, M. (1997). Tabu search in multiobjective optimisation: Mots. In T.S. Honert & R. van den, eds., *13th International Conference on Multiple Criteria Decision Making*, 574–586, Springer-Verlag, Berlin. 47

[56] Hansen, M. & Jaszkiewicz, A. (1998). Evaluating the quality of approximations of the nondominated set. Tech. rep., Tech.Univ. of Denmark. 140

[57] Hifi, M., Michrafy, M. & Sbihi, A. (2004). Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society*, **55**, 1323–1332. 41, 125

[58] Hirschheim, R. & Miller, J. (1993). Implementing empowerment through teams: the case of texaco's information technology division. In *SIGCPR '93:*

*Proceedings of the 1993 conference on Computer personnel research*, 255–264, ACM, New York, NY, USA. 25, 80

[59] Hughes, E.J. (2005). Evolutionary many-objective optimisation: Many once or one many? **1**, 222–227. 44

[60] Hung, R. (2002). A note on nurse self-scheduling. *Nursing Economic*, **20**, 37–39. 27

[61] Ishibuchi, H. & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, **28**, 392–403. 49

[62] Jaeggi, D., Parks, G., Kipouros, T. & Clarkson, P. (2008). The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, **185**, 1192–1212. 48

[63] Jain, A. & Dubes, R. (1988). *Algorithms for clustering data*. Prentice Hall Advanced Reference Series. 53

[64] Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, **137**, 50–71. 43, 49

[65] Jaszkiewicz, A. (2002). On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, **6**, 402–412. 126, 131

[66] Jaszkiewicz, A. & Zielniewicz, P. (2009). Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. *European Journal of Operational Research*, **193**, 885–890. xii, 53, 125, 139, 149, 150, 153

[67] Jones, D., Mirrazavi, S. & Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, **137**, 1–9. 44, 131

[68] Jozefowiez, N., Semet, F. & Talbi, E. (2007). Target aiming pareto search and its application to the vehicle routing problem with route balancing. *Journal of Heuristics*, **13**, 455–469. 47

[69] Kilby, P., Prosser, P. & Shaw, P. (1999). Guided local search for the vehicle routing problem. In *Proceedings of the 2nd International Conference on Metaheuristics*, 473–486. 40

[70] Kirkpatrick, S. & Sorkin, G. (1998). Simulated annealing. In *The handbook of brain theory and neural networks*, 876–879, MIT Press. 36

[71] Knowles, J.D. & Corne, D.W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, **8**, 149–172. 45, 51

[72] Kuhn, H. (2008). The hungarian method for the assignment problem. *50 Years of Integer Programming 1958-2008*. 20

[73] Kumar, R. & Singh, P. (2007). Pareto evolutionary algorithm hybridized with local search for biobjective tsp. In A. Abraham, C. Grosan & H. Ishibuchi, eds., *Hybrid Evolutionary Algorithms*, vol. 75 of *Studies in Computational Intelligence*, 361–398, Springer Berlin / Heidelberg. 53, 139

[74] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, **43**, 408–416. 22

[75] Lau, T. & Tsang, E.P. (1998). The guided genetic algorithm and its application to the generalized assignment problem. In *Tenth IEEE International Conference on Tools with Artificial Intelligence*, 336–343, IEEE. 6, 37, 40

[76] Lesaint, D., Voudouris, C. & Azarmi, N. (2000). Dynamic workforce scheduling for british telecommunications plc. *Interfaces*, **30**, 45–56. 15, 17

[77] Lesaint, D., Voudouris, C., Azarmi, N., Alletson, I. & Laithwaite, B. (2003). Field workforce scheduling. *BT Technology Journal*, **21**, 23–26. 18, 61

[78] Lever, J., Wallace, M. & Richards, B. (1995). Constraint logic programming for scheduling and planning. *British Telecom Technology Journal*, **13**, 73–80. 18

[79] Li, H. & Landa-Silva, D. (2008). Evolutionary multi-objective simulated annealing with adaptive and competitive search direction. In *IEEE Congress on Evolutionary Computation (CEC2008)*, 3311–3318, IEEE. 48

[80] Li, H. & Landa-Silva, D. (2009). An elitist grasp metaheuristic for the multi-objective quadratic assignment problem. *Evolutionary Multi-Criterion Optimization*, **5467**, 481–494. 48

[81] Liefooghe, A., Mesmoudi, S., Humeau, J., Jourdan, L. & Talbi, E.G. (2009). A study on dominance-based local search approaches for multiobjective combinatorial optimization. In *International Workshop on Engineering Stochastic Local Search Algorithms: Designing, Implementing and Analyzing Effective Heuristics, SLS 2009, Brussels, Belgium*, vol. 5752, 120–124. 52

[82] Love, R. & Hoey, J. (1990). Management science improves fast-food operations. *Interfaces*, **20(2)**, 21–29. 29

[83] Lust, T. & Teghem, J. (2010). Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, **16**, 475–510. xii, 53, 116, 125, 139, 149, 150, 153

[84] Mester, D. & Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, **32**, 1593–1614. 6

[85] Mester, D. & Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, **34**, 2964–2975. 40

[86] Miettinen, K. (2008). Introduction to multiobjective optimization: Noninteractive approaches. *Multiobjective Optimization: Interactive and Evolutionary Approaches*, **5252**, 1–26. 45

[87] Miller, M.L. (1984). Implementing self-scheduling. *Journal of Nursing Administration*, **14**, 33–36. 27

[88] Mills, P. & Tsang, E.P.K. (2000). Guided local search for solving SAT and weighted MAX-SAT problems. *Journal of Automated Reasoning*, **24**, 205–223. 37

[89] Mills, P., Tsang, E.P.K. & Ford, J. (2003). Applying an extended guided local search to the quadratic assignment problem. *Annals of Operations Research*, **118**, 121–135. 37, 40, 41

[90] Muller, C., Magill, E. & Smith, D. (1993). Distributed genetic algorithms for resource allocation. Technical report, Strathclyde University. 18

[91] Naveh, Y., Richter, Y., Altshuler, Y., Gresh, D.L. & Connors, D.P. (2007). Workforce optimization: identification and assignment of professional workers using constraint programming. *IBM J. Res. Dev.*, **51**, 263–279. 18, 20

[92] Osman, I. (1995). Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Or Spectrum*, **17**, 211–225. 64

[93] Paquete, L. & Stützle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. *Evolutionary Multi-Criterion Optimization, Proceedings*, **2632**, 479–493. 46

[94] Paquete, L. & Stützle, T. (2007). Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In T. Gonzalez, ed., *Handbook of Approximation Algorithms and Metaheuristics*, 1–29, Computer and information science series. Boca Raton, FL, USA: Chapman & Hall/CRC. 44, 47, 51

[95] Paquete, L., Chiarandini, M. & Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. *Metaheuristics for Multiobjective Optimisation*, **535**, 177–200. 6, 42, 46, 51, 53, 116, 117, 125, 135, 137, 171

[96] Pentico, D. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, **176**, 774–793. 20, 21, 64

[97] Randhawa, S.U. & Sitompul, D. (1993). A heuristic-based computerized nurse scheduling system. *Comput. Oper. Res.*, **20**, 837–844. 30

[98] Sahin, R. & Turkbey, O. (2009). A simulated annealing algorithm to find approximate pareto optimal solutions for the multi-objective facility layout problem. *International Journal of Advanced Manufacturing Technology*, **41**, 1003–1018. 48

[99] Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, **13**, 87–127. 21

[100] Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st international Conference on Genetic Algorithms*, 93–100, L. Erlbaum Associates Inc. 44, 49

[101] Serafini, P. (1992). Simulated annealing for multiple objective optimization problems. In *Proceedings of the tenth international conference on multiple criteria decision making*, vol. 1, 87–96. 44, 48

[102] Shah, N., Tsang, E.P.K., Borenstein, Y., Dorne, R., Liret, A. & Voudouris, C. (2009). Intelligent Agent Based Workforce Empowerment. *Agent and Multi-Agent Systems: Technologies and Applications*, 163–172. 3, 32, 33

[103] Silver, S., Randolph, W.A. & Seibert, S. (2006). Implementing and sustaining empowerment: Lessons learned from comparison of a for-profit and a nonprofit organization. *Journal of Management Inquiry*, **15**, 47–58. 25, 32

[104] Smith, K., Everson, R. & Fieldsend, J. (2004). Dominance measures for multi-objective simulated annealing. In *The Congress on Evolutionary Computation (CEC2004)*, vol. 1, 23–30, IEEE. 48

[105] Suman, B. (2002). Multiobjective simulated annealing-a metaheuristic technique for multiobjective optimization of a constrained problem. *Foundations of Computing and Decision Sciences*, **27**, 171–191. 48

[106] Suman, B. & Kumar, P. (2005). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, **57**, 1143–1160. 48

[107] Tang, H., Miller-Hooks, E. & Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E: Logistics and Transportation Review*, **43**, 591–609. 19, 20

[108] Tarantilis, C.D., Zachariadis, E.E. & Kiranoudis, C.T. (2008). A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal On Computing*, **20**, 154–168. 40

[109] Teahan, B. (1998). Implementation of a self-scheduling system: a solution to more than just schedules! [corrected] [published erratum appears in j nurs manage 1999 jan; 7(1): 65]. *Journal of Nursing Management*, **6**, 361–368. 27, 28

[110] Thangiah, S., Osman, I. & Sun, T. (1994). Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. *Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27*. 66

[111] Tien, J. & Kamiyama, A. (1982). On manpower scheduling algorithms. *Siam Review*, **24**, 275–287. 14

[112] Toth, P. & Vigo, D., eds. (2001). *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. 21, 22

[113] Tsang, E.P.K. (1993). *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego. 85, 90

[114] Tsang, E.P.K. & Voudouris, C. (1997). Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters, Elsevier Science Publishers, Amsterdam*, **20**, 119–127. 18, 29, 30, 37, 41, 59, 64, 65, 95

[115] Tsang, E.P.K., Gosling, T., Virginas, B., Voudouris, C. & Owusu, G. (2005). Retractable contract network for distributed scheduling. In *2nd Multidisciplinary International Conference on Scheduling: Theory & Applications (MISTA)*, 485–500, New York. 3, 29

[116] Tsang, E.P.K., Gosling, T., Virginas, B., Voudouris, C., Owusu, G. & Liu, W. (2008). Retractable contract network for empowerment in workforce scheduling. *Multiagent and Grid Systems*, **4**, 25–44. 3, 32

[117] Tsang, E.P.K., Virginas, B., Gosling, T. & Liu, W. (2008). Multi-agent systems for staff empowerment. In C. Voudouris, D. Lesaint & G. Owusu, eds., *Service Chain Management*, 263–272, Springer Berlin Heidelberg. 32

[118] Ulungu, E. & Teghem, J. (1994). Multi-objective combinatorial optimization problems: A survey. *Journal of Multi Criteria Decision Analysis*, **3**, 83–104. 44

[119] Ulungu, E., Teghem, J., Fortemps, P. & Tuyttens, D. (1999). Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, **8**, 221–236. 48

[120] Ursu, M., Virginas, B., Owusu, G. & Voudouris, C. (2005). Distributed resource allocation via local choices: A case study of workforce allocation. *Int. J. Know.-Based Intell. Eng. Syst.*, **9**, 293–301. 58

[121] Van Veldhuizen, D.A. & Lamont, G.B. (1998). Evolutionary computationand convergence to a pareto front. In J.R. Koza, ed., *Late Break- ing Papers at the Genetic Programming 1998 Conference*, 221–228, Stanford University Bookstore, Stanford University, California. 43

[122] Vanhoucke, M. & Maenhout, B. (2009). On the characterization and generation of nurse scheduling problem instances. *European Journal of Operational Research*, **196**, 457–467. 76

[123] Vansteenwegen, P., Souffriau, W., Berghe, G.V. & Van Oudheusden, D. (2009). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, **196**, 118–127. 37

[124] Voudouris, C. (1997). Guided local search for combinatorial optimisation problems. Phd thesis, University of Essex. 5, 36

[125] Voudouris, C. & Tsang, E.P.K. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, **113**, 469–499. 37, 40, 127, 149

[126] Voudouris, C., Owusu, G., Dorne, R. & Lesaint, D. (2008). *Service Chain Management: Technology Innovation for the Service Business*. Springer Berlin Heidelberg. 2, 5

[127] Voudouris, C., Tsang, E.P.K. & Alsheddy, A. (2010). *Effective Application of Guided Local Search*. Wiley Encyclopedia of Operations Research and Management Science, John Wiley & Sons, Inc. 36

[128] Voudouris, C., Tsang, E.P.K. & Alsheddy, A. (2010). Guided local search. In M. Gendreau & J. Potvin, eds., *Handbook of metaheuristics*, 321–362, Springer. 36, 37, 128

[129] Voudouris, C., Tsang, E.P.K. & Alsheddy, A. (2010). *Guided Local Search*. Wiley Encyclopedia of Operations Research and Management Science, John Wiley & Sons, Inc. 37

[130] Wilkinson, A. (1998). Empowerment: theory and practice. *Personnel Review*, **27**, 40–56. 4, 24, 25

[131] Xu, J. & Chiu, S. (2001). Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, **7**, 495–509. 18

[132] Yagiura, M., Ibaraki, T. & Glover, F. (2004). An ejection chain approach for the generalized assignment problem. *INFORMS Journal On Computing*, **16**, 133–151. 21, 64

[133] Yang, R. (1996). Solving a workforce management problem with constraint programming. In *The 2nd International Conference on the Practical Application of Constraint Technology*, 373–387. 18

[134] Yura, K. (1994). Production scheduling to satisfy workers preferences for days off and overtime under due-date constraints. *International Journal of Production Economics*, **33**, 265–270. 29

[135] Zhang, Q., Sun, J., Tsang, E.P.K. & Ford, J. (2003). Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, in GECCO 2003*, 42–48. 41

[136] Zhang, Q.F. & Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, **11**, 712–731. xii, 49, 143, 144, 148, 185

[137] Zhong, Y. & Cole, M.H. (2005). A vehicle routing problem with backhauls and time windows: a guided local search solution. *Transportation Research Part E: Logistics and Transportation Review*, **41**, 131–144. 40

[138] Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257–271. 43, 49, 125, 126, 129

[139] Zitzler, E., Deb, K. & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, **8**, 173–195. 42

[140] Zitzler, E., Laumanns, M. & Thiele, L. (2001). Spea2: improving the strength pareto evolutionary algorithm. swiss federal institute of technology, lausanne. Tech. rep., Switzerland, Technical Report TIK-Rep 103, 2001. 45, 129