MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

OPTIMIZING THE VEHICLE ROUTES IN THE PRESENCE OF SHIFT MANAGEMENT

GÖZDE ALP

Ph.D. THESIS

Computer Engineering Department

Thesis Supervisor

Prof. Dr. Ali Fuat ALKAYA

ISTANBUL, 2021

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

OPTIMIZING THE VEHICLE ROUTES IN THE PRESENCE OF SHIFT MANAGEMENT

GÖZDE ALP

(724113701)

Ph.D. THESIS

Computer Engineering Department

Thesis Supervisor

Prof. Dr. Ali Fuat ALKAYA

ISTANBUL, 2021

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor Prof. Dr. Ali Fuat Alkaya for his support, patience, and guidance throughout my graduate study and completion of this thesis. He always encouraged me and he did his best like an older brother than an advisor.

Beside my advisor, I would also thank to my thesis committee members Assoc. Prof. Dr. Ali Haydar Özer and Assoc. Prof. Dr. Murat Şensoy for their valuable comments, kindness and help during the progress presentations.

I would like to thank all my colleagues from the Computer Science and Engineering Department at Marmara University for their sincere friendship and support throughout the years. My thanks go in particular to Asst. Prof. Dr. Sanem Arslan Yılmaz for her friendship and encouragement. I would also warmly thank to my dear colleague Merve Nur Aktan from Aydın Adnan Menderes University for listening all my concerns.

I would like to also thank to my parents Ayla Aslan and Reşat Aslan for raising me as a responsible individual and my brother Ali Uğur Aslan for his worthy support. Additionally,i would like to express special gratitude to my dear cousin Fatma Bilgin Güven for her spiritual support.

Lastly, it my pleasure to thank my beloved husband, Dr. Ahmet Burak Alp for his endless support and sacrifices from the first day to the last day of this long journey. At the end, a special thanks to my son Efe Alp for his existence and eternal love.

TABLE OF CONTENTS

ÖZET

VARDİYA YÖNETİMİ VARLIĞINDA ARAÇ ROTASI OPTİMİZASYONU

İş operasyonlarının bir çoğunu planlarken, vardiya planlama ve araç rotalama süreçlerini yönetmek kaçınılmazdır. Bu çalışmada, bu süreçler öncelikle ayrı ayrı daha sonra birlikte ele alınmıştır. Çalışmanın ilk kısmında; adalet tabanlı vardiya çizelgeleme probleminin matematiksel modeli sunulmuştur (FOSSP). Tek amaçlı optimizasyon problemleri için göçmen kuşlar optimizasyon algoritması (MBO) içerisinde hiper sezgisel (HH) komşuluk arama hareketlerine sahip yeni bir hibrit algoritma sunulmuştur (HHMBO). Deneyler, yeni hibrit algoritmanın, özellikle FOSSP'nin büyük boyutlu örnekleri için umut verici olduğunu göstermektedir. Çalışmanın ikinci kısmında; iş gücü çizelgeleme ve araç rotalama problemleri bir araya getirilerek vardiya ataması varlığında araç rotalama problemi (VRPSA) tanımlanmıştır. Problemin çok amaçlı matematiksel modeli, çözücü üzerinde doğrulanmıştır ve bir dizi evrimsel algoritma kullanılarak çözülmüştür. Çok amaçlı optimizasyon problemleri için dinamik komşu oluşturma (DNG) çerçevesi tanıtılmış ve VRPSA'nın çözümü için kullanılmıştır. Hesaplamalı deneylerin sonuçları, önerilen çerçevenin, hiper hacim (HV) ve ters nesil mesafe (IGD) göstergeleri açısından büyük boyutlu problem örneklerinde kesinlikle umut verici ve sağlam sonuçlar sunduğunu göstermektedir. Ayrıca, DNG çerçevesi iyi bilinen çok amaçlı optimizasyon algoritmalarına entegre edilerek başarısı analiz edilmiştir. Kıyaslama problemleriyle yapılan deneyler, algoritmaların DNG tabanlı versiyonlarının orijinal varyantlarından sırasıyla HV ve IGD göstergeleri açısından belirgin olarak daha iyi olduğunu göstermektedir.

Aralık, 2021 **Gözde ALP**

ABSTRACT

OPTIMIZING THE VEHICLE ROUTES IN THE PRESENCE OF SHIFT MANAGEMENT

Shift scheduling and vehicle routing processes are indispensable while planning most of the business operations. In this study, we firstly handled these processes separately and then we combined them. In the first part of the study; the mathematical formulation of a fairness oriented shift scheduling problem (FOSSP) is offered. A novel hybrid algorithm that has hyper heuristic (HH) neighborhood search behaviors embedded in migrating birds optimization (MBO) is introduced (HHMBO). HHMBO is applied on FOSSP and is compared with the reputed algorithms through extended computational experiments. Results prove that new hybrid computational intelligence technique is promising especially for large sized FOSSP instances. In the second part of the study; the workforce scheduling and vehicle routing problems are brought together and vehicle routing problem in the presence of shift assignment (VRPSA) is introduced to the literature. The multi-objective mathematical model of the problem is verified on a solver and also solved using a set of evolutionary algorithms. Dynamic neighbour generation framework for multi-objective optimization problems is introduced to the literature for solving VRPSA. Experimental results show that the proposed framework is definitely promising and robust in large sized problem instances in terms of hypervolume (HV) and inverted generational distance (IGD) metrics. Additional experiments are conducted on multi-objective benchmark problems to analyze the achievement of DNG framework. DNG is integrated to a set of multi-objective optimization algorithms. Experiments demonstrate that DNG based versions of algorithms are significantly better than their original variations on the average in terms of both IGD and HV metrics.

December, 2021 **Gözde ALP**

CLAIM FOR ORIGINALITY

OPTIMIZING THE VEHICLE ROUTES IN THE PRESENCE OF SHIFT MANAGEMENT

The significant contributions of this thesis can be summarized as follows:

A novel shift scheduling problem including the fairness perspective is defined. We call this complex optimization problem as fairness oriented integrated shift scheduling problem (FOSSP).

A hybrid computational intelligence algorithm is proposed to the optimization community. This novel hybrid algorithm has hyper heuristic (HH) neighborhood search movements embedded into a recently introduced migrating birds optimization algorithm (MBO). Therefore, it is called as HHMBO.

A novel multi-objective problem model is introduced to the literature that brings workforce scheduling and vehicle routing problems together in a way that has never been done before and consequently vehicle routing problem in the presence of shift assignment (VRPSA).

Lastly, dynamic neighbour generation (DNG) framework is proposed for solving multi-objective optimization problems. An extensive comparison research study is conducted and presented for analyzing the performance of DNG.

December, 2021 Prof. Dr. Ali Fuat ALKAYA Gözde ALP

SYMBOLS

$\boldsymbol{S}$: Set of schemas

s : Schema

$\boldsymbol{T}$ *:* Set of time slots of week

$\boldsymbol{T}$: Timeslot

E : Set of employees

e : Employee

W : Set of weeks

w : Week

$\boldsymbol{c_s}$: Cost of each schema

$\boldsymbol{x_s}$: Number of times schema *s* is used

$\boldsymbol{r_t}$ *:* The requirement at time slot *t*,

$\boldsymbol{a_{s,t}}$: Schema timeslot coverage matrix

$\boldsymbol{A_{e,w,s}}$: Assignment of employee *e* on week *w* and schema *s*

$\boldsymbol{M}$: Mean cost for all employees

$\boldsymbol{N_e}$: Mean cost for employee *e*

$\boldsymbol{D_e}$: Absolute difference for employee *e*

$\boldsymbol{a}$: Cooling rate

$\boldsymbol{R}$: The number of iterations at each temperature.

$\boldsymbol{b}$: The increment variable of *R*.

$\boldsymbol{Ri}$: Initial score

$\boldsymbol{Cq}$: Change quantity

$\boldsymbol{Ub}$: Upper boundary

$\boldsymbol{Lb}$: Lower boundary

$\boldsymbol{d_{ij}}$: The Euclidean distance of route from employee location i to employee location *j*.

$\boldsymbol{y_{es}}$: Assignment variable for employee e to shift s

$\boldsymbol{\gamma_i}$: Polynomial probability distribution

$\boldsymbol{n_m}$: Polynomial distribution index

$\boldsymbol{\eta}$: Crossover index

ABBREVIATIONS

AS :Adaptive Searching

AS_MC :Adaptive Searching Monte Carlo

AS_OI :Adaptive Searching Only Improvement

BKS :Best Known Solutions

CVRP :Capacitated Vehicle Routing Problem

DE :Differential Evolution

DM :Decision Maker

DNG :Dynamic Neighbour Generation

EA :Evolutionary Algorithm

FOSSP :Fairness Oriented Shift Scheduling Problem

GA :Genetic Algorithm

GAP :Generalized Assignment Problem

HH :Hyper Heuristics

HHMBO :Hyper Heuristic Migrating Birds Optimization

HV :Hypervolume

IGD :Inverted Generational Distance

MBO :Migrating Birds Optimization

MC :Monte Carlo

MHypEA :Multi-Objective Hyper-Heuristic Evolutionary Algorithm

MMBO :Multi-Objective Migrating Birds Optimization

MOP :Multi-Objective Optimization Problem

MOPSO :Multi Objective Particle Swarm Optimization

NSGA :Non Dominated Sorting Genetic Algorithm

NSGA-II :Fast and Elitist Dominated Sorting Genetic Algorithm

OI :Only Improvement

OVRP :Open Vehicle Routing Problem

POF :Pareto Optimal Front

PSA :Pareto Simulated Annealing

QAP :Quadratic Assignment Problems

RP :Random Permutation

RP_MC :Random Permutation Monte Carlo

RP_OI :Random Permutation Only Improvement

SA :Simulated Annealing

SBX **:**Simulated Binary Crossover

SOP **:**Single Objective Optimization Problem

SPEA2 **:**Strength Pareto Evolutionary Algorithm 2

SR **:**Simple Random

SR_MC **:**Simple Random Monte Carlo

SR_OI **:**Simple Random Only Improvement

TS **:**Tabu Search

VNG **:**Versatile Neighbour Generation

VRP **:**Vehicle Routing Problem

VRPSA **:**Vehicle Routing Problem in the Presence of Shift Assignment

WSRP **:**Workforce Scheduling and Routing Problem

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Number of large–scale organizations increases due to population growth in the world. Many problems that have been the subject of combinatorial optimization such as workforce scheduling, task scheduling and vehicle routing are especially encountered within the organizations those employ large number of employees including health institutions, call centers, and production factories.

Workforce scheduling is a general concept for referring to rostering problems such as shift scheduling, days off scheduling and task assignment. Shift scheduling applications involve taking orders or service requests, determining how many employees are needed to cover those orders, and assigning workers to duties during specific time intervals [**1**]. On the other hand, vehicle routing is an integer programming problem that aims to find out the optimal set of routes for a set of vehicles, in order to deliver a good or item to a given set of customers.

Scheduling and routing processes are indispensable while planning most of the business operations. These processes that can help to improve each other's efficiency are generally treated separately. Nevertheless, there are some problems presented in the literature those bring routing and scheduling logic together [**2**]-[**4**]. The common name for these problems is known as workforce scheduling and routing problem (WSRP). Nurse scheduling problem, technician scheduling problem, home health care routing and scheduling problem are samples of WSRP.

The models that bring routing and scheduling logic together are generally exposed as multi-objective optimization problems (MOP) [**5**]. Unlike the unique results in single objective optimization problems (SOP), a MOP exposes a set of non-dominated solutions. A non-dominated solution is not dominated by any other candidate in the solution space, also known as Pareto optimal solution. MOP contains three phases including model building, optimization, and decision making. Decision maker (DM) finds out the most beneficial solution among the non-dominated set. Optimization techniques applied on MOP may be divided into four categories according to the decision maker strategy. Those are no preference method, a priori methods, a posteriori methods and progressive methods [**6**].

A posteriori methods are widely applied and accepted on MOP, where a representative set of non-dominated solutions is first found and then the DM chooses the most useful one.

Evolutionary algorithms are popular approaches for generating Pareto optimal solutions on MOPs. Currently, most of the multi-objective evolutionary optimization algorithms apply Pareto-based ranking schemes. Researchers have adapted reputed evolutionary algorithms to MOPs in course of time.

The motivation of this thesis has emerged from the employee transportation and shift arrangement requirements of a real company. Collaborative is a large-scale company located far from residential areas and provides transportation of employees by service buses. Therefore, a significant budget is allocated for employee transportation. While employee shuttle is being planned, assigning the minimum number of employees to the appropriate shifts and vehicles, and building the shortest routes for each vehicle must be tackled simultaneously. In the current practice, when employees are assigned to the same shift, the locations where they will get into the shuttle are not considered. As a result, employees in two remote locations of the city may be assigned to the same shift. Besides, due to unwise planning, the capacities of service vehicles, which are transportation resources, are wasted. In some cases, high-capacity shuttles are used to transport only one person or two or three vehicles are operated when a single vehicle is sufficient to transport employees assigned to a single shift.

1.1 Scope and Objectives

Based on the scheduling and routing requirements of our collaborative, firstly we conducted separate studies about in both routing and scheduling problems. Working with the problems separately made us ready to bring them together. In the next step, we have conducted research on the problem of solving vehicle routing and shift scheduling processes together.

In the first stage, we have introduced a novel problem model to the literature which is the Fairness Oriented Shift Scheduling Problem (FOSSP). Problem description and the mathematical formulation is given in Section 3.1. The presented problem has a sophisticated solution space. Exploring the solution space by more than one neighborhood search methods could increase the exploration capability thereby contribute finding promising results. Hyper heuristics is a widely known way of applying multiple heuristic techniques. On the other hand, recently proposed migrating birds optimization algorithm is receiving growing attention from the researchers, due to the ability of providing good quality solutions over a wide range of instances of a problem. As a result we wanted to integrate the exploitation capability of MBO with HH's exploration capability for solving the problem. Then we have introduced a novel hybrid algorithm for SOPs to the literature

which has hyper heuristic movements embedded in the migrating birds optimization problem (HHMBO). Introduced hybrid technique is explained in Section 3.2.5.

In the second stage, we brought employee scheduling and vehicle routing problems together in a novel way. For this, we have introduced the multi-objective mathematical model of vehicle routing problem in the presence of shift assignment (VRPSA) to the literature. The properties and the limitations of VRPSA is described in Section 4.1. VRPSA has a compex solution space and a single neighbouring technique is insufficient to obtain promising results. More than one neighborhood search methods with different characteristics are required to obtain more efficient exploration in the solution space. Applying multiple heuristic operators is basically restricted with HH applications in the literature. However, multiple alteration operators may be utilized in all meta-heuristic algorithms. Multiple mutation applications on meta-heuristic techniques had been investigated before on single objective optimization problems and called as dynamic mutation [**7**]. Based on this idea, we have presented a novel Dynamic Neighbor Generation (DNG) framework for MOPs. The most significant feature of DNG is its easy applicability. Detailed information about DNG is given in Section 4.3.1. Finally, we have integrated DNG with the well-known heuristic based multi-objective solution techniques. An extensive comparative study on a set of multi-objective test problems is presented in Chapter 5.

1.2 Thesis Contributions

The main contributions of this thesis study can be summarized as follows:

- The necessary mathematical model for Fairness oriented integrated shift scheduling problem (FOSSP) model is proposed.
- A novel hybrid algorithm has hyper heuristic (HH) neighborhood search movements embedded into a recently introduced migrating birds optimization algorithm (MBO) is introduced (HHMBO).
- Multi objective mathematical model of vehicle routing problem in the presence of shift assignment (VRPSA) is introduced.
- Dynamic neighbor generation (DNG) framework for multi-objective optimization problems is offered.
- DNG framework is integrated with the state of the art techniques in multi-objective domain and a comprehensive research study presented.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows. An extensive literature investigation is presented related with the scope of thesis in Chapter 2. HHMBO is presented and the FOSSP model is presented in Chapter 3. A multi-objective VRPSA formulation is given in Chapter 4 along with the experimental studies conducted. Dynamic neighbour generation framework for multi-objective optimization problems is explained and comparative studies conducted are described in Chapter 5. Finally, we conclude the thesis document by providing summary and directions for future work in Chapter 6.

2. LITERATURE REVIEW

We present a summary of related work in three subsections. Workforce scheduling literature is summarized in the first subsection. Routing problems literature is reviewed in the second subsection. The multi-objective optimization problems and the multi-objective solution techniques are summarized in the third subsection.

2.1 Workforce Scheduling

Many scientific studies have been examined in order to locate the problem that we have dealt with. There are many review studies that extensively examine the workforce scheduling literature [**8**]-[**12**]. Brucker et. al. [**13**] categorized the shift scheduling problems. Baker classified workforce scheduling problems into three types [**14**]; shift scheduling problem, days-off scheduling problem and tour scheduling problem. Shift scheduling comprises choosing a set of the most suitable shifts from a pool of nominee shifts on a single day by satisfying employee requirement in each time slot. On the other hand, main concern in days off scheduling is to assign the off-work days for each employee over the rostering horizon, rather than to assign the employee particular shifts on working days. In contrast to days off scheduling, tour scheduling selects off days for the employees and determines shift types for each of their working days over the planning horizon. Mathematical model of tour scheduling problem is summarized and problem classification is presented by Alfares [**15**]. Erhard et. al. [**12**] presented the first review study that focuses on quantitative methods for physician scheduling literature. They indicated in their study that it is more common to include fairness aspects for physician scheduling domain than to ignore them.

Brunner and Stolletz [**16**] prepared a schedule for the employees in check in counters at airport. Atlason et. al. [**17**] presented a shift assignment study in which planning horizon is divided into short periods. Morris and Showalter [**18**] presented a rewarding study that clarifies the usage of set covering formulation for shift scheduling, days off scheduling and tour scheduling problems [**19**]-[**21**]. Alfares [**22**] proposed a specific type of days-off scheduling, solved by proposed solution technique. Altner et. al. [**23**] presented a days-off scheduling formulation. Lau [**24**] introduced a special tour scheduling problem.

Many meta-heuristic based solution techniques are applied for solving the workforce scheduling problems so far. They include artificial bee colony algorithm [**25**], particle

swarm optimization [**26**], migrating birds optimization [**27**], genetic algorithm (GA) [**28**], an adaptive multiple crossover GA [**29**], a modified differential evolution (DE) algorithm ([**30**]-[**33**]). A tabu-search hyper heuristics solution for nurse scheduling problem is presented in Burke et. al. [**34**]. Pan et. al. [**35**] proposed a hybrid heuristic, combining tabu search and large neighbourhood search techniques. Leandro et. al. [**36**] presented a lagrangian relaxation and a meta-heuristic for the multi-activity shift scheduling problem. A hybrid discrete water wave optimization algorithm is presented for scheduling problems [**37**].

Combinatorial optimization problems have a sophisticated solution space and a single improvement technique is insufficient to obtain promising results. More than one neighborhood search methods with different characteristics may be promising to obtain more efficient exploration in the solution space. One widely known way of applying multiple heuristic techniques is hyper heuristics. Hyper heuristics (HH) is a meta-heuristic technique formally defined as, a heuristic to select and manage a set of low level heuristics. Many HH variations for optimization and scheduling problems are defined up to now, such as tensor based hyper heuristic for nurse scheduling [**38**], column based hyper heuristic for bus driver scheduling problem [**39**], a tabu-search hyper heuristics [**40**], a simulated annealing hyper heuristic [**41**]. Hyper heuristics are also integrated with swarm based computational intelligence techniques such as; hybrid particle swarm optimization [**42**], ant colony optimization based hyper heuristic [**43**], hyper-heuristic based artificial bee colony algorithm [**44**]. On the other hand, Duman et. al. [**45**] recently proposed a new meta-heuristic algorithm named migrating birds optimization (MBO). They applied their algorithm to quadratic assignment problems (QAP) and proved its efficiency.

2.2 Routing Problems

A brief summary of remarkable or current studies on VRP [**46**] is given in this section. A survey of some of the recently employed solution methodologies in the field of VRP and its variants are presented in [**47**]. There are many VRP variations that had been identified over the time, due to diversity of problem constraints and requirements. One of them, VRP with time windows, is studied in [**48**], where each customer has a time limit to be visited. VRP with uncertain demands is another variation, stochastic in customer demand [**49**]. Open VRP is another open route VRP variant [**50**].

Particle swarm optimization algorithm was implemented for VRP in [**51**]. GA containing TS movements is applied to OVRP in [**52**]. A meta-heuristic approach is proposed for

VRP in [**53**]. The proposed technique actually works similar with SA in point of giving chance to worse results. A local search like heuristic improvement procedure, that utilizes integer linear programming improvements is proposed in [**54**]. New CVRP benchmark instances were proposed in 2017 in [**55**]. DE that has local search movements inside is applied on CVRP in [**56**] in 2015. GA based approach is proposed to solve two-echelon CVRP in 2017 [**57**]. A novel SA framework is improved to solve CVRP in 2018 [**58**]. A meta-heuristic algorithm based on a probabilistic granular TS applied on CVRP in [**59**] and a real data set taken from a collaborator company.

2.3 Multi Objective Optimization

MOPs have become a separate research field with their own techniques, test problems and performance measures. The problems in many industrial areas such as logistics, scheduling, engineering have been modeled as MOPs [**60**]-[**62**]. In addition, many synthetic multi-objective problems are presented in scientific studies [**63**], [**64**]. MOPs have two or more objective functions those may be compatible or in conflict with each other. In any case, optimal decision set need to be discovered in the presence of a balance or strategy between goals.

Numerous nature inspired Eas for MOPs have been introduced up to the present; such as non-dominated sorting genetic algorithm (NSGA) [**65**], fast and elitist non-dominated sorting genetic algorithm (NSGA-II) [**66**], strength Pareto evolutionary algorithm 2 (SPEA2) [**67**]-[**70**] multi objective particle swarm optimization (MOPSO) [**71**], bat algorithm for multi-objective optimization [**72**], multi-objective gray wolf optimization [**73**], multi-objective migrating birds optimization (MMBO) [**74**]. Local search algorithms and probabilistic techniques are also applied for solving MOPs including multi-objective cuckoo search [**75**], Pareto simulated annealing (PSA) [**76**].

WSRP is a well-studied MOP that brings vehicle routing and workforce scheduling together. It basically selects the most suitable employees and assigns them to carry out tasks at previously known geographical locations. An employee is selected for each task and a route is constructed for the vehicle that provides transportation of that employee along previously known locations. The main purpose of WSRP is minimizing the total operational cost including employee costs and transportation costs. Salazar et.al. [**77**] present a survey study about WSRP. Algethami et.al. [**78**] applied GA for solving WSRP. Bruecker et. al. [**79**] integrate developing shift schedules and waste collection routes. A custom mixed integer programming model for workforce scheduling and routing problem is presented in [**80**]. The model contains working region constraints besides various

constraint formulations from the literature. Ciancio et.al. [**2**] presents an integrated approach for local public bus companies; that configures the bus schedule according to the given set of rides to do, while assigning the most suitable driver to the vehicle. The first part of the problem corresponds to multi depot vehicle scheduling problem whereas the second one corresponds to crew scheduling problem. Fikar and Hirsch [**4**] present a valuable survey study about home healthy care routing and scheduling. Milburn and Spicer [**81**] introduce a variation of home health nurse routing and scheduling problem which represent the interests of the agency, patient, and nurse with its three objectives about travel cost, nurse consistency, and balanced workload. Schrotenboer et.al. [**3**] introduced technician allocation and routing problem that aims jointly optimize the daily allocation of technicians with different skills and the daily vessel routes transporting those technicians performing all maintenance activities.

3. HYPER HEURISTIC BASED MIGRATING BIRDS OPTIMIZATION FOR A FAIRNESS ORIENTED SHIFT SCHEDULING PROBLEM

Fairness oriented shift scheduling problem model and the hyper heuristic based migrating birds optimization algorithm is introduced in this chapter. Extensive computational experiments are also presented with statistical analysis.

The real world problem we tackled aims to provide fair personnel work schedules for a large-scale manufacturing company that works 24 hours and seven days a week. Personnel requirements for each day and shift differ and are updated periodically. Employee requirement changes in every four weeks, regularly. On the other hand, there is a high rate of employee circulation. Large numbers of new employees are recruited and large numbers of employees leave at the end of each period. Therefore, possible maximum level of fairness within each planning period is sought. There are also some essential legal regulations for employee schedules which make the problem much more complicated.

The presented problem has a sophisticated solution space. We have envisioned that exploring the solution space by more than one neighborhood search methods could contribute finding promising results. Hyper heuristics is a widely known way of applying multiple heuristic techniques. On the other hand recently proposed migrating birds optimization algorithm is receiving growing attention from the researchers, due to the ability of providing good quality solutions over a wide range of instances of a problem. As a result we wanted to integrate the exploitation capability of MBO with HH's exploration capability for solving the problem. Consequently, we introduce a novel technique that has hyper heuristic movements embedded in the migrating birds optimization algorithm (HHMBO). To the best of our knowledge, there has been no investigation about hybridizing hyper heuristics with MBO in the literature. HHMBO is compared with the well-known algorithms and it is shown that the proposed hybrid algorithm is promising especially for large sized instances of Fairness Oriented Shift Scheduling Problem (FOSSP).

HHMBO obtains very successful results on the FOSSP model. Hence, we believe HHMBO is also promising for the other combinatorial optimization problems. Therefore, we aimed to measure its performance on a widely tackled problem's benchmark instances.

Based on this idea, we have implemented HHMBO on the quadratic assignment problem (QAP). As a result, HHMBO is again found to be very successful on converging to the best known solutions for QAP instances with different densities.

The contribution of this chapter is three-fold: (i) defining and validating the mathematical model of the fairness oriented shift scheduling problem (FOSSP), (ii) proposing a new hybrid meta-heuristic algorithm that is composed of hyper heuristic movements embedded in migrating birds optimization (HHMBO) and (iii) showing the superiority of HHMBO through computational experiments conducted on both FOSSP data set and well-known QAP benchmark instances.

The remainder of chapter is organized as follows: Problem description details are presented in Section 3.1. Proposed solution techniques are described in detail in Section 3.2. Experimental setup is demonstrated in Section 3.3. Results are presented and investigated in Section 3.4. Section 3.5 concludes and comprises suggestions for further research.

3.1 Problem Description

In this section, we try to define and clarify the proposed and tackled problem. Firstly, the properties and limitations of the problem are explained. Then, cost measurement method is described. Lastly, the mathematical model of FOSSP is presented. The terms time period, time slot and working slot are used interchangeably throughout this chapter.

3.1.1 Properties and limitations of the problem

The main motivation behind defining this new problem is obtaining a fair distribution of working slots (time periods) among fixed number of employees during the specified planning period. Production factory operates incessantly 24 hours a day, seven days a week. There are totally three shifts in each day; those are day shift, evening shift and night shift. Employee requirement list contains weekly employee demand for each day of the week and each shift of a day. The planning horizon which is specified as four weeks in our case may consist of several weeks. Employee requirement list is updated by the management department at the end of each period, and the same requirement list is valid for all weeks in planning horizon. Multi-week employee schedules are prepared supplying personnel demand. That is, the output schedule must satisfy the number of workers needed for each day and shift, besides ensuring a fair distribution between employees along the given weeks.

Employees are assigned to a specific stationary shift type during one week and each employee has two xoff days. However, for each worker, shift types and off days may change from one week to another. Additionally, there are some essential legal regulations for employee schedules. Some of those rules may be listed as follows; if an employee has night shift on Sunday, (s)he cannot be assigned to day shift on the following Monday, if an employee has weekend off, he/she cannot be off on adjacent Monday and an employee cannot work longer than six consecutive days. These additional legal constraints increase the complexity of the problem. Under these constraints, the aim is to provide a fair schedule distribution among employees throughout the planning period.

3.1.2 Determination of schema costs

A tour or a schema may be defined by a seven lettered string containing D, E, N or X which mean day shift, evening shift, night shift and off day, respectively and each letter corresponds to the days of the week starting from Monday up to Sunday. An example schema string is "XXEEEEE" which means Monday and Tuesday are off and the week continues with five successive evening shifts. Thus, there occur exactly 63 different schemata with three shift types and two days off rules.

Three shifts in a day (namely day, evening and night) and seven days of the week make a total of 21 time slots, starting from Monday day shift and ending with Sunday night shift. In the factory, day shift is between 07:00 and 15:00, evening shift is between 15:00 and 23:00, and night shift is between 23:00 and 07:00. Each employee is assigned to one of those shifts each week; additionally each employee has two off days during a week.

Taking individual employee preferences into account while modeling the problem was not appropriate, as it would make the already complex problem more complex. Hereby, the preferability of time zones have been determined by a survey conducted with the employees. We had classified the off days, according to their desirability. We have identified five off day types; (i) weekend off, (ii) Friday-Saturday off, (iii) Sunday-weekday off, (iv) consecutive weekday off and (v) separate weekday off. For determining the quality of a schema, we assigned numerical values to off day types and shift types. Based on the survey results obtained, enumeration for shift types are determined as one, two and three for day, evening and night shifts, respectively. Similarly, off day types; weekend off, Friday-Saturday off, Sunday-weekday off, consecutive weekday off and separate weekday off are enumerated from one to five, respectively (Table 3.1).

Table 3.1. Shift type and off day type enumeration.

Type	Cost
Day shift	1
Evening shift	2
Night shift	3
Weekend off	1
Friday - Saturday off	2
Sunday weekday off	3
Consecutive weekday off	4
Separate weekday off	5

Each schema (out of 63 schemata) has a predetermined cost, obtained by the multiplication of its off day type enumeration and shift type enumeration. An illustrative cost calculation example for some of the schemata is given in Table 3.2. One can easily recognize that more preferable schemata have lower cost when the schema cost calculation is examined. So, the objective of the problem turns out to be minimizing the cost. As an example "DDDDDXX" is the most preferred and lowest cost tour.

Table 3.2. Some schema cost computation examples.

Schema	String	c(s)
21	DDDDDXX	1 (1*1)
42	EEEEEXX	2 (1*2)
63	NNNNNXX	3 (1*3)
8	DXDXDDD	5 (5*1)
29	EXEXEEE	10(5*2)
50	NXNXNNN	15(5*3)

A small sample requirement input for four employees is illustrated in Table 3.3. Schema ID is schema identifier; a schema may be identified by a number between one and 63. FOSSP scheduling decides schema usage quantities by considering the employee requirements for each time period. The requirements for Monday day shift and evening shift are zero and no employees are assigned to day shift on Monday. The requirement for Monday night shift is one and one employee will be assigned to night shift on Monday with schema ID 63. FOSSP output satisfies employee requirements by selecting the most suitable schema with the required usage quantity.

Since the requirements are same for all weeks during the planning period (four weeks in our case), the same usage quantity of each schema must be used for each week. The sample output of FOSSP is given in Table 3.4, schema ID one is assigned to employee one and employee three in week one, another two employees in week two and so on. Schema ID two is not assigned to anyone. Schema ID 25 is assigned to one employee and schema

ID 63 is assigned to one employee in each week. The problem assigns the same schemas with same quantities to different employees for given number of weeks.

Table 3.3. Sample requirement input for all 21 timeslots. Timeslots are given with their definitions.

Time slot ID	Definition	Requirement
1	Monday day shift	0
2	Monday evening shift	0
3	Monday night shift	1
4	Tuesday day shift	1
5	Tuesday evening shift	1
6	Tuesday night shift	1
7	Wednesday day shift	1
8	Wednesday evening shift	1
9	Wednesday night shift	1
10	Thursday day shift	2
11	Thursday evening shift	1
12	Thursday night shift	1
13	Friday day shift	2
14	Friday evening shift	0
15	Friday night shift	1
16	Saturday day shift	2
17	Saturday evening shift	1
18	Saturday night shift	0
19	Sunday day shift	2
20	Sunday evening shift	1
21	Sunday night shift	0

Table 3.4. Sample schema assignment for four employees and four weeks.

Employee Id	Week1	Week2	Week3	Week4
1	1	1	25	63
2	63	25	1	1
3	1	63	1	25
4	25	1	63	1

3.1.3 Fairness oriented shift scheduling problem model

The scheduling problem we handled in this chapter was modeled as two consecutive problems in previous studies. Tour scheduling problem and employee assignment problem [**82**] are jointly sufficient to define this particular scheduling issue. Note that, tour scheduling problem reduces to set covering problem [**18**] which is NP-Hard. Similarly employee assignment problem reduces to the generalized assignment problem [**21**] which is also shown to be NP-Hard. Since FOSSP corresponds to the combination of both problems, it is also NP-Hard. Solving the problem as two sub problems offers only sub-optimal solutions, however whole parts of the issue are significant and we need to find a

solution for an integrated single problem. FOSSP is the first sample that put those problems together and solves them as a unique problem in literature, to best of our knowledge. In the following, we provide the necessary notation and then the mathematical formulation of FOSSP.

Notation:

x_{ws} is the number of times schema *s* is used for each week *w*.

S is the set of schemas. (Remember a schema is a seven letter string denoting the shift types and off days. Each schema is denoted by a number from 1 to 63.)

T is the time slots of week. (There are totally 21 time slots in a week, one day contains three shifts (planning time slot), seven days comprise 21 time slots similarly. Each time slot is denoted by a number from one to 21)

E is the set of employees,

W is the set of weeks,

c_s is the cost of schema *s*,

r_t is the requirement (number of required workers) at time slot *t*.

$$A_{ews} = \begin{cases} 1, & \text{if employee } e \text{ assigned to schema } s \text{ in week } w \\ 0, & \text{otherwise} \end{cases}$$

A_{ews} is the assignment variable and shows the assignments of each employee *e* for each week *w* and for each schema *s*. *A* is equal to one if an employee *e* is assigned to schema *s* in week *w*, and it is equal to zero otherwise.

$$b_{st} = \begin{cases} 1, & \text{if schema } s \text{ covers timeslot } t \\ 0, & \text{otherwise} \end{cases}$$

b_{st} indicates the time slot coverage of each schema. If schema *s* comprises time slot *t* than; *b* is equal to one, it is equal to zero otherwise. $M = \frac{\sum_s \sum_w \sum_e A_{ews} * c_s}{|E|}$ is the mean cost value for all employees, calculated by total of schema assignment number and schema cost multiplication, divided by the number of employees.

$N_e = \sum_w \sum_e A_{ews} * c_s$ is the all weeks' total cost value. This value is calculated separately for each employee. If employee *e* is assigned to schema *s* during week *w*, A_{ews} variable is equal to one and schema cost is added to the total. The sum off all weeks' schema cost is

calculated and divided by total number of planning week number W. So that mean cost value is calculated for each employee.

$D_e = (M - N_e)^2$ is the sum of squared distances of each employee individual total N_e and the general mean total *M*. The sum of square deviations which measures the pairwise differences is one of the inequality indices widely applied in the literature [**83**], [**84**].

$$\text{minimize} \sum_s \sum_w \sum_e A_{ews} * c_s \quad (3.1)$$

$$\text{minimize} \sum_e D_e \quad (3.2)$$

$$\sum_s A_{ews} = 1 \text{ for } \forall\, e, w \quad (3.3)$$

$$\sum_e A_{ews} = x_{ws} \text{ for } \forall\, w, s \quad (3.4)$$

$$\sum_s x_{ws} * b_{st} \geq r_t \text{ for } \forall\, w, t \quad (3.5)$$

The problem model has two objective functions. The first objective function aims to minimize total assignment costs. The second objective aims minimizing unfairness. Constraint 3.3 means that each employee is assigned to only one schema in each week. Constraint 3.4 ensures that the total number of employees assigned to the schema *s* for each week is equal to x_{ws} (number of total employee assignment in each week and for each schema). Constraint 3.5 ensures that requirement in each time slot *t* (that is, the required number of employees) is satisfied.

$$A_{ews} * b_{s(21)} + A_{e(w+1)s1} * b_{(s1)1} \leq 1 \text{ for } \forall\, e, w, s, s1 \quad (3.6)$$

$$\sum_{k=16}^{21} A_{ews} * b_{sk} + \sum_{k=1}^{3} A_{e(w+1)(s1)} * b_{(s1)k} \geq 1 \text{ for } \forall\, e, w, s, s1 \quad (3.7)$$

$$\sum_{k=3q+1}^{21} A_{ews} * b_{sk} + \sum_{k=1}^{3q} A_{e(w+1)(s1)} * b_{(s1)k} \leq 6 \text{ for } \forall\, e, w, s, s1 \text{ and } q = (0..6) \quad (3.8)$$

s,s1∈S,t,k∈T,e∈E,w,w+1∈W

Variables *w+1*and *s1* are used in constraint 3.6 to 3.8. *w+1* demonstrates the following week and *s1* corresponds to the schema assignment of the following week.

Additional constraints (Equations. 3.6, 3.7, 3.8) may be explained as follows: Constraint 3.6 carries out the rule; if employee has night shift on Sunday, (s)he cannot be assigned to day shift on adjacent Monday. Time slot 21 corresponds to Sunday night shift and time slot one corresponds to Monday day shift. This constraint prevents assigning any two employees to adjacent shifts for consecutive weeks *w* and *w+1*. Another legal regulation is; if an employee has weekend off, he/she cannot be off on Monday (the first day of the next week), implemented by constraint 3.7. Time slot numbers from 16 to 21 embraces weekend period, while time slot one to three corresponds to three shifts of Monday. Total assignment for those three days must be greater than or equal to one. Hence, if someone had weekend off, that person will work on Monday at next week (*w+1*), or in other words an employee cannot be off during consecutive three days. Constraint 3.8 enforces the rule; an employee cannot work consecutive more than six days. If *q* is equal to zero it checks only the first week, if *q* is equal to one it calculates the total for six days of first week starting from Tuesday and one day (Monday) from second week and so on.

The proposed model is verified using GAMS software utilizing CPLEX solver. Even small problem instances took many hours to produce an acceptable result. GAMS is capable for solving the model, but it is poor in terms of execution time, especially for large scale problems. Therefore, we have implemented heuristic based solution techniques, and developed a novel hybrid technique described in detail in the next section.

3.2 Proposed Solution Methods

In this section, the solution methods are clarified. Initial individual generation process for FOSSP, hyper heuristics, simulated annealing, migrating birds optimization algorithms and a novel hybrid algorithm are explained respectively.

3.2.1 Initial individual generation for FOSSP

Recall from the small example that the ultimate output of FOSSP is a matrix containing schema numbers assigned to employees for each week. Therefore, a solution (also called as an individual) to a problem instance will refer to a feasible output in the remaining of the thesis. Again recall that, due to the nature of the problem, same set of schema numbers are used in each week with different permutations. A sample FOSSP individual is illustrated in Table 3.5 for ten employees and four weeks. Employee number one is assigned to schema number five in week one, employee number two is assigned to schema

twenty in week one and so on.

Table 3.5. Sample individual illustration for ten employees and four weeks.

Week	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10
1	5	20	30	35	40	53	58	8	18	27
2	8	18	27	30	35	40	53	58	5	20
3	8	18	27	53	58	40	30	35	5	20
4	40	53	27	30	35	8	18	58	5	20

We want to remind that requirement list is updated periodically before preparing the schedules and the requirements are same for all weeks in planning horizon. Initial individual creation progress of FOSSP is as follows: Schemas are assigned to employees for the first week satisfying the requirements in each time interval. The schedules of subsequent weeks (week two, week three,..) are constructed by shuffling assigned schema values in week one. Additional controls for meeting the requirements and controlling official rules are also taken into account.

3.2.2 Hyper heuristics

Hyper Heuristic (HH) is a neighborhood search technique used for selecting, applying and managing low-level heuristics and they are widely used in solving combinatorial optimization problems [**34**]. Pseudo code of the HH is presented in Figure 3.1. HH has two important functionalities; one of which is heuristic selection mechanism, the other one is the move acceptance mechanism. Heuristic selection mechanism is responsible for selecting a low-level heuristic within the pool. Move acceptance mechanism decides to accept new solutions or not. In our implementation simple random heuristic selection, random permutation heuristic selection and adaptive searching heuristic selection strategies are used. Only Improvement and Monte Carlo acceptance strategies are exploited as acceptance strategies.

```
Initialization i =0
Initial candidate solution S_0
Final solution S_f
S_New =S_0, S_f=S_0
while(i<maxIterations)
    Select heuristic according to heuristic selection strategy
    S_New=applySelectedHeuristic(S_f)
    if (S_New is accepted according to move acceptance strategy)
        S_f=S_New
    end if
    i++
end while
return S
```

Figure 3.1. Pseudocode of HH.

Now let us explain the low level heuristics applied for tackled problems respectively.

3.2.2.1 Low level heuristics for FOSSP

There are totally four low level heuristics for FOSSP in this chapter. Each heuristic checks the additional constraints specified in the model.

Heuristic 1 (H1): H1 selects a random schema among the one those are already assigned to an employee. Another random schema is selected with tournament selection with size five. Five schemas are selected randomly among 63 schemas, one of those five schemas is chosen as second schema. Tournament selection increases the possibility of choosing a lower cost scheme.

Heuristic 2 (H2): H2 randomly selects two employees for each week and swaps their schema assignment values.

Heuristic 3 (H3): H3 selects two employees randomly for each week and inserts the schema value in subsequent location to the next place in prior employee location.

Heuristic 4 (H4): H4 selects two employees randomly for each week and inverts schema assignment values between these employee locations.

3.2.2.2 Low level heuristics for QAP

Four low level heuristics are implemented on QAP. Those may be explained basicly as follows:

Swap: Swap heuristic simply selects two points in a QAP individual and relocate their places.

Insert: Two random points are selected within an individual and the subsequent one is placed right after the first random point. All points between selected points move one step to the right.

Inverse: Two random points are selected within an individual. All assignments between the selected points reversed.

Scramble: Two random points are selected on an individual. Assignments between the selected points are scrambled.

3.2.2.3 Heuristic Selection strategies

Selecting the low level heuristics require a strategy. We applied three different heuristic selection strategies. These are simple random (SR), random permutation (RP) and adaptive searching (AS).

- SR randomly selects one of the low level heuristics and applies only once.
- RP firstly constructs a permutation of low-level heuristics and applies each one once according to the permutation.
- AS is a smart selection mechanism that we designed in this chapter. Each low level heuristic has an equal initial score (*Ri*) and those scores are updated during the run. If heuristic improves the solution its score is increased as much as the change quantity (*Cq*), if it worsens the solution its score is decreased as much as *Cq*. A successful heuristic's score may not exceed predefined upper boundary (*Ub*), analogously an abortive heuristic's score may not be less than the predefined lower boundary (*Lb*). The heuristic with greater score has more probability to be selected. It is applied once and the scores are updated according to its performance.

3.2.2.4 Acceptance Strategies

Once a low level heuristic is selected and applied, the acceptance of new solution also requires a strategy. We applied two strategies, these are Only Improvement (OI) and Monte Carlo (MC).

- OI acceptance strategy only accepts better solutions.
- MC acceptance strategy accepts better solutions certainly. Additionally, it accepts worse solutions with a small probability (Monte Carlo constant) to not to get stuck at the local optima.

3.2.3 Simulated annealing

Simulated annealing (SA) is a well-known local search algorithm proposed by Kirkpatrick et. al. [**85**], inspired from the annealing process of metal work. A starting temperature *T* is determined. The larger this value, the more inferior solutions are encouraged. After a predetermined number of iterations, *T* is set to *T/a*. When *a* is large, the temperature decrease ratio is large and the acceptance of inferior solutions become less likely at a greater rate. When temperature becomes less than a predefined value, SA stops (the stopping condition). Number of iterations at each temperature is limited by *R* and greater values of *R* correspond to slower cooling, that is, more neighbour solution trial is occurring when there is a greater likelihood of inferior exchanges being accepted. In our

implementation, R value changes and increases by an initially specified b value. When a random neighbor is generated, it is compared with the previous solution. If it is better than the previous one, it is accepted. If it is worse than the previous one it may still be accepted with a small probability. This probability is calculated based on the difference of old and new solution quality and the temperature. The pseudo code of SA is given in Figure 3.2.

```
Generate a random initial solution and indicate it as current solution, cs,
initialize temperature T
Best solution(bs) = cs
While termination condition is not satisfied
    For R times
        Obtain a neighbor solution, ns of cs
        delta=cost of ns – cost of cs
        if delta < 0
            cs = ns
        else if random() <e^(|delta|/T)
            cs = ns
        if cost of ns < cost of bs
            bs = ns
    end for
    T = T / a
    R = R * b
end while
```

Figure 3.2. The pseudo code of SA.

3.2.4 Migrating birds optimization

Migrating Birds Optimization (MBO) is a neighborhood search technique proposed by Duman et. al. [**45**], inspired from the V formation flight of the migrating birds. It has been recently proposed but successfully applied in various research areas such as task allocation problem [**86**], flow shop scheduling problem [**87**]-[**89**], U-shaped assembly line balancing problems with workers assignment [**74**], continuous functions [**90**].

Pseudocode of MBO is given in Figure 3.3. There are some essential parameters for MBO those are number of birds (*nob*), number of neighbours (*non*), number of flakes (*nof*) and overlap factor (*olf*). At the beginning "*nob*" amount of initial solutions are randomly generated . The locations of those solutions are determined on a virtual V formation. The best candidate solution or the leader is located at the sharpest end of “V”, then the neighbours are located one beyond another. The algorithm runs until the termination condition is satisfied. The leading solution is improved by exploited techniques. The neighbors next to the leader are improved subsequently by the help of individuals created by the leader. Each solution go through the same process up to the last individual respectively. Each solution is improved by the help of its front neighbours. Consequently,

the solutions except the leading solution of the flock have the opportunity to be developed by its front neighbours. This process is repeated "*nof*" times. Then, the leading solution is moved to the end of the flock and the leader is updated by one of its followers. When the termination condition is satisfied, the best solution in the flock is presented as the outcome of the MBO algorithm.

Generate *nob* initial solutions randomly and place them on a hypothetical V formation arbitrarily
While termination condition is not satisfied
  for *nof* times
    Try to improve leading solution by generating and evaluating non neighbors of it
    foreach solutions s_i I in the flock (except leader)
    Try to improve s_i by evaluating (*non-olf*) neighbors of it and *olf* unused best neighbors from the solution in the front
    end for
  end for
  Move the leader solution to the end and forward one of the solutions following it to the leader position
end while

Figure 3.3. Pseudo code of MBO.

3.2.5 A new hybrid algorithm: HHMBO

Now, we present a novel hybrid technique by integrating various hyper-heuristic techniques with the migrating birds optimization algorithm. To the best of our knowledge, this work is the first attempt of hybridizing MBO with hyper-heuristics.

MBO is a meta-heuristic algorithm that explores new solutions by single step heuristics as illustrated in Figure 3.4. Neighbor generation technique is fixed and does not change throughout the execution of the algorithm. Therefore the performance of classical MBO is much dependent on the neighbor generation technique. However, giving chance more than one heuristics with different characteristics to discover search space may increase its exploration capability. Based on this idea, we integrated the recently presented MBO algorithm, with the hyper-heuristic principles. Instead of a single discovery method we took the advantage of exploiting a set of low level heuristics in order to construct the neighbor solutions in the population.

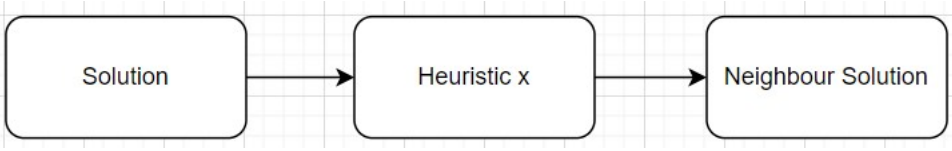

Figure 3.4. Neighborhood generation illustration in classical MBO.

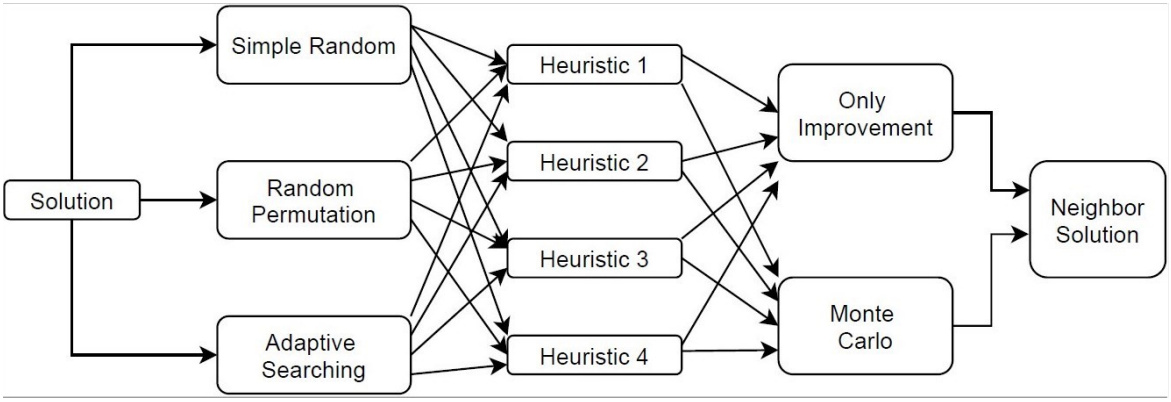

Figure 3.5. Neighborhood generation mechanism illustration in HHMBO.

Neighborhood generation process of hyper heuristics embedded migrating bird optimization (HHMBO) is presented in Figure 3.5. The neighbor solution generation part of MBO is hybridized by HH characteristics which is capable of managing heuristic pool. A set of heuristic selection strategies are handled for choosing the low level heuristics. In our implementation we have applied simple random, random permutation and adaptive searching heuristic selection strategies. The new neighboring solution is accepted or rejected due to the current move acceptance specifications. We have implemented only improvement and monte carlo move acceptance strategies. HHMBO operates in different ways according to the adjusted heuristic selection strategy and the move acceptance strategy. As a result, the HHMBO types emerged in the application can be listed as follows. Simple random only improvement (SR_OI), simple random monte carlo (SR_MC), random permutation only improvement (RP_OI), random permutation monte carlo (RP_MC), adaptive searching only improvement (AS_OI), adaptive searching monte carlo (AS_MC). HHMBO generates neighbor solutions by one of those hyper heuristic variations. The pseudo code of the algorithm is given in Figure 3.6.

Generate *nob* initial solutions randomly and place them on a hypothetical V formation arbitrarily
Determine the heuristic selection strategy (Simple random, random permutation or adaptive searching)
Determine the move acceptance strategy (Only improvement or monte carlo)
While termination condition is not satisfied
for *nof* times
Try to improve leading solution by generating and evaluating non neighbors of it applying selected low level heuristic
for each solutions s_i I in the flock (except leader)
Try to improve s_i by evaluating (*non-olf*) neighbors of it and *olf* unused best neighbors from the solution in the front
Accept or reject the neighbour solution according to current move acceptance strategy
end for
end for
Move the leader solution to the end and forward one of the solutions following it to the leader position
end while

Figure 3.6. The pseudo code of HHMBO.

HHMBO offers a collaboration of hyper heuristic and migrating birds optimization algorithm. HH provides an extensive exploration capability due to their ability to quickly adapt according to the problem instance at hand. MBO exploits the leader solution more detailed each time and the leader is updated in each generation. HHMBO takes the advantage of exploring the solution space by the exploration talent of HH in addition to the exploitation skills of MBO. The properties of the HHMBO which distinguishes it from the other meta-heuristic approaches may be listed as follows:

- A number of solutions running in parallel,
- The benefit mechanism among the solutions,
- Exploiting the leader solution more than the others in each generation,
- Exploring the search space by multiple operators where the best strategy is determined by HH.

3.3 Experimental Setup

In this section, we provide the details of computational experiment settings. Firstly the dataset used in experiments is described. Then the parameter settings of MBO, HH, SA and HHMBO are explained. Lastly, the benchmark data set is clarified.

The GAMS is used to verify and test the mathematical model. All algorithm

implementations are done using Java programming language on Eclipse IDE running on Intel i7-5500U CPU, and 16 GB memory computer operated by Windows 10 operating system. Besides the basic constraints on the formulation presented, additional legal regulation constraints have also been applied while implementing each solution technique.

3.3.1 Data preparation

We had a limited amount of real data provided by our collaborator company. However, we needed a great number of experimental data to achieve a fair comparison among solution techniques we had implemented. Therefore, a simple application is prepared, that generates synthetic data, which has the general characteristics of the real data. Requirements are selected randomly among a certain range, where lower and upper boundaries are calculated based on employee number. A real input dataset for 200 employees and sample synthetic requirement data values is illustrated in Table 3.6. The employee numbers are shown in the top row and it varies from 30 to 500. This illustration comprises a sample requirement input for varied number of employees. First column of the table contains the time slot ids, each one corresponds to a day and a shift type. The requirements in each time slot for a company with 200 (real), 30, 50, 100, 150, 200 and 500 employees are given in the subsequent columns.

Table 3.6. Sample real input and sample synthetic inputs for 30, 50, 100, 150, 200 and 500 employees.

		Real Data	**Synthetic Data**					
Id	**Time slot meaning**	**200**	**30**	**50**	**100**	**150**	**200**	**500**
1	Monday day shift	58	3	12	15	33	12	36
2	Monday evening shift	36	8	11	2	16	12	30
3	Monday night shift	50	8	7	23	9	42	76
4	Tuesday day shift	58	5	6	20	28	10	2
5	Tuesday evening shift	36	9	10	4	17	7	60
6	Tuesday night shift	50	9	7	1	1	34	41
7	Wednesday day shift	58	6	9	13	25	29	94
8	Wednesday evening shift	36	5	1	16	26	4	40
9	Wednesday night shift	5	3	4	14	15	40	3
10	Thursday day shift	58	6	8	6	28	40	88
11	Thursday evening shift	36	3	11	16	25	27	31
12	Thursday night shift	50	7	5	19	30	12	78
13	Friday day shift	58	8	7	5	33	2	103
14	Friday evening shift	36	1	4	14	16	17	89
15	Friday night shift	50	8	6	17	6	26	48
16	Saturday day shift	2	6	10	2	14	24	80
17	Saturday evening shift	5	3	11	1	24	1	96
18	Saturday night shift	5	2	2	5	15	10	79
19	Sunday day shift	5	1	13	1	1	39	55
20	Sunday evening shift	49	1	11	23	26	35	56
21	Sunday night shift	36	5	10	22	14	43	40

3.3.2 Parameter settings

MBO and SA perform a unique heuristic technique for improving solutions. Since we have four different heuristic techniques, we have conducted an experiment to understand which of the four operators in the system is compatible with which algorithm. In order to assess a fair comparison among all applied techniques, we set the stopping criterion of the algorithms based on the number of instances (*noi*) created while each algorithm is running. This experiment is done on each of the datasets, while number of instances (*noi*) is set to 10,000 and each case is repeated 50 times. Heuristic operators are listed from the best performing one to the least performing one in Table 3.7. H1 through H4 correspond to heuristic one through four, respectively, which are explained in Section 3.2.2.1 . Swap, insert, inverse and scramble are applied on QAP and explained in Section 3.2.2.2 .

Table 3.7. The performances of heuristic operators on FOSSP is tested. Heuristic operators are listed from the best performing one to the least performing one.

FOSSP	**Low level heuristics**	**QAP**	**Low level heuristics**
MBO	**H2**, H4, H3, H1	MBO	**Swap**, Insert, Inverse, Scramble
SA	**H1**, H3, H4, H1	SA	**Swap**, Insert, Inverse, Scramble

In order to get the best performance from the algorithms for FOSSP, we need to use their best performing values. These best performing values are discovered with fine tuning experiments. The number of solutions that algorithm can generate while surfing in the solution space is limited to 50,000 for each algorithm. The experiments are repeated ten times and the results acquired from the average performances are represented in Table 3.8 and Table 3.9. Best parameter values are emphasized in bold font.

Table 3.8. Parameter fine tuning experiment results for MBO and HHMBO parameters on FOSSP, achieved from ten runs when noi is set to 50,000 for each algorithm.

		HHMBO					
Parameter	**MBO**	**SR_OI**	**SR_MC**	**RP_OI**	**RP_MC**	**AS_OI**	**AS_MC**
Nob	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**
Non	**3**,5,7	3,**5**,**7**	3, **5**,7	3, **5**,7	3, **5**,7	3,5,7	**3**,5,7
Nof	5, **10**	**5**, 10	**5**, 10	5, **10**	**5**, 10	5, **10**	**5**, 10
Olf	**1**,2,3	**1**,2,3	**1**,2,3	1,**2**,3	**1**,2,3	**1**,2,3	1,**2**,3
Monte Carlo Constant			0.0005, **0.001,** 0.005		0.0005, 0.001, **0.005**		0.0005, 0.001, **0.005**

Table 3.9. Parameter fine tuning experiment results for SA and Adaptive Searching parameters on FOSSP, achieved from ten runs when noi is set to 50,000 for each algorithm.

Parameter	**SA**	**Parameter**	**Adaptive Searching**
T	**50.000**, 75.000, 100.000	Ri	10, **15**, 20
A	**1.1**, 1.2, 1.3	Cq	0.5, 0.8, **1**
R	10, **20**, 30	U b	**40**,50,60,100
B	**1.1**, 1.2, 1.3	Lb	**0**, 5, 10

3.3.3 Experimental setup of benchmark data

Apart from the specific problem defined in this chapter, computational experiments are also conducted with QAP to measure the success of the HHMBO. Computational experiments presented has been done using QAPLIB benchmark dataset where QAP instances are available together with their best known solutions [**91**]. Fine tuning experiments are handled to find out the best performing values. The number of solutions that the algorithm can generate while navigating the solution space is limited to 50,000. The experiments were repeated ten times and the results from the average performances are shown in Table 3.10 indicating the best parameter values in bold.

Table 3.10. Parameter fine tuning experiment results for MBO and HHMBO variations on QAPLIB. The best performing parameters are emphasized in bold font.

		HHMBO							
Parameter	**MBO**	**SR_OI**	**SR_MC**	**RP_OI**	**RP_MC**	**AS_OI**	**AS_MC**	**Parameter**	**Adaptive Searching**
nob	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	5, 21, **51**	Ri	10, 15, **20**
non	**3**,5,7	**3**,5,7	3,**5**,7	**3**,5,7	3,**5**,7	**3**,5,7	**3**,5,7	Cq	0.5, 0.8, 1
nof	5, **10**	5, **10**	**5**, 10	**5**, 10	5, **10**	5, **10**	**5**, 10	U b	40,50,60,100
olf	**1**,2,3	**1**,2,3	**1**,2,3	**1**,2,3	**1**,2,3	**1**,2,3	**1**,2,3	Lb	0, 5, 10
Monte Carlo Constant			0.0005, **0.001**, 0.005		**0.0005**, 0.001, 0.005		**0.0005**, 0.001, 0.005		

3.4 Results and Discussion

In this section, we present and analyze the results of computational experiments. Firstly, the impact of second objective on the FOSSP model is discussed. The first objective value

is increased within a particular tolerance ratio; then results for the maximization and the minimization of the second objective are compared. Secondly, the results obtained by heuristic solution techniques on FOSSP are presented. Lastly, experimental outcomes on QAPLIB are presented.

3.4.1 Comparison of the worst and best second objective values for FOSSP model

The FOSSP model is dual-objective, however we have solved it as a single objective model on the GAMS using CPLEX solver. The model is firstly solved for minimizing the first objective that aims decreasing schema assignment costs. Then it is solved again caring the second objective function where the objective value found in the first run is added as a constraint to the second run. The second objective of FOSSP model aims to take fairness aspect into account. In order to present the impact of the second objective, this experiment compares the best and worst value of the second objective, which has the optimal first objective value. Best values for the second objective are calculated by the minimization of the second objective, as shown in the original model. The worst values for the second objective are calculated by reversing the model sense into maximization for the second objective. First objective value is added as another constraint with an allowed tolerance ratio *alpha*, minimum schema assignment cost and maximum fairness are preserved.

Table 3.11. Comparison of the best and worst fairness benefit values provided by the second objective of FOSSP model for tolerance value of 1% for the first objective value.

Number of Employees	Worst Fairness Value		Best Fairness Value		Improvements in means %
	Mean	Stdev	Mean	Stdev	
30	65.7	34.7	57.1	29.9	15.06%
50	15.2	5.7	13.1	4.7	16.03%
100	78.6	40.7	67.2	35.6	16.96%
150	305.8	102.5	257.6	95	18.71%
200	53.3	20.2	44.3	18.4	20.32%
500	28.6	7.4	23.4	6.7	22.22%
Overall					18.22 %

In the first experiment the *alpha* value is used as 1%. In this test case, assignment costs is not wanted to be increased at all, tolerance value is used as much as an error margin. Experiment results done for employee number 30 to 500 is reported in Table 3.11. Results show that the worst fairness value is improved (decreased) when we relaxed the first objective function with *alpha* coefficient. The improvement slightly increases by the

increment of employee number in the experiments. For the highest employee number or the most challenging instance unfairness value is improved up to 22.22% due to the particular given limits. The reason for this limitation is that the total schema assignment costs must be simultaneously considered and fair but expensive solutions must be prevented.

In the second experiment, the *alpha* value is increased to 5%. This means that the assignment costs can increase 5% in order to provide more fair employee schedules. In this case there is a better increase in the mean improvement ratios compared to the experiment when *alpha* value was 1%. As can be seen on Table 3.12, fairness improvement increased approximately 10%, while the increase reaches up to 39.22%. This shows that model can provide more fair schedules when the assignment costs are increased even a small amount.

Table 3.12. Comparison of the best and worst fairness benefit values provided by the second objective of FOSSP model for tolerance value of 5% for the first objective value.

Number of Employees	Worst Fairness Value		Best Fairness Value		Improvements in means %
	Mean	Stdev	Mean	Stdev	
30	68.4	37.4	56.9	32.6	20.21%
50	81.3	43.4	67	38.3	21.34%
100	52	22.9	42.1	21.1	23.52%
150	318.5	108.2	251.4	97.7	26.69%
200	17.9	8.4	12.9	6.4	38.76%
500	32.3	15.1	23.2	8.4	39.22%
Overall					28.29 %

In the last experiment of this section, the *alpha* value is increased to 10%. The improvement is significant for experiments in all employee numbers. The results are shown on Table 3.13, fairness increased significantly, while the increase reaches up to 50.46%. This result show that when the *alpha* value increases, the fairness has increased significantly in the solution of FOSSP.

Table 3.13. Comparison of the best and worst fairness benefit values provided by the second objective of FOSSP model for tolerance value of 10% for the first objective value.

Number of Employees	Worst Fairness Value		Best Fairness Value		Improvements in means %
	Mean	Stdev	Mean	Stdev	
30	72.7	42.4	55.2	37.6	31.7 %
50	337.8	113.2	255.7	102.7	32.11 %
100	90.6	48.4	65.3	43.3	38.74 %
150	40.6	20.1	28.5	13.4	42.46 %
200	25.2	13.4	17.2	11.4	46.51 %
500	65.3	27.9	43.4	26.1	50.46 %
Overall					40.33 %

3.4.2 Results obtained by utilizing synthetic FOSSP dataset

In this subsection, results of heuristic based solution techniques on FOSSP are presented.

Solution techniques are tested for employee numbers 30, 50, 100, 150, 200 and 500. We have generated 10 sample synthetic inputs for each case. The sample synthetic problem instances are solved using MIP CPLEX solver in GAMS. We have limited the runtime as ten minutes for each test and aimed to find out the best solutions. For the heuristic based techniques we have also limited the runtime as ten minutes for each test on the specified machine.

Mean percentage deviations of cost values from the best found solutions gained from the ten runs are demonstrated in Table 3.14. For all employee numbers HHMBO variants discover the best results. For employee numbers 30 and 50, HHMBO SR_MC variant outperforms the others on the mean. When employee number increases to 100,150, 200 and 500 HHMBO RP_OI variant outperforms the other techniques.

Table 3.14. Mean percentage deviations of all algorithms from the best found solution by MIP solver.

			HHMBO						HH					
Emp. number	**MBO**	**SA**	**SR_OI**	**SR_MC**	**RP_OI**	**RP_MC**	**AS_OI**	**AS_MC**	**SR_OI**	**SR_MC**	**RP_OI**	**RP_MC**	**AS_OI**	**AS_MC**
30	8.6	5.2	3.4	**1.0**	4.2	5.1	5.1	6.3	6.8	7.2	6.3	6.6	11.7	12.0
50	9.3	8.9	5.6	**1.1**	4.1	7.1	9.6	11.7	7.9	8.3	7.4	7.7	12.8	13.1
100	7.1	15.0	8.4	2.6	**2.1**	5.5	6.0	12.5	12.2	12.6	9.2	9.5	17.1	17.4
150	6.9	30.9	8.8	7.4	**1.5**	4.7	8.4	15.3	12.7	13.1	9.7	10.0	17.6	17.9
200	7.8	45.1	8.6	8.2	**1.3**	8.4	12.2	17.1	12.9	13.3	12.4	11.7	17.8	18.1
500	10.0	85.9	12.7	8.5	**1.2**	7.5	10.2	17.9	21.0	22.1	15.5	15.8	25.9	26.2
Overall	8.3	31.8	7.9	4.8	**2.4**	6.4	8.6	13.5	12.3	12.8	10.1	10.2	17.2	17.5

When the performances are analyzed in accordance to the increase of employee number, it is apparent that MBO and HHMBO variants retain their accomplishment as complexity rises. However, MBO is 7.6%, 8.2%, 5%, 5.4%, 5.5% and 8.8% worse on the mean than the best performing HHMBO variant for employee numbers 30, 50, 100, 150, 200 and 500 respectively. It is clear that both algorithms are successful in spite of increasing problem complexity. However, HHMBO variants obtains slightly better results than MBO in each condition.

The best performing HHMBO variant is 4.2%, 7.8%, 12.9%, 29.4%, 43.8% and 84.7% better than SA for employee numbers 30, 50, 100, 150, 200 and 500 respectively. As the complexity of the problem increases, the performance of SA decreases and the difference between the two algorithms increases. Best performing HHMBO variant is 5.3%, 6.3%, 7.1%, 8.2%, 10.4 and 14.3% better than the best performing HH variant for employee numbers 30, 50, 100, 150, 200 and 500 respectively. As the number of employees increases, the distinction between the algorithms involving in the experiment becomes more prominent. HHMBO-RP_OI produced the best results for the most challenging experiment, when employee number is equal to 500. HHMBO versions figure out lower (better) results when they are compared with their HH versions.

HHMBO (RP_OI) obtained significantly better results in complex FOSSP instances. RP_OI applies each four problem specific heuristics consecutively due to the permutation order. Each heuristic is applied to equal number of instances. Each heuristic has a particular ability to discover search space. Accomplishment of HHMBO (RP_OI) exposes that when the algorithm is given a large number of search ability, using heuristics equal numbers of times help us to find the best results. HHMBO is successful on catching values close to global optimum. HHMBO provides diversification along the search space by applying variety of heuristics alternately that is why it is promising for solving combinatorial optimization problems.

We have also applied t-test to check if the differences are statistically significant. The best performing HHMBO-RP_OI and all other algorithms in the experiment are tested and the statistical results obtained by two-tailed t-test are given in Table 3.15. We observed that the p-values for pairwise t-tests among the best performing proposed variant HHMBO-RP_OI and the state-of-the-art algorithms is 0.002% on the average. The results imply that HHMBO-RP_OI is statistically different from other algorithms, indeed.

Table 3.15. The t-test results of HHMBO-RP_OI and the state of the art algorithms on FOSSP.

Algorithm 1 <-> Algorithm 2	**30**	**50**	**100**	**150**	**200**	**500**
HHMBO-RP_OI <-> MBO	8.30E-03	5.90E-03	1.10E-04	1.70E-02	2.70E-23	1.40E-11
HHMBO-RP_OI <-> SA	2.60E-05	2.70E-05	5.20E-05	6.90E-06	1.40E-21	7.70E-09
HHMBO-RP_OI <-> HH-SR_OI	5.40E-05	5.10E-05	1.30E-08	3.40E-12	1.40E-17	4.90E-12
HHMBO-RP_OI <-> HH-SR_MC	2.70E-05	2.60E-04	2.40E-10	3.30E-11	6.10E-18	6.00E-12
HHMBO-RP_OI <-> HH-RP_OI	1.60E-05	2.30E-03	1.90E-11	3.70E-03	5.80E-14	5.20E-05
HHMBO-RP_OI <-> HH-RP_MC	8.00E-05	2.90E-03	6.40E-10	4.90E-04	3.90E-12	4.10E-08
HHMBO-RP_OI <-> HH-AS_OI	4.20E-04	3.50E-03	1.70E-05	2.70E-22	7.80E-16	2.20E-11
HHMBO-RP_OI <-> HH-AS_MC	7.10E-05	2.20E-03	2.90E-06	2.20E-19	2.20E-15	4.00E-15

3.4.3 Results on benchmark problems in QAPLIB

After observing that HHMBO variants get very successful results for the FOSSP instances, we wanted to see their ability in converging optimal solutions on a widely used problem. We applied HHMBO on QAP where MBO is firstly applied and tested. If N is the size of a QAP instance, we limited the run time with N^3 iterations. This corresponded to 0.01 s for a problem of size 12 and to 48 s for a problem of size 100 on the specified machine.

We have utilized 41 QAPLIB instances with different densities. Density corresponds to the percentage of nonzero entries in the flow matrix. The instance names, densities, size of the instances, best known solutions (BKS) and the percentage deviations of heuristic results from BKS are given in Table 3.16. Problem instances are listed from sparse one to the dense one. Only the minimum costs obtained in 10 runs are considered since the aim is to check the ability of HHMBO in finding BKS.

The results indicate that AS_IO variant of HHMBO is accomplished on catching promising solutions. HHMBO (AS_OI) is up to 14.6% better than MBO on converging to the BKS. According to the average percentage deviation gained from 41 instances, HHMBO (AS_OI) converges 7.5% to BKS while MBO is 8.2% close to it. HHMBO (AS_OI) performs 0.7% better than MBO on the average. Best performing variant of this experiment is adaptive searching only improvement HHMBO. Adaptive searching strategy exploits low level heuristics depending on their performances. This adaptive feature ensures choosing the most beneficial low level heuristic for each specific case during the run. It has been observed from the experiments conducted on QAPLIB benchmark problems that HHMBO consistently allows to find better results, albeit slightly.

3.5 Summary

In this chapter, we introduced a novel hybrid computational intelligence algorithm to the literature and the mathematical model of a shift scheduling problem of a manufacturing company defined by including the fairness perspective, which is typically ignored especially in manufacturing industry. The novel algorithm is designed by embedding hyper heuristic (HH) improvements in the migrating bird optimization algorithm (MBO), called HHMBO. In this way, the exploitation capability of MBO is improved and the solution space is explored in a more diversified manner.

Table 3.16. The percentage deviations of heuristic results from the best known solution.

					HHMBO					
Problem	**Density**	**Size**	**BKS**	**MBO**	**SA_OI**	**SR_MC**	**RP_OI**	**RP_MC**	**AS_OI**	**AS_MC**
esc64a	3.17	64	116	0	0	0	58.6	58.6	0	0
tai64c	4.13	64	1,855,928	0	0	0	11.4	10.1	0	0.1
chr22a	8.68	22	6,156	9.6	16.8	13.3	41.2	45.8	8.1	7.1
esc16j	9.38	16	8	0	0	0	0	25	0	0
chr20a	9.5	20	2,192	31.3	55.3	56.6	147.7	132.6	27.9	20.1
chr20b	9.5	20	2,298	25.7	35.9	47.5	67.6	119.4	17.8	36.3
chr18a	10.49	18	11,098	54.2	71	81.6	197.2	190.1	53.8	55.2
esc16i	11.72	16	14	0	0	0	28.6	14.3	0	0
chr15a	12.44	15	9,896	22.6	12.7	40.4	76.2	116.9	35.8	30.1
chr15b	12.44	15	7,990	38.2	70.5	37.7	193.3	168.5	33.4	51.4
chr15c	12.44	15	9,504	53.8	39.2	62.9	95.4	96.6	39.2	42
chr12c	15.28	12	11,156	14.9	28.1	14.3	60.4	46	18.8	8.5
chr12a	15.28	12	9,552	20.5	37.5	19.7	60.2	58.5	19.8	5.7
esc16d	16.41	16	16	0	0	0	12.5	0	0	0
esc16e	16.41	16	28	0	0	0	14.3	14.3	0	0
esc16g	16.41	16	26	0	0	0	15.4	23.1	0	0
esc32d	17.58	32	200	0	3	4	37	35	0	0
esc32c	25.59	32	642	0	0	0	13.7	14	0	0
esc32h	27.54	32	438	0	2.3	2.3	26.5	21	0	0
esc16a	29.69	16	68	0	0	0	8.8	8.8	0	0
scr15	37.33	15	51,140	7	11.3	7	29.4	28.1	6.9	9.2
esc16c	39.84	16	160	0	0	0	5	8.8	0	0
esc16b	71.88	16	292	0	0	0	0	0	0	0
lipa20b	89.25	20	27,076	11.2	14.8	17.2	22	21.9	9.8	11
esc16h	89.84	16	996	0	0	0	1.6	0	0	0
lipa20a	90.25	20	3,683	2.2	3	3	4.5	4.5	2.4	2.7
had12	91.67	12	1,652	0.4	0.7	0.1	3.4	2.4	1	0.8
rou12	91.67	12	235,528	3	4.7	2.8	9.8	9	5.7	4.9
rou15	93.33	15	354,210	5.1	7.7	6.8	13.7	13.4	5	4.8
nug16b	93.75	16	1,240	3.9	6.9	3.4	12.9	13.7	4.5	4.4
els19	94.74	19	17,212,548	4.7	8.7	5.5	39.7	47.3	1.8	2.7
lipa40b	94.81	40	476,581	5.3	21	19.6	26	26.3	3.5	6.5
lipa40a	95.06	40	31,538	1.3	2	2	2.9	2.8	1.6	1.6
lipa50b	95.88	50	1,210,244	17.9	21.8	21.7	25.3	25.5	8.3	11.2
tai60b	98.33	60	608,215,054	1.1	2.2	2.7	40.2	41	0.6	1.1
sko100a	99	100	152,002	0.7	5.9	5.7	13.9	13.9	0.7	2.6
bur26b	100	26	3,817,852	0.1	0.5	0.3	2.5	3	0	0.1
bur26a	100	26	5,426,670	0.1	0.4	0.4	2.8	1.8	0.1	0.2
bur26c	100	26	5,426,795	0	0.2	0.2	3.1	3.1	0	0.1
bur26e	100	26	5,386,879	0	0.3	0.4	3.4	3.6	0	0.1
bur26g	100	26	10,117,172	0	0.1	0.4	2.6	3	0.1	0
Average				8.2	11.8	11.7	34.9	35.9	**7.5**	7.8

The problem that we tackled is called as fairness oriented shift scheduling problem (FOSSP). The model of FOSSP is firstly verified with a solver where the solver is capable on catching global optima with large execution time even for small instances. Therefore, exploitation of heuristic based techniques was an apparent need. In line with this need, we applied HHMBO together with simulated annealing (SA), hyper heuristics (HH) and classical migrating bird optimization (MBO). We have conducted computational experiments in order to assess the performance of the algorithms on synthetic data. Results show the superiority of HHMBO-RP_OI and it is observed that it outperforms its rivals by 9.3% on the average for FOSSP. Besides, it is seen that HHMBO-RP_OI definitely offers better solutions in large problem instances, which is a desired property in practice and gap in the literature. We believe that this good performance is due to integrating the exploitation capability of HH with MBO's exploration capability.

To justify the superiority of HHMBO over the other algorithms, we used QAP instances from the QAPLIB. This experiment was significant to understand whether the achievement of the algorithm is specific to the particular problem. We conducted computational experiments on 41 QAPLIB instances with assorted densities. According to the results of this experiment, AS_OI variant of the new hydrid algorithm could achieve up to 14.6% better results than MBO.

4. AN INVESTIGATION OF NATURE INSPIRED ALGORITHMS ON A PARTICULAR VEHICLE ROUTING PROBLEM IN THE PRESENCE OF SHIFT ASSIGNMENT

This chapter contains the mathematical presentation of the multi-objective vehicle routing problem in the presence of shift assignment. A number of naturally inspired algorithms are applied on the problem and the performance analysis are presented.

The defined model offers an integrated framework of vehicle routing problem and shift assignment problem. Both of those affairs are necessary while preparing business operations in most of the large scale companies. These problems are generally treated separately even though they can help to reduce each other's costs. When people are assigned to the same shift without considering their living addresses, employees far from each other may be assigned the same shift. This causes resources to be wasted while routing. In practice, for some extreme circumstances it is observed that a high-capacity shuttle is assigned to transport a single person or multiple shuttles are exploited when a single vehicle is sufficient to transport employees assigned to a single shift. When there is no clever arrangement of shift assignment, superabundant number of staff may be assigned to a shift or the number of assigned employees may be insufficient. Shift assignment needs the location information of VRP and VRP needs clever shift arrangement capability of shift assignment.

The motivation behind this work has emerged from the employee transportation and shift arrangement requirements of a real company. We bring the workforce scheduling and vehicle routing problems together in a way that has never been done before and consequently vehicle routing problem in the presence of shift assignment (VRPSA) is introduced to the literature. After building the mathematical model of the problem it is verified on a solver and then real large-sized instances taken from a company are solved using a set of evolutionary algorithms. Three novel solution techniques are introduced based on our framework called dynamic neighbor generation. As one of the contributions of this study, dynamic neighbor generation framework may easily be extended to include other multi objective optimization algorithms to increase their exploration capability, and thus it offers an alternative development facility for solving multi objective optimization

problems. Results of computational experiments show that the proposed framework definitely offers promising and robust results in large sized problem instances in terms of hypervolume and inverted generational distance indicators.

We have conducted research on the problem of solving routing and shift assignment processes together. To our best, literature investigation indicates that there is no other defined problem that integrates these two well-known problems in the direction that collaborative company requested from us. Therefore, we have defined a new integrated problem, whose optimization was predicted to reduce operational costs considerably. Exhaustive problem description is given in Section 4.1.

The contributions of this part of the study is three-fold : (i) defining vehicle routing problem in the presence of shift assignment (VRPSA) as well as building and validating the mathematical model of it, (ii) proposing dynamic neighbor generation version of NSGA, NSGA-II, and dynamic neighbor generation version of MMBO and (iii) presenting an easily extendable framework to the optimization research community.

The remainder of this chapter is organized as follows: the problem we tackled is described and then the mathematical formulation of the problem is given in Section 4.1. Existing evolutionary algorithms applied for solving the proposed problem are described in detail in Section 4.2. Proposed solution techniques for solving VRPSA are given in Section 4.3. Experimental setup implicating performance metrics and parameter fine tuning experiments are expressed in Section 4.4. Experimental results together with their discussions are illustrated in Section 4.5. Finally concluding remarks and suggestions for future research are presented in Section 4.6.

4.1 Problem Description

In this section, the tackled problem is presented and clarified. Firstly, the properties and limitations of the problem are explained and then the mathematical model of vehicle routing problem in the presence of shift assignment is presented.

There are two types of working arrangements in the collaborative company; those are regular scheme and shift scheme. While regular scheme employees have steady and normative working hours, shift scheme employees' working hours are flexible. In the regular working regulation, shift hours are fixed between 8 a.m. and 5 p.m.; while shift working system contains a number of different shift types with different start hour, end hour and duration. Employees who are pregnant, on milk leave and have similar special circumstances are assigned to regular working order. All processes involved in this chapter

are designed for planning shift scheme orders.

The collaborative company has three premises in separate locations. The salary received by employees differs according to shift type, shift duration and location of business campus. Shift id, start hour, end hour, duration, and unit wage information are listed in Table 4.1. There are totally 26 different shift types covering different time periods of the day. Unit wage#1 corresponds to the wage paid to employees in location one, unit wage#2 corresponds to the unit wage of employees in location two and unit wage#3 corresponds to the unit wage paid in location three. For example, while shift id is equal to one start hour is 00:00, end hour is 07:45 and the duration is seven hours and 45 minutes. The employee who is assigned to shift id one earns 62 unit for a day in factory location one, earns 68.2 in factory location two and earns 74.4 in factory location three. Those price estimations are given us by our collaborative.

Table 4.1. Shift start-end hours, duration and unit wages for all 26 shift types.

Shift Id	Start Hour	End Hour	Duration	Unit wage#1	Unit wage#2	Unit wage#3
1	00:00	07:45	07:45	62	68.2	74.4
2	00:00	08:00	08:00	64	70.4	76.8
3	00:00	08:30	08:30	68	74.8	81.6
4	00:00	09:00	09:00	72	79.2	86.4
5	00:00	10:00	10:00	80	88	96
6	00:00	12:15	12:15	98	107.8	117.6
7	01:45	08:00	06:15	50	55	60
8	02:00	08:00	06:00	48	52.8	57.6
9	07:00	13:00	06:00	48	48	48
10	07:00	17:00	10:00	80	80	80
11	08:00	10:00	02:00	16	16	16
12	08:00	12:00	04:00	32	32	32
13	08:00	14:00	06:00	48	48	48
14	08:00	18:00	10:00	80	80	80
15	09:00	14:00	05:00	40	40	40
16	09:00	17:00	08:00	64	64	64
17	10:00	17:00	07:00	56	56	56
18	10:00	19:00	09:00	72	72	72
19	11:00	20:00	09:00	72	72	72
20	15:00	02:00	11:00	88	96.8	105.6
21	16:00	02:00	10:00	80	88	96
22	17:00	02:00	09:00	72	79.2	86.4
23	18:00	02:00	08:00	64	70.4	76.8
24	19:00	04:00	09:00	72	79.2	86.4
25	22:00	07:00	09:00	72	79.2	86.4
26	23:00	08:00	09:00	72	79.2	86.4

Employee requirements differ for every 15 minutes period in a day. That is, there are totally 96 timeslot assortments in the requirements dataset. The shortened sample requirement list is given in Table 4.2. Requirements correspond to the number of employees needed at specific short time duration. In the given table, timeslot id one starts at 00:00 and ends at 00:15. No employees are required during timeslot one for all premises. Same information is available for every 15 minutes time frame in a day.

Table 4.2. Sample employee requirements.

Timeslot Id	Start hour	End hour	Requirement#1	Requirement#2	Requirement#3
1	00:00	00:15	0	0	0
2	00:15	00:30	5	6	0
3	00:30	00:45	0	4	0
...	...	...	...	...	...
...	...	...	...	...	...
92	22:45	23:00	12	12	17
93	23:00	23:15	11	11	0
94	23:15	23:30	11	10	0
95	23:30	23:45	11	9	0
96	23:45	00:00	10	7	0

Table 4.3. Shortened coverage matrix. The matrix shows the match of 26 shift types and 96 timeslots. If a shift covers the corresponding timeslot it is indicated by 1 otherwise it is indicated by 0.

			Time slot id											
Shift id	Shift Start	Shift End	1	2	3	4	5	6	91	92	93	94	95	96
1	00:00	07:45	1	1	1	1	1	1 ...	0	0	0	0	0	0
2	00:00	08:00	1	1	1	1	1	1 ...	0	0	0	0	0	0
...	...	...	...	... ...										
15	09:00	14:00	0	0	0	0	0	0 ...	0	0	0	0	0	0
16	09:00	17:00	0	0	0	0	0	0 ...	0	0	0	0	0	0
...	...	...	...	... ...										
25	22:00	07:00	1	1	1	1	1	1 ...	1	1	1	1	1	1
26	23:00	08:00	1	1	1	1	1	1 ...	0	0	1	1	1	1

As mentioned before, the company has 26 different shift types. Each shift covers a set of time slots within specified 96 periods. We have associated shift types along with the timeslots they cover and stored this data in a two dimensional binary coverage matrix. It is not possible to insert the entire matrix to this thesis, as it will take up a lot of space, but the matrix is shortened and given in Table 4.3. As an example, shift id one covers timeslot id one, two, three, four, five and six while it does not cover timeslot id 91, 92, 93, 94, 95 and 96. This matrix helps us analyzing inclusion of shifts and shortens the operating time for

both the solver and the evolutionary algorithms that we apply.

A high-level illustration of the solution procedure to the VRPSA problem is shown in Figure 4.1. The proposed model gets the employee requirements for 96 timeslots, employment costs varying in each shift type and each location, employee locations in coordinates where workers get in the shuttle and the vehicle capacities as inputs. Then, solution techniques developed for VRPSA construct employee shift arrangements and shuttle routes by minimizing the employment costs and minimizing route path lengths. Hereby employee shift arrangements and vehicle routes are produced as outputs. The model aims to minimize both labor costs and transportation costs, which are important expenses for the company.

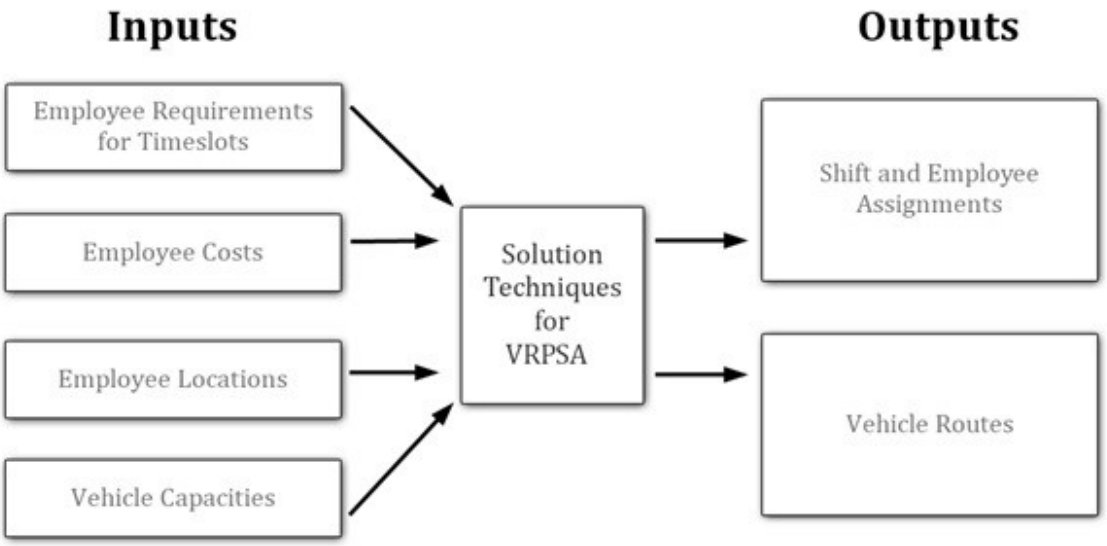

Figure 4.1. VRPSA process diagram with inputs and outputs.

The presented model tends to assign employees whose home locations are close to each other to the same shift and to the same personnel service vehicle, as well as to determine the minimum number of employees to meet the needs specified for each timeslot. Thereby, while a sufficient number of personnel are appointed, efficient use of vehicle capacity is ensured.

The exact equivalent of the problem presented here is the integration of vehicle routing problem and shift assignment problem. Shift assignment problem is a special version of generalized assignment problem [**24**] which is a common NP-Hard optimization problem [**92**]. Also VRP is a well-known NP-hard problem [**46**]. Hereby the presented work is the integration of real-world complex problems with their NP-hard nature.

In the remaining of this section, we provide the notation and then the formulation of

VRPSA.

Notation:

E is the set of employees.

T is the set of timeslots. (There are totally 96 time slots, covering all fifteen minute intervals in one day.)

S is the set of shifts. (There are totally 26 shift types.)

d_{ij} is the Euclidean distance of route from location i to location *j*.

$$x_{ij} = \begin{cases} 1, & \text{if there is a path from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

c_s is the cost of shift *s*.

r_t is the requirement in timeslot *t*.

$$y_{is} = \begin{cases} 1, & \text{if employee } i \text{ is assigned to shift } s \\ 0, & \text{otherwise} \end{cases}$$

$$v_{st} = \begin{cases} 1, & \text{if shift } s \text{ covers timeslot } t \\ 0, & \text{otherwise} \end{cases}$$

N is the set of employee locations and the depot location. *N* is required for routing process.

M is the number of vehicle routes.

Q is the vehicle capacity and q_i is the total load in each location (q_i is equal to one for our case), u_i is the load of a vehicle just after leaving location *i*.

The multi-objective VRPSA formulation has two objective functions. First objective aims minimizing the total distance travelled. Second objective function minimizes the total number and the total cost of employees that assigned to shifts.

$$\text{minimize} \sum_{i \epsilon N} \sum_{j \epsilon N} d_{ij} * x_{ij} \quad \text{for } i \neq j \quad (4.1)$$

$$\text{minimize} \sum_{i \epsilon E} \sum_{s \epsilon S} c_s * y_{is} \quad (4.2)$$

$$\sum_{s\epsilon S} y_{is} \leq 1 \quad \forall\, i\epsilon E \qquad (4.3)$$

$$\sum_{i\epsilon E}\sum_{s\epsilon S} y_{is} * v_{st} \geq r_t \quad \forall\, t\epsilon T \qquad (4.4)$$

$$\sum_{j\epsilon N} x_{ij} = \sum_{s\epsilon S} y_{is} \quad \forall\, i\epsilon E\,, i \neq j \qquad (4.5)$$

$$\sum_{i\epsilon N} x_{ij} = \sum_{s\epsilon S} y_{js} \quad \forall\, j\epsilon E\,, i \neq j \qquad (4.6)$$

$$x_{ij} \leq \sum_{s\epsilon S} y_{is} * y_{js} \quad \forall\, (i,j)\epsilon E \qquad (4.7)$$

$$\sum_{i\epsilon E} x_{i0} = M \qquad (4.8)$$

$$\sum_{j\epsilon E} x_{0j} = M \qquad (4.9)$$

$$u_i - u_j + Qx_{ij} \leq Q - q_j \quad (i,j)\epsilon E, i \neq j \qquad (4.10)$$

$$q_i \leq u_i \leq \mathrm{Q} \quad i\epsilon E \qquad (4.11)$$

Constraint 4.3 means that each employee may be assigned to at most one shift. Constraint 4.4 ensures that the total number of employees assigned to timeslot *t* is greater than or equal to personnel requirement in timeslot *t* by the aid of the coverage matrix v_{st}. Constraint 4.5 means that, if employee *i* has been assigned to any of the shifts, then there may be a path starting from *i*. Similarly, constraint 4.6 means that if employee *j* has been assigned to any of the shifts then there may be path ending at *j*. Constraint 4.7 ensures that, there may be a path between employee *i* and employee *j*, if both of them have been assigned to same shift. Constraint 4.8 and constraint 4.9 ensure the number of routes start and end at the depot is limited by *M*. Constraint 4.10 ensures that the solution contains no sub tours disconnected from the depot. Constraint 4.11 determines the limits of u_i.

4.2 State-Of-The-Art Solution Techniques for MOP

The mathematical model of VRPSA given in the previous section is firstly verified with the Gurobi solver on Python. Even small instances took many hours to produce an

acceptable result. So that, we have applied a set of evolutionary meta-heuristic algorithms which have previously proven their success on MOP. Those are NSGA, NSGA-II and MMBO. In the following subsections we give the details of these algorithms for the sake of a compact work in this chapter.

4.2.1 Non-dominated sorting genetic algorithm

Since non-dominated sorting genetic algorithm (NSGA) has been proposed by [**65**], many NSGA variations are offered. NSGA basically adapts genetic algorithm for multi-objective optimization problems. The working mechanism of genetic algorithm is briefly as follows: An initial population of size *N* is generated using the selected initial population generation strategy. Two parent individuals are chosen among the population. Crossover operator is applied according to crossover probability *c*, mutation technique is applied according to mutation probability *m* and two offsprings are produced and added to the population. This process continuous till the population size reaches *2N*. After that the expanded population is sorted from the best solution to the worst one. The best *N* individuals are transferred to the next population. This process is repeated up to the limits determined in the application. Unlike single objective optimization problems, multi-objective optimization problems have a set of results instead of a single optimal one. This result set is called as non-dominated solution set or Pareto optimal set and non-dominated sorting technique is developed for sorting those results. A non-dominated solution corresponds to a solution that is not dominated by any member in solution set. The pseudo code of NSGA is given in Figure 4.2.

```
Initialize parent population size N
Evaluate the fitness of parent population
    while (not termination-condition) do
    for(i=0;i<N/2;i++) do
        Select two parents
        Generate two offspring by crossover with probability c
        Perform mutation with probability m
        Add offsprings to population and expand the population
    end for
    Apply non-dominated sorting on expanded population size 2N
    Update population by first N individuals from the sorted generation list
end while
Output the non-dominated solutions
```

Figure 4.2. The pseudo code of NSGA.

4.2.2 Fast and elitist non-dominated sorting genetic algorithm

Fast and elitist non-dominated sorting genetic algorithm (NSGA-II) is one of the most popular algorithm that has proven its success in the MOP domain proposed by [**66**]. It has

been applied to MOP in several areas and its many different variations has been suggested. NSGA-II brings out crowding distance concept and revolutionize the individual selection mechanism while constructing next generations in NSGA. Figure 4.3 illustrates the mechanism exhaustively. Let us assume the size of population is $Pt=Qt=N$, Qt is the number of solutions generated by crossover and mutation operators. The expanded $2N$ population formed by the union of Pt with Qt is sorted using non-dominated sorting. The simple NSGA assigns the first N individuals to next population in this stage; however NSGA-II has another mechanism sequentially. The main difference and contribution of NSGA-II is an explicit diversity preserving mechanism called crowding distance. The best individual produced after non-dominated sorting is most probably the solution that is non-dominated. This is followed by the solutions dominated once, twice and so on. Crowding distance defines virtual clusters from those sorted individuals. There are totally Fn clusters including ($F1,F2,..Fn$). The first cluster $F1$ is directly transferred to child population if its size does not overrun the population size N, $F2$ transferred to the child population if the total transferred individuals including $F2$ do not exceed N. When a cluster Fx exceeds the population size than crowding distance is applied on all individuals in Fx. The distance of each member to each other member in Fx is calculated and Fx cluster is sorted according to this crowding distance scores. The best solution according the crowding distance is the solution that is the furthest from others in the cluster. The solutions are sorted from the furthest one to the closest one. Thence, crowding distance gives priority to the most discrete solution among candidates and ensures diversity.

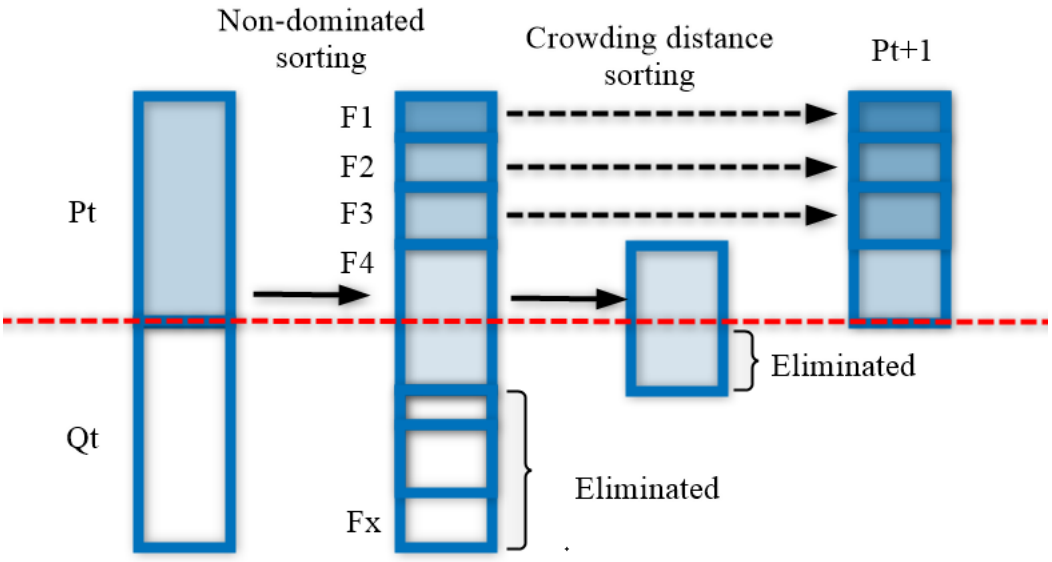

Figure 4.3. NSGA-II working mechanism illustration.

The pseudo code of NSGA-II is given in Figure 4.4. Selection of individuals passed on to the next generation is designed to increase diversity and is different from the NSGA.

```
Initialize parent population size N
Evaluate the fitness of parent population
    while (not termination-condition) do
    for(i=0;i<N/2;i++) do
        Select two parents
        Generate two offspring by crossover with probability c
        Perform mutation with probability m
        Add offsprings to population and expand the population
    end for
    Apply non-dominated sorting on expanded population and determine clusters F(F1..Fx)
    Transfers cluster member toPt+1until they exceed population size N
    if (adding Fk exceeds N)
        Compute Crowding Distance of Fk
    Transfer members of Fk to Pt+1 until population size N is reached
end while
Output the non-dominated solutions
```

Figure 4.4. The pseudo code of NSGA-II.

4.2.3 Multi-objective migrating birds optimization

Migrating birds optimization (MBO) is an algorithm inspired by the V flight formation of migrating birds recently proposed [**45**]. Since it was proposed, the MBO has been successfully applied to various research areas, such as low carbon scheduling [**93**], hybrid cloud systems [**94**], task allocation problem [**86**], flow shop scheduling problem [**95**], land distributing problem [**96**], continuous functions [**90**]. A multi-objective version of migrating birds optimization (MMBO) algorithm is recently proposed by [**97**] includes a shuffling process and applied for solving a hybrid flow shop rescheduling problem. We have applied MMBO without the shuffling process. Pseudo code of MMBO is given in Figure 4.5. The process improves on a set of non-dominated solutions in accordance with the nature of multi objective optimization problems. Results are sorted using non-dominated sorting.

The main idea behind MMBO is the utility mechanism between the solutions lined up in a V-shaped population. The algorithm contains four phases, including initialization, leader evolution, follower's evolution and leader change. In initialization phase parameters are determined and initial population is produced. The second phase the leader is to be improved by evaluating its neighboring solutions generated in its neighborhood. In the third phase followers are evaluated. Each solution in the two split sides of the leader is improved by the help of not only their own neighbors generated in the neighborhood but

also the neighbors received from the solution in front of it. In the last phase when a loop is completed, the leader solution is updated by the help of non-dominated sorting. The leader moves to the end of the left and right lines by turns, then the first solution in the corresponding line is forwarded to become the leader. This process is repeated as many as a predetermined number. Then, the leading solution is moved to the end of the flock and the leader is updated by one of its followers. When the termination condition is satisfied, the best non-dominated solution in the flock is presented as the outcome of the MBO algorithm.

```
Initialize the parameters and the population
While T is not met do
    for i=1 to t do
        Evolve the leader
        Evolve the followers in the left line
        Evolve the followers in the right line
    end for
    Change the leader
end while
Output the non-dominated solutions
```

Figure 4.5. Pseudo code of MMBO.

4.3 Novel Solution Techniques Proposed for Solving MOP

In this section, we have explained the proposed solution techniques applied for solving VRPSA. Three new methods are presented for solving the problem. Definitely, the proposed techniques are applicable for any MOP.

Multi-objective problems have a sophisticated solution space and a single improvement technique is insufficient to obtain promising results. More than one neighborhood search methods with different characteristics are required to obtain more efficient exploration in the solution space. One widely known way of applying multiple heuristic techniques is hyper heuristics. Hyper heuristics (HH) is a meta-heuristic technique formally defined as, a heuristic to select and manage a set of low level heuristics. Many HH variations for MOP are defined up to now, such as choice function based HH [**98**], HH based multi-objective genetic algorithm [**99**], multi-objective hyper heuristic evolutionary algorithm [**100**], multi objective particle swarm optimization based HH [**101**]. HH based techniques and have been applied for solving many different MOPs such as, space allocation and time tabling problem [**102**], 2d packing problem [**103**], job scheduling problem [**104**], aircraft scheduling problem [**105**].

Each meta-heuristic has a unique operating mechanism. HH is a meta-heuristic technique based on managing a number of heuristics. The solution improvement of other meta-

heuristic algorithms is generally restricted by a single neighborhood search method. There is not a sufficient investigation about applying multiple heuristic operators on common meta-heuristic techniques accepted in the literature and it is basically restricted with HH applications. However, multiple alteration operators may be utilized in all meta-heuristic algorithms. Multiple mutation applications on meta-heuristic techniques had been investigated before on single objective optimization problems and called as dynamic mutation [**7**]. Hong had applied dynamic mutation on GA in their study.

To the best of our knowledge, there has been no investigation about applying multiple heuristics dynamically on the multi-objective problems in the literature. Based on this idea, we have examined and discussed the effect of handling multiple mutation operators or handling multiple heuristic operators on popular multi-objective evolutionary algorithms. In this part of the study, we introduce dynamic neighbor generation variations of NSGA, NSGA-II and MMBO to the literature. Those are non-dominated sorting genetic algorithm with dynamic neighbor generation (NSGA-DNG), non-dominated sorting genetic algorithm ii with dynamic neighbor generation (NSGA-II-DNG) and multi-objective migrating birds optimization algorithm with dynamic neighbor generation (MMBO-DNG). We aimed to show the effect of applying various mutation operators instead of a single mutation operator for GA-based approaches, and to show the effect of applying various heuristic operators for MMBO. Four problem specific heuristic operators are defined and those heuristics are applied as mutation operators in GA based techniques (see Section 4.3.5.2). Therefore, mutation operator and heuristic operator definitions have been used to name the same concepts. The terms solution and individual are used interchangeably throughout this chapter. Similarly, the terms population and generation refer to the same concept, are used interchangeably throughout the chapter.

4.3.1 Dynamic neighbour generation framework

Multi-objective optimization problems bring distinct difficulties together, due to the diversification in each objective and constraint. MOP needs several instruments for exploring the solution space. Therefore, it may be pointed out that multi-heuristic approaches are necessary for gaining promising results on MOP. Dynamic neighbor generation model is offered based on this idea. This model is capable of improving solutions by searching solution space in different directions. Three well-known algorithms are integrated to dynamic neighbor generation model. The dynamic neighbor generation model, illustrated in Figure 4.6 , is designed to be used at the stage in which solutions are improved. The working mechanism of the relevant algorithm remains exactly the same

except for the solution improvement part. The offered framework can be easily extended to integrate into other MOP algorithms. The framework we provide to the community is easily extensible and the source code is accessible.

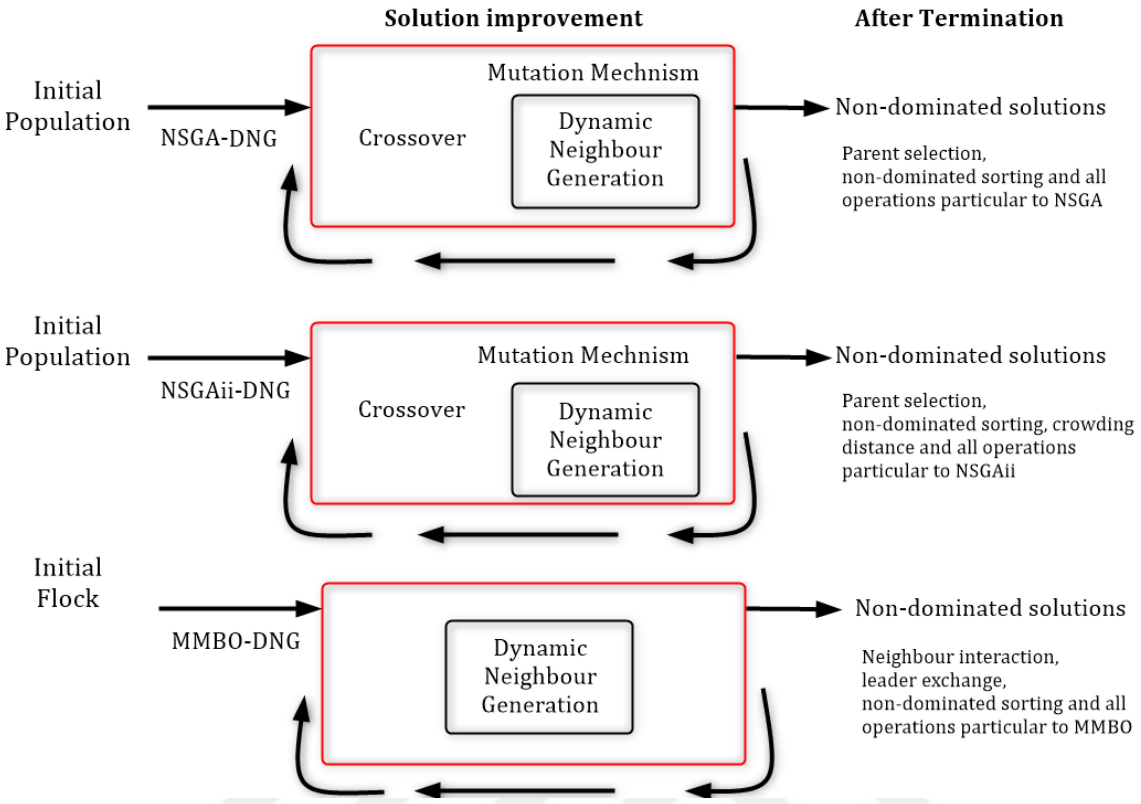

Figure 4.6. Dynamic neighbour generation model flowchart of NSGA-DNG, NSGA-II-DNG and MMBO-DNG.

Dynamic neighbour generation (DNG) is a framework that offers a new perspective to neighbour generation process. Many heuristic based solution techniques including evolutionary algorithms, swarm based techniques and local search algorithms use a single neighboring method for improving the solutions. Utilizing multiple neighboring operators may have pearls and pitfalls. It increases the search ability but can also deviate from the optimal set. However, considering the challenging solution space of the MOPs due to their often non-convex nature, searching the solution space effectively is definitely very critical.

DNG exploits more than one heuristic technique which are capable of searching different areas in the solution space. The flow chart of DNG is given in Figure 4.7. A permutation of heuristic operators are predetermined and applied to the solutions in order. The input of DNG is a solution or an individual. The selected heuristic from a heuristic pool is applied to the solution and the neighbouring solution is produced as an output. DNG mechanism is directly placed as the improvement heuristic of heuristic based techniques or is utilized as the mutation mechanism in evolutionary techniques. Any number of mutation operators can be defined for this mechanism.

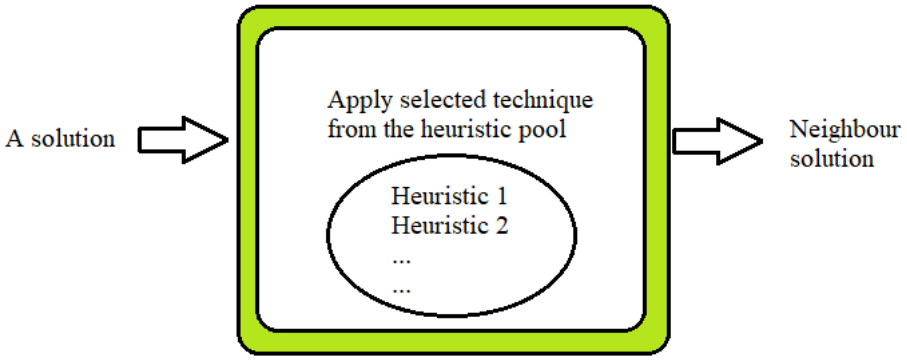

Figure 4.7. Dynamic neighbour generation working mechanism illustration.

4.3.2 Non-dominated sorting genetic algorithm with dynamic neighbor generation

As mentioned above applying multiple mutations may be promising for multi-objective optimization problems. Based on this idea we are offering a new NSGA variant and call it as non-dominated sorting genetic algorithm with dynamic neighbor generation (NSGA-DNG).

The pseudo code of NSGA-DNG is given in Figure 4.8. The basic difference of NSGA and NSGA-DNG is selecting and applying multiple mutations operators. A permutation of all existing mutation operators is constructed at the beginning. Mutation operators are applied to each offspring sequentially with mutation probability *m*.

```
Initialize parent population size N
Generate a permutation R of heuristics in the system
Evaluate the fitness of parent population
    while (not termination-condition) do
    for(i=0;i<N/2;i++) do
        Select two parents
        Generate two offspring by crossover with probability c
        Select the next mutation Rm operator in permutation R
        Perform mutation with probability m
        Add offsprings to population and expand the population
    end for
    Apply non-dominated sorting on expanded population size 2N
    Update population by first N individuals from the sorted generation list
end while
Output the non-dominated solutions
```

Figure 4.8. The pseudo code of NSGA-DNG.

4.3.3 Fast and elitist non-dominated sorting genetic algorithm with dynamic neighbor generation

Since it has been proposed, NSGA-II has been applied to MOPs in several areas such as resource scheduling [**106**], multi objective task scheduling [**107**] and multi-objective parallel flow shop scheduling [**108**]. Many modified versions of NSGA-II has been introduced to the literature up to now [**109**]-[**111**]. The working mechanism of NSGA-II-DNG is clarified in this section.

Fast and elitist non-dominated sorting genetic algorithm (NSGA-II) [**66**] comprises typical evolutionary mechanisms including crossover, mutation, parent selection, and child selection. The distinguishing feature of NSGA-II is the crowding distance mechanism which for transferring the individuals into the next generation.

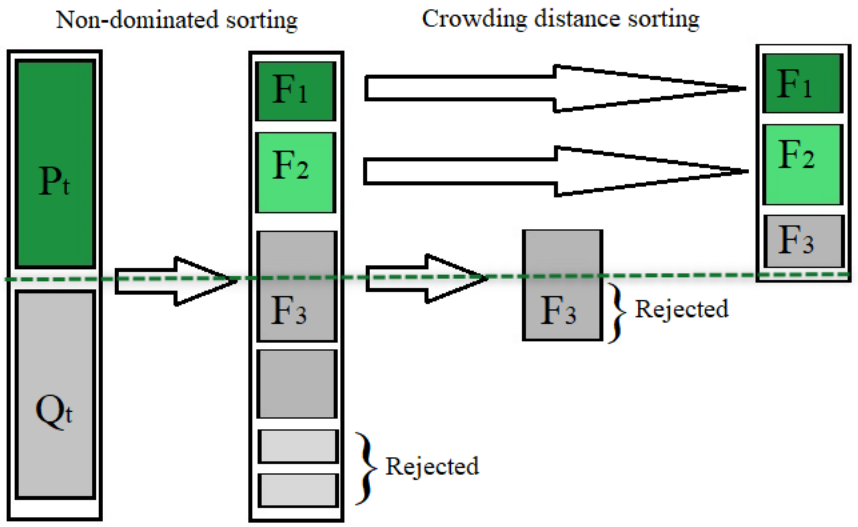

Figure 4.9. NSGA-II crowding distance mechanism.

Crowding distance mechanism allows selecting the best performing and the most diversified children and provides diversity among the generations. The mechanism is illustrated by a flowchart in Figure 4.9. The population size N is equal to P_t where also $P_t = Q_t$. The initial parent population P_t is doubled by crossover and mutation operators and child population with size Q_t emerges. The expanded population with size P_t+Q_t is formed by the union of P_t and Q_t. It is initially sorted using non-dominated sorting. The best individual produced after non-dominated sorting is most probably a non-dominated solution. This is followed by the solutions dominated once, twice and so on. Virtual clusters are defined from the non-dominated sorted list. There are totally F_x clusters including (F_1, F_2,.. F_x). In the illustration, the first cluster F_1 is directly transferred to child

population because its size does not overrun the population size N, F_2 transferred to the child population since the total transferred individuals including F_2 do not exceed N. Cluster F_3 exceeds the population size, hence crowding distance is applied on all individuals in F_3. The distance of each member to each other member in F_3 is computed and F_3 cluster is arranged according to this crowding distance scores. The best individual with respect to the crowding distance is the farthermost one from the others in the cluster. Hereby, the solutions in F_3 are sorted from the furthest one to the closest one. Solutions in F_3 cluster are transferred into the next generation one by one in order, until the population limit P_t is reached. Thence, crowding distance contributes variety prioritizing to the most discrete solution among candidates.

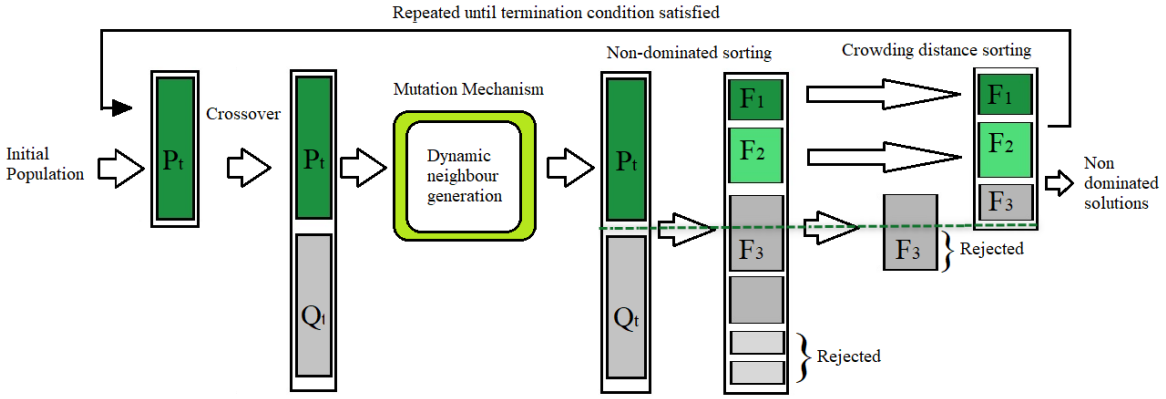

Figure 4.10. The flow chart of NSGA-II-DNG.

```
Initialize parent population size N
Initialize crossover probability c and mutation probability m
Generate a permutation of mutation operators R
Evaluate the fitness of parent population
while (not termination condition) do
    for(i=0;i<N/2;i++) do
        Select two parents
        Generate two offspring by crossover with probability c
        Select the next mutation operator from R
        Perform selected mutation operator with probability m
        Add offsprings to population and expand the population
    end for
    Apply non-dominated sorting on expanded population and determine clusters F(F1..Fx)
    Transfers cluster member to Pt+1 until they exceed population size N
    if (adding Fk exceeds N)
        Compute Crowding Distance of Fk
    Transfer members of Fk to Pt+1 until population size N is reached
end while
```

Figure 4.11. The pseudo code of NSGA-II-DNG.

The flow chart of NSGA-II-DNG is given in Figure 4.10. The pseudo code of the algorithm is presented in Figure 4.11. An initial population is generated in a predetermined

size *N* at the beginning and the population size is doubled with the particular crossover technique applied by the crossover probability and the DNG applied according to the mutation probability. At the end of the evolution process, the population size reaches to *2N*, and the best *N* individuals are elected and transferred to the next generation by NSGA-II rules.

4.3.4 Multi objective migrating birds optimization with dynamic neighbor generation

Recently proposed Migrating Birds Optimization (MBO) [**45**] is inspired by the V flight formation of migrating birds. A multi-objective version of migrating birds optimization (MMBO) algorithm is introduced by Zhang in 2019 [**97**]. MMBO has been successfully applied to various problems, such as multi-objective task allocation problem [**86**], flow shop scheduling problem [**95**], land distributing problem [**96**], assembly line balancing problem [**112**], [**113**].

The main idea behind MBO is the benefit mechanism between the individuals adjusted in a V-shaped population.

The four fundamental stages of the algorithm are initialization, leader evolution, follower's evolution and leader change. In initialization phase parameters are appointed and first population is produced. The leader is improved by appraising its *k* neighbor solutions in its neighborhood in second phase. Followers are evaluated in the third phase. Both split sides of the leader is developed by evaluating both their own neighbors generated in the neighborhood and the neighbors received from the solution in front of it. The leader solution is renovated in the last phase. The leader is transferred to the end of the left and right lines in order, then the best solution in the corresponding line is forwarded to become the leader.

The flow chart is given in Figure 4.12 and the pseudo code of the algorithm is given in Figure 4.13. The flock is constructed initially and then neighboring solutions for the leader and neighboring solutions for the other birds are created by exploiting the DNG mechanism. A permutation of the predefined heuristic operators is determined and these operators are sequentially applied according to the permutation for improving each particle. Thereafter, the neighboring solutions are shared with the other birds. All particles are ranked according to the non-dominated sorting technique. The first *N* solutions are selected for the next generation and they are placed on the hypothetical V-shaped population sequentially. This process is repeated until the termination condition is reached.

The output of the process is a set of non-dominated solutions in accordance with the nature of multi objective optimization problems.

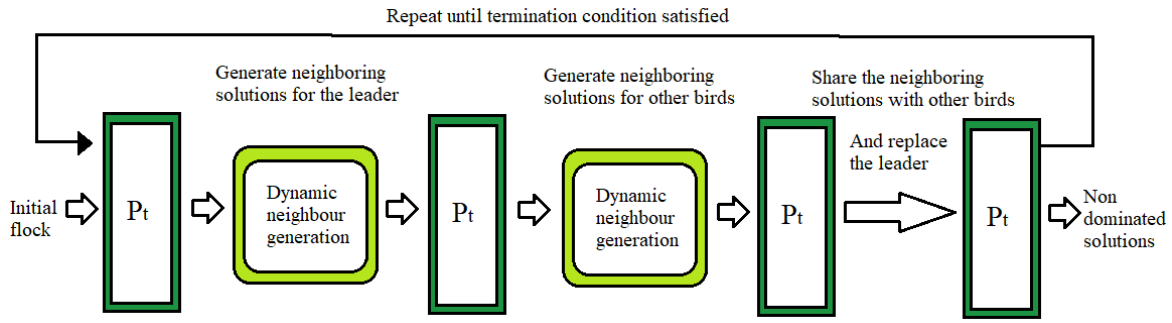

Figure 4.12. The working mechanism of MMBO-DNG.

Initialize the parameters number of birds (*nob*) as *N*
Initialize the flock
Generate a permutation *R* of heuristics in the system
while(not termination condition)**do**
    for *i=1* to *t* **do**
        Select the next heuristic technique *Rh* in permutation *R*
        Evolve the leader applying the next heuristic in *Rh*
        Evolve the followers in the left line applying the next heuristic in *Rh*
        Evolve the followers in the right line applying the next heuristic in *Rh*
        Rank all solutions according to the non-dominated sorting technique. And the first *N* solutions are selected for the next tour and they are placed on the hypothetical V-shaped population sequentially.
    end for
    Change the leader
end while

Figure 4.13. The pseudo code of MMBO-DNG.

4.3.5 Common parts in implementation

There are some common operations handled for all algorithms; such as population initialization method, heuristics and heuristic usage order adjustment. Heuristics are applied as mutation operators for genetic algorithm variants. Population initialization and heuristics' working mechanisms and one-point crossover recombination technique are described in this subsection exhaustively.

4.3.5.1 Population initialization

A sample individual that operates totally for four vehicles is illustrated in Figure 4.14. Vehicles belong to 26 shift type and 200 employees from e1 to e200. Each vehicle is assigned to a shift type and is responsible for the transportation of the employees in that specific shift type. Depending on the number of people assigned to the shifts, there may be more than one vehicle serving for a single shift. Each vehicle has a route starting from the depot and ending at the depot. To explain through the figure, vehicle one's shift type is

five, its route starts from the depot and its first stop is e190, then it continues to its route with e2, e25 and so on. Finally, the route ends at the depot. Vehicle one and vehicle two provide transportation of employees those are assigned to shift id five. Whereas vehicle three belongs to shift type 26 and vehicle four belongs to shift type two.

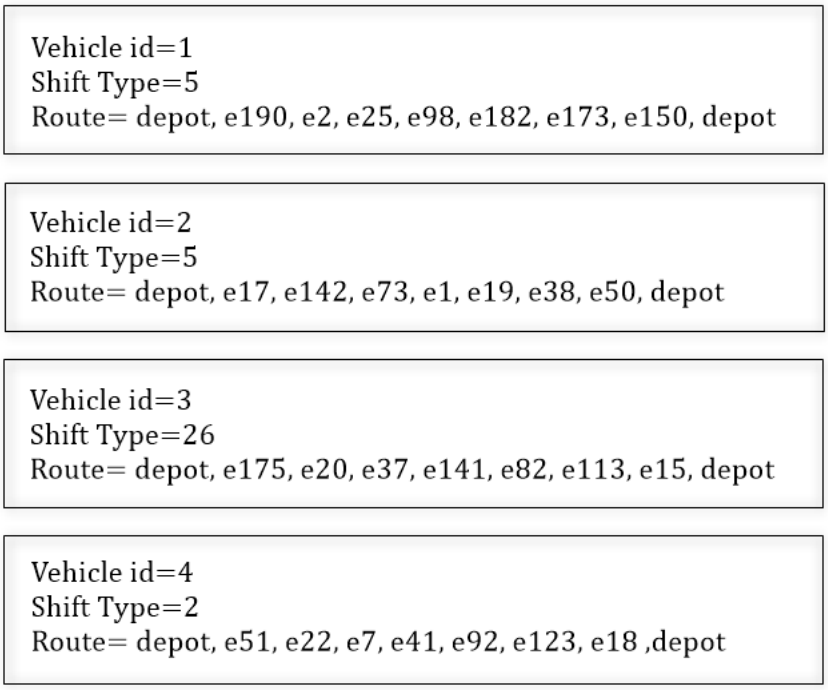

Figure 4.14. A sample individual operating four vehicles.

Population is a community that consists of a set of individuals. The initial population generation mechanism is the same for all encoded algorithms in this chapter. A simple individual is composed as follows: the number of employees assigned to each shift is adjusted considering the requirements in each time slot. All shifts and their assignment numbers are matched randomly. In the following process employees are assigned to vehicles and routes are constructed randomly. Each vehicle naturally provides transportation of one shift at a time, so that each vehicle belongs to a shift type. All employees assigned to that specific vehicle belong to one shift. Thereby a feasible initial solution is composed that meets the workforce requirements.

4.3.5.2 Problem specific heuristic algorithms

Four heuristic algorithms used to evolve solutions are described here. The first two of them make modifications on shift assignments and remaining two make modifications on vehicle assignments. Illustrative figures are presented to clarify working mechanisms of algorithms.

Heuristic 1 - Shift assignment modify shifts: A new random shift type is selected among the ones that has not been selected or assigned to any employees before. Another random

existing shift is selected among the ones those are already assigned to employees. The employees in existing shift are transferred to the new one. An example solution formed by totally two vehicles and the updated solution after applying heuristic one is given in Figure 4.15. The employees in vehicle one was initially assigned to shift id five. Heuristic one updated shift designation, by a random shift (shift id eight) that has not been assigned to any employees before. The updated fragments are emphasized with red and bold font.

Individual											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	E74	E42	E7	D0	D0	E86	E101	E196	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Individual after applying heuristic 1											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	E74	E42	E7	D0	D0	E86	E101	E196	D0
	Vehicle=1, Shift Id=8						**Vehicle=2, Shift Id=21**				

Figure 4.15. Individual illustration before and after applying heuristic 1.

Heuristic 2 - Shift assignment random exchange: An employee that is already assigned to one of the shifts is replaced by another randomly selected one that has not been appointed to any shift before. Working mechanism of heuristic two is represented in Figure 4.16. E86 is selected from the initial individual and replaced by a random employee E29 that has not been assigned to any shift or vehicle before.

Individual											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	E74	E42	E7	D0	D0	**E86**	E101	E196	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Individual after applying heuristic 2											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	E74	E42	E7	D0	D0	**E29**	E101	E196	D0
	Vehicle=1, Shift Id=8						**Vehicle=2, Shift Id=21**				

Figure 4.16. Individual before and after applying heuristic 2.

Heuristic 3 - Random inter vehicle exchange: Two employees are randomly selected those are already assigned to different vehicles and shift. Those two employees are simply swapped and pass interchangeably. The individual before applying heuristic three and after applying heuristic three is illustrated in Figure 4.17. Two employees E74 and E196 belonging to different vehicles were randomly selected and replaced.

Individual											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	**E74**	E42	E7	D0	D0	E86	E101	**E196**	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Individual after applying heuristic 3											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	**E196**	E42	E7	D0	D0	E86	E101	**E74**	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Figure 4.17. Individual before and after applying heuristic 3.

Heuristic 4 - In vehicle swap: A random vehicle is selected and two points indicating employees on vehicle's route is simply swapped. An example illustration for heuristic four is given in Figure 4.18. A random vehicle, vehicle id one is selected and two employees E102 and E42 are exchanged within selected single vehicle.

Individual											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	**E102**	E74	**E42**	E7	D0	D0	E86	E101	E196	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Individual after applying heuristic 4											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	**E42**	E74	**E102**	E7	D0	D0	E86	E101	E196	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Figure 4.18. Individual before and after applying heuristic 4.

4.3.5.3 One point crossover

One-point crossover is applied while implementing genetic algorithm based techniques. A simplified example is given in Figure 4.19 to explain one point crossover applied. Parent one, parent two, offspring one and offspring two are illustrated. Each individual's length in this example is equal to eleven starting from index zero to index ten. Employees are indicted as Ex and vary from E1 to E200. Depot node is node "D0", each route starts and ends at the depot. Each vehicle belong to a shift id varies from one to 26. Parent one operates two vehicles, first vehicle belongs to shift type five and its route starts in depot D0, then E102, E74, E42, E7 are visited and the route ends at the depot. The second vehicle in parent one is assigned to shift type 21. Second vehicle's route starts from the depot, it visits E86, E101, E196 then returns back to depot.

Parent 1											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	E74	E42	E7	D0	D0	E86	E101	E196	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=21**				

Parent 2											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E95	E101	E52	E86	D0	D0	E103	E75	E43	D0
	Vehicle=1, Shift Id=16						**Vehicle=2, Shift Id=5**				

Offspring 1											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E102	E74	E52	E86	D0	D0	E103	E75	E43	D0
	Vehicle=1, Shift Id=5						**Vehicle=2, Shift Id=16**				

Offspring 2											
Index->	0	1	2	3	4	5	6	7	8	9	10
Route->	D0	E95	E101	E42	E7	E86	D0	D0	E196	E102	D0
	Vehicle=1, Shift Id=16							**Vehicle=2, Shift Id=21**			

Figure 4.19. Sample one point crossover illustration.

One-point crossover selects a random cut point and copies the part before cut point from first parent to the offspring and the rest is completed by second parent starting from the cut point. Random cut point is two for this illustration and indicated by a red line. When we analyze offspring one and offspring two, nodes from index zero to two are taken from parent one and parent two respectively including the shift id information. The rest is taken from the other parent starting from cut point by caring the vehicle capacities and avoiding the repetitions. If a node exceeds vehicle capacity; it is added to a new vehicle. If the offspring individuals supply requirements in the affected time slots then they are admitted.

4.4 Experimental Setup

In this section, performance measures, parameter fine tuning experiments, experimental setup and essential implementation details are provided and discussed. VRPSA is verified on Gurobi solver with Python and the algorithm implementations are done using Java programming language on Eclipse IDE running on Intel i7-5500U CPU, and 16 GB memory computer operated by Windows 10 operating system. We conducted our experiments on three real datasets.

4.4.1 Performance metrics

In multi-objective optimization problems, the solution consists of a solution set instead of a single value. Numerous methods have been proposed to measure performance in multi-objective optimization problems to measure the quality of the solution set. [**114**] present a review of 54 multi-objective optimization metrics in the literature and indicate in their

study that, the hypervolume is the most used metric, followed by the generational distance, the epsilon indicator and the inverted generational distance. In our experiments, two metrics are used systematically to measure both the diversity and convergence, those are hypervolume indicator and inverted generational distance.

4.4.1.1 Hypervolume

The hypervolume (HV) [**115**] is a widely accepted metric that measures both the diversity and the convergence of a number of non-dominated solutions in the objective space also known as s-metric or space coverage. A reference point is used to calculate the mentioned covered area that is worse than all discovered non dominated solutions. The volume of the shape composed by non-dominated solutions and the reference point corresponds to the hypervolume value. Figure 4.20 is a sample illustration for two objective minimization problem. *f1* and *f2* are the objective functions. Black points indicate the non-dominated solutions and the red reference point is worse than all non-dominated solutions. The area of the region between non-dominated points and the reference point gives us the hypervolume value. The larger the hypervolume value, the better the algorithm is.

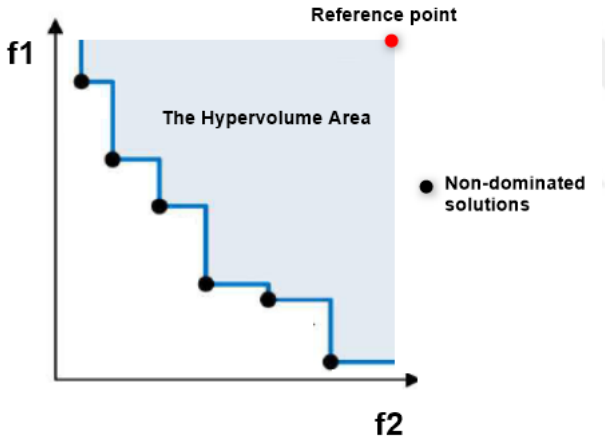

Figure 4.20. Hypervolume calculation illustration.

4.4.1.2 Inverted generational distance

Inverted generational distance (IGD) is also a well-known indicator proven its success on measuring performance of multi-objective optimization problems.

A randomly generated and uniformly distributed Pareto optimal front (POF) is required for calculating IGD. Let P^* be a set of uniformly distributed points in POF. The size of P^* is $|P^*|$. Each member in POF must dominate all members in solution space found by the algorithms. Let A be sample approximation set and assume we try to measure the IGD value of *A (IGD(P*,A))*. The minimum Euclidean distance between each point *v* in P^* and all points in *A* are computed *(d(v,A))* and summed up. This total is divided by the size of

P^*, hence *IGD* is calculated. *IGD* is formally defined in equation (4.12):

$$IGD(P^*, A) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \qquad (4.12)$$

Figure 4.21 is a sample illustration for two objective minimization problem. *f1* and *f2* are the objective functions. Black points indicate the non dominates solutions in set *A*, red points indicate the non dominated solutions in set *B*, and the black hollow circles indicate Pareto optimal front points or reference points. IGD arises the average minimum distance of set *A* to each reference point and arises the average minimum distance of set *B* to reference points. The algorithm with less average distance to the reference set is considered as more successful.

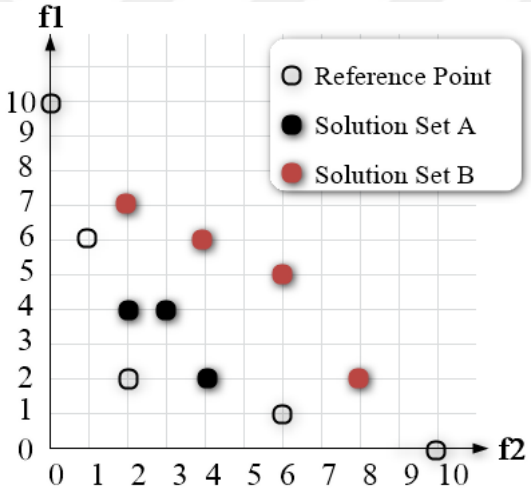

Figure 4.21. Sample two dimensional solution set including Pareto fronts.

If the size of POF is large enough, IGD could measure both the diversity and convergence in a sense. In this work size of P^* or the number of points in POF is determined as 500 as done in [**64**].

4.4.2 Parameter fine tuning

Four heuristics are defined for improving VRPSA, the detailed information about those algorithms are given in Section 4.3.5.2. NSGA, NSGA-II enforce a single mutation technique and MMBO performs a unique heuristic technique for improving solutions. Since we have four different heuristic techniques, we have conducted an experiment to understand which of the four operators in the system is compatible with which algorithm. In order to assess a fair comparison among all applied techniques, we set the stopping

criterion of the algorithms based on the number of instances (*noi*) created while each algorithm is running.

This experiment is done on each of three datasets, while number of instances (*noi*) is set to 10,000 and each case is repeated 50 times. Average objective results are given in the Table 4.4. H1 through H4 correspond to heuristic one through four, respectively. Considering that both routing and scheduling objective values are aimed to minimize, H1 outcomes are approximating the global optimum. NSGA and NSGA-II give the best results for both objectives when H1 is applied as improvement technique. MMBO gives the best scheduling results in scheduling objective with H1 and gives the best routing results in scheduling objective with H3 and followed closely by H1. Considering these results, it was decided to use H1 in the implementation of the original NSGA, NSGA-II and MMBO algorithms.

Table 4.4. Parameter fine tuning experiment results applied neighbour generation techniques.

Algorithm	**Neighbour generation techniques (ordered from best to worst)**
NSGA	**H1**, H4, H2, H3
NSGA-II	**H1**, H2, H4, H3
MMBO	**H1**, H3, H2, H4

In order to get the best performance from the algorithms, we need to use their best performing values. These best performing values are discovered with fine tuning experiments. After defining best parameter values, in the extensive experiments, the number of solutions that algorithm can generate while surfing in the solution space is limited to 50,000 for each algorithm. The experiments are repeated ten times and results are acquired from the average performances in non-dominated solution sets of these ten runs. Parameter fine tuning results are represented in Table 4.5.

Table 4.5. Parameter fine tuning results.

Parameter	**NSGA**	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**Parameter**	**MMBO**	**MMBO-DNG**
pSize	**100**, 150, 200	**100**, 150, 200	**100**, 150, 200	**100**, 150, 200	**nob**	5, 21, **51**	5, 21, **51**
c	**0.5**, 0.7, 0.9	0.5, 0.7, **0.9**	0.5, 0.7, **0.9**	**0.5**, 0.7, 0.9	**non**	**3**,5,7	3,**5**,7
m	**0.1**, 0.3, 0.5	0.1, 0.3, **0.5**	0.1, 0.3, **0.5**	**0.1**, 0.3, 0.5	**nof**	5, **10**	**5**, 10
					olf	**1**,2,3	**1**,2,3

4.5 Experimental Results Discussion

In this section, experimental results are presented with their discussions. The parameter values of each algorithm are adjusted according to the parameter fine tuning results. The reader should remember that for the sake of obtaining a fair comparison, the stopping criterion is the number of instances/solutions (*noi*) created by an algorithm. The performances of the algorithms are compared under several different stopping criterion values. Firstly, we made tests where the algorithms stop after a small *noi* value (i.e. noi=50,000) to discuss the immediate performance of algorithms. Then we had increased the *noi* value to 100,000 and 200,000 to uncover long term performances. All test results in this section are tabulated in a similar format.

The experiments are conducted on three real datasets twenty times. HV and IGD indicators are computed by using the non-dominated solution sets. In addition to the mean values, standard deviation of HV and IGD indicators are also presented in all tables to demonstrate the robustness of the algorithms.

4.5.1 Results when noi=50,000

The mean and standard deviation of HV performances obtained when *noi* is equal to 50,000 on three datasets by each algorithm is given in the Table 4.6. Larger HV value means that results indicate higher coverage ability or in other words outcomes are approximating to the optimal results more than their competitors. Detailed information about the indicators is presented in Section 4.4.1. When *noi* is equal to 50,000 NSGA-II is better than other algorithms on the mean in terms of HV. More specifically, NSGA is better than NSGA-DNG, NSGA-II is better than NSGA-II-DNG and MMBO is better than MMBO-DNG according to mean HV values.

The mean and standard deviation IGD obtained on each dataset by each algorithm when noi is equal to 50,000 is also given in Table 4.6. We should remind that, smaller IGD value demonstrates results those are closer to Pareto optimal front, therefore a more thriving performance. NSGA-II-DNG is better to approximate to the Pareto optimal front than its competitors, it exploits the best mean results in terms of IGD for all datasets on the mean.

Table 4.6. Mean and standart deviation of HV and IGD values while noi=50,000.

	HV						IGD					
	NSGA	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**MMBO**	**MMBO-DNG**	**NSGA**	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**MMBO**	**MMBO-DNG**
Dataset 1												
Avg	1.36E+08	1.31E+08	**1.38E+08**	1.29E+08	1.32E+08	1.32E+08	5,946	4,762	6,095	**3,066**	5,383	5,348
StdDev	4.65E+06	2.40E+06	5.40E+06	2.59E+06	2.12E+06	1.82E+06	1,699	1,029	1,385	168	754	524
Dataset 2												
Avg	1.36E+08	1.32E+08	**1.40E+08**	1.31E+08	1.33E+08	1.34E+08	5,722	5,113	5,943	**2,906**	5,252	5,004
StdDev	3.81E+06	2.49E+06	3.61E+06	8.89E+05	1.50E+06	1.42E+06	1,474	1,228	1,303	202	502	516
Dataset 3												
Avg	8.83E+07	7.51E+07	**1.10E+08**	6.31E+07	8.51E+07	8.51E+07	15,907	12,624	14,540	**7,568**	11,895	11,849
StdDev	1.20E+05	2.05E+05	3.61E+05	2.01E+05	4.67E+05	6.84E+05	3,486	3,389	3,682	243	1,739	1,167

The results gained by *noi* is equal to 50,000 brings out that, NSGA-II outperforms other rivals in terms of HV on the mean and NSGA-II-DNG outperforms other rivals in terms of IGD on mean. NSGA-II is better on coverage but NSGA-II-DNG is more successful on finding results those are closer to Pareto optimal front for restricted small number of instances. HV and IGD indicators measure the performance from different point of views. Obtaining better figures at least in one metric, gave us clues about a promising performance of our algorithms and therefore motivated us to go ahead. Besides, proposed algorithms apparently appear more robust than the original versions of the algorithms.

Table 4.7. Mean and standart deviation of HV and IGD values while noi=100,000.

	HV						IGD					
	NSGA	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**MMBO**	**MMBO-DNG**	**NSGA**	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**MMBO**	**MMBO-DNG**
Dataset 1												
Avg	1.23E+08	1.26E+08	1.28E+08	1.28E+08	1.30E+08	**1.32E+08**	6,143	6,144	6,342	6,076	6,283	**4,578**
StdDev	1.79E+06	4.49E+06	3.11E+06	1.69E+06	1.72E+06	2.41E+06	562	794	485	857	658	460
Dataset 2												
Avg	1.26E+08	1.30E+08	1.30E+08	1.31E+08	1.32E+08	**1.33E+08**	5,809	5,612	5,979	5,657	6,039	**4,431**
StdDev	3.01E+06	3.96E+06	1.14E+06	2.77E+06	1.20E+06	1.53E+06	422	714	397	965	485	585
Dataset 3												
Avg	4.66E+07	6.00E+07	6.01E+07	6.82E+07	6.73E+07	**8.66E+07**	15,254	14,455	15,807	15,526	15,794	**10,087**
StdDev	1.84E+05	1.38E+05	4.66E+05	1.41E+05	2.00E+05	5.41E+05	591	2,091	2,523	615	772	920

4.5.2 Results when noi=100,000

The HV and IGD results gained when *noi* is equal to 100,000 are given in the Table 4.7. MMBO-DNG has achieved the most valuable HV and IGD results for all datasets on the mean. Moreover dynamic neighbor generation versions of algorithms beat their standard versions according to both metrics. As a result of this experiment, it has been observed

that the achievement of the new algorithms is promising and this experiment encouraged us to the subsequent step.

4.5.3 Results when noi=200,000

Table 4.8. Mean and standart deviation of HV and IGD values while noi=200,000.

	HV						IGD					
	NSGA	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**MMBO**	**MMBO-DNG**	**NSGA**	**NSGA-DNG**	**NSGA-II**	**NSGA-II-DNG**	**MMBO**	**MMBO-DNG**
Dataset 1												
Avg	1.29E+08	1.32E+08	1.32E+08	1.33E+08	1.35E+08	**1.40E+08**	6,489	5,977	5,173	5,109	5,364	**2,937**
StdDev	5.54E+06	2.07E+06	4.31E+06	2.36E+06	2.17E+06	9.09E+05	1,630	1,393	710	1040	734	195
Dataset 2												
Avg	1.31E+08	1.32E+08	1.33E+08	1.34E+08	1.36E+08	**1.38E+08**	6,241	5,782	4,981	4,928	5,140	**3,943**
StdDev	3.15E+06	5.22E+06	1.49E+06	2.36E+06	2.11E+06	3.34E+06	896	848	1,854	1558	595	824
Dataset 3												
Avg	5.92E+07	6.91E+07	8.51E+07	8.78E+07	8.05E+07	**9.83E+07**	16,637	15,822	11,730	14,737	11,176	**10,143**
StdDev	1.85E+05	1.31E+05	6.83E+05	6.25E+05	9.15E+05	2.39E+05	3,483	2,026	1,524	2308	1,690	5,668

The HV and IGD results obtained when *noi* is equal to 200,000 are given in Table 4.8. MMBO-DNG has achieved the most precious HV results for all datasets on the mean than the other algorithms. The hopeful performance of MMBO-DNG becomes more apparent with this experiment. Furthermore, dynamic neighbor generation versions of each algorithm beat its original counterparts. That is, in terms of both metrics, NSGA-DNG produced better result than NSGA, NSGA-II-DNG produced better result than NSGA-II and MMBO-DNG produced better results than MMBO. Dynamic neighbor generation model enhances the algorithm performances of regular algorithms on covering solution space and approximating the Pareto optimal front.

Box plots given in Figure 4.22 and Figure 4.23 illustrate HV and IGD values, respectively, for all datasets when *noi*=200,000. It is clearly seen in the charts that; MMBO-DNG figures out higher (better) HV results and lower (again better) IGD results than others as anticipated. Furthermore dynamic neighbor generation based models are more successful than their original versions. When NSGA and NSGA-II based techniques are compared; NSGA-II and NSGA-II-DNG mostly produces higher HV values than NSGA and NSGA-DNG. The reason of this condition is the crowding distance property of NSGA-II.

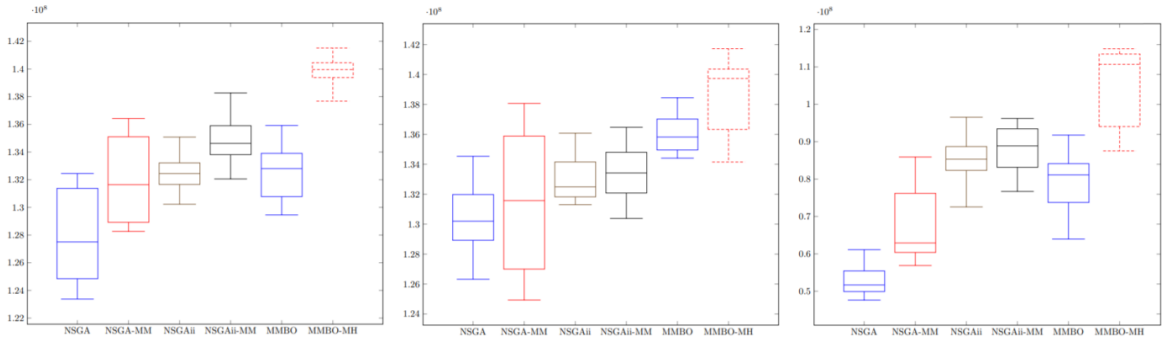

Figure 4.22. Box plot for HV values of dataset 1, 2, 3 respectively, when noi=200,000.

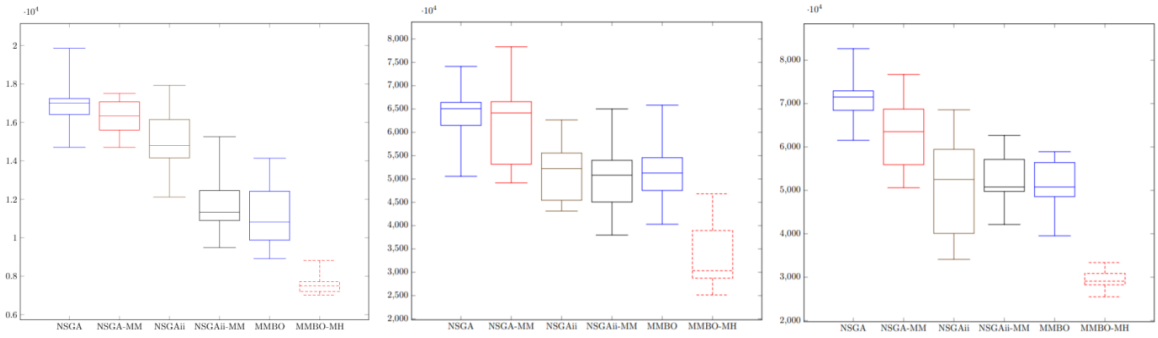

Figure 4.23. Box plot for IGD values of dataset 1, 2, 3 respectively, when noi=200,000.

MMBO is a promising algorithm that has been popular in recent years. MMBO-DNG, which is applied for the first time, provided favorable results, indeed. The experiments uncover that MMBO-DNG definitely offers better solutions in large problem instances, which is a desired property and therefore fills a gap in the literature. We believe that this good performance is due to integrating the exploitation capability of applying dynamic neighbor generation with MBO's exploration capability. MMBO-DNG is a novel but promising computational intelligence technique especially for large instances of scheduling problems. Besides this, the achievements of dynamic neighbor generation based techniques are also apparent. NSGA-DNG produces better outcomes than NSGA, NSGA-II-DNG generates preferable results than NSGA-II and MMBO-DNG produces more expedient results than MMBO in terms of HV and IGD metrics for especially large number of instances. Consequently, the success of dynamic neighbor generation framework clearly appears.

Table 4.9. Results overview of the ANOVA test considering HV and IGD indicators.

			Sum of Squares	df	Mean Square	F	Sig.
HV	**Dataset1**	Between Groups	1.00E+19	5	3.00E+18	24,924	<.001
		Within Groups	1.00E+19	114	1.00E+17		
		Total	3.00E+19	119			
	Dataset2	Between Groups	7.00E+18	5	1.00E+18	14,358	<.001
		Within Groups	1.00E+19	114	1.00E+17		
		Total	2.00E+19	119			
	Dataset3	Between Groups	2.00E+20	5	4.00E+19	18,576	<.001
		Within Groups	2.00E+20	114	2.00E+18		
		Total	4.00E+20	119			
IGD	**Dataset1**	Between Groups	2.00E+20	5	3.00E+19	11,615	<.001
		Within Groups	3.00E+20	114	3.00E+18		
		Total	5.00E+20	119			
	Dataset2	Between Groups	5.00E+19	5	1.00E+19	3,202	<.001
		Within Groups	4.00E+20	114	3.00E+18		
		Total	4.00E+20	119			
	Dataset3	Between Groups	2.00E+20	5	4.00E+19	6,655	<.001
		Within Groups	7.00E+20	114	6.00E+18		
		Total	9.00E+20	119			

Considering the standard deviation values obtained, it is observed that the implementations are sufficiently robust. We have also applied the one way analysis of variance (ANOVA) test at an $\alpha \leq 0.05$ significance level to check if the mean differences are statistically significant. The ANOVA results for each indicator (HV and IGD) while *noi* is equal to 200,000 are listed in Table 4.9. Additionally, we conducted Tukey's honestly significant difference (HSD) post-hoc test to analyze algorithms's pairwise differences. Results of Tukey HSD test for each dataset and each indicator is reported in Table 4.10. Pairwise mean differences are significant at the $\alpha \leq 0.05$ significance level for nearly every comparison. Results show that differences between the algorithms are statistically significant.

Table 4.10. Pair-wise comparison of algorithms for HV and IGD indicators due to Tukey HSD. (A-B) presents the mean differences between the algorithms at columns A and B. (*) indicates that the corresponding mean difference is significant at the $\alpha \leq 0.05$ significance level.

		HV			IGD		
		Dataset1	Dataset2	Dataset3	Dataset1	Dataset2	Dataset3
A	B	(A-B)	(A-B)	(A-B)	(A-B)	(A-B)	(A-B)
NSGA	**NSGA-DM**	**-3.75E+8***	**-4.70E+8***	**-1.64E+9***	**1.59E+9***	**2.23E+9***	**3.54E+9***
NSGA	**NSGAii**	**-4.95E+8***	**-3.55E+8***	**-1.78E+9***	**1.59E+9***	**2.33E+9***	**3.62E+9***
NSGA	**NSGAii-DM**	**-5.85E+8***	**-3.95E+8***	**-2.12E+9***	**1.81E+9***	**2.95E+9***	**3.23E+9***
NSGA	**MMBO**	**-3.40E+8***	-2.90E+08	**-2.34E+9***	**2.01E+9***	1.82E+09	1.51E+09
NSGA	**MMBO-DNG**	**-4.35E+8***	**-4.15E+8***	**-1.61E+9***	**1.76E+9***	**2.54E+9***	**3.30E+9***
NSGA-DM	**NSGAii**	**-7.70E+8***	**-3.15E+8***	**-2.93E+9***	**3.18E+9***	**2.05E+9***	**3.27E+9***
NSGA-DM	**NSGAii-DM**	**-3.10E+8***	**-7.50E+8***	**-1.82E+9***	**2.13E+9***	**2.19E+9***	**3.87E+9***
NSGA-DM	**MMBO**	2.50E+08	1.80E+08	**1.88E+9***	**-2.19E+9***	**-2.42E+9***	**-3.16E+9***
NSGA-DM	**MMBO-DNG**	**-4.00E+8***	**-3.55E+8***	**-1.60E+9***	**1.72E+9***	**2.58E+9***	**3.84E+9***
NSGAii	**NSGAii-DM**	**-1.08E+9***	**-6.50E+8***	**-3.91E+9***	**3.39E+9***	**2.14E+9***	**3.20E+9***
NSGAii	**MMBO**	**-7.95E+8***	**-5.45E+8***	**-2.05E+9***	**1.86E+9***	**2.31E+9***	**3.41E+9***
NSGAii	**MMBO-DNG**	**-7.30E+8***	**-4.70E+8***	**-1.82E+9***	**2.16E+9***	**2.77E+9***	**3.75E+9***
NSGAii-DM	**MMBO**	2.85E+08	3.05E+08	**2.86E+9***	**-2.41E+9***	1.77E+09	**3.29E+9***
NSGAii-DM	**MMBO-DNG**	**-3.50E+8***	**-5.80E+8***	**-2.58E+9***	**2.23E+9***	**2.59E+9***	**3.52E+9***
MMBO	**MMBO-DNG**	**-6.50E+8***	**-7.50E+8***	**-1.73E+9***	**1.88E+9***	**2.36E+9***	**3.64E+9***

4.6 Summary

In this chapter, we have defined a new problem called VRPSA and introduced it to the literature, which integrates the features of shift assignment problem and the capacitated vehicle routing problem. The problem has emerged from a set of realistic requirements of a company. We have presented the mathematical model of VRPSA and verified it using the Gurobi solver. We have applied the state of the art evolutionary algorithms and a popular swarm algorithm to solve the proposed problem; those are NSGA, NSGA-II and MMBO. In addition to all we have introduced, three novel algorithms based on dynamic neighbor generation model are also presented; those are NSGA-DNG, NSGA-II-DNG, MMBO-DNG.

In order to observe the performance of the implemented algorithms extensive computational experiments were carried out by setting noi (the stopping criterion) equal to varying values on three data sets provided by the collaborative. MMBO-DNG produced

the best mean experimental HV and IGD outcomes and dynamic neighbor generation versions of each algorithm beated its original version for larger noi values.

The outcomes uncover that MMBO-DNG definitely offers better solutions in large-sized problem instances, which is a desired property and gap in the literature. We believe that this good performance is due to integrating the exploitation capability of applying dynamic neighbor generation with MBO's exploration capability. It is shown that MMBO-DNG is a novel but promising computational intelligence technique especially for large-sized instances of scheduling problems.

5. DYNAMIC NEIGHBOUR GENERATION FRAMEWORK FOR MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

This chapter presents a comparative experimental study about the dynamic neighbour generation.

Even though multi-objective optimization problems' solution space is inherently complex, heuristic based algorithms often search the solution space by a single neighbour creation technique. Dynamic neighbour generation (DNG) allows searching solution space by multiple heuristic operators and brings a new perspective to neighbour creation process especially for the multi-objective optimization problems. This chapter presents extensive comparative experiments for the purpose of analyzing and revealing the achievement of our proposed DNG framework on a set of benchmark problems. DNG is integrated with the fast and elitist multi-objective genetic algorithm (NSGA-II), multi-objective migrating birds optimization algorithm (MMBO), the strength Pareto evolutionary algorithm 2 (SPEA2) and Pareto simulated annealing (PSA). Multi-objective hyper-heuristic evolutionary algorithm (MHypEA) is also implemented for obtaining a more effective comparison. Experiments demonstrate that DNG based versions of algorithms are better than their original variants. DNG based variants are 24.39% and 28.35% better on the average than their original variants in terms of inverted generational distance indicator and in terms of hypervolume indicator respectively. MMBO-DNG was the best performing algorithm among the algorithms participating to the experiment. We believe that DNG framework will be widely used in multi-objective optimization domain due to its easy applicability.

Heuristic based approaches are popular and useful choices on optimization domain which were initially developed for SOPs. Researchers have adapted reputed techniques to multi-objective optimization problems MOPs in course of time. Since MOPs have a set of results as their output instead of a single outcome, distinctive techniques are required for solving them. Besides, different performance metrics are inherently required for measuring the quality of the solutions. Thus, MOPs have become a separate research field with their own techniques, test problems and performance measures.

An MOP brings out a non-dominated solution set as the result of the optimization process.

A non-dominated solution, also known as Pareto optimal solution is not dominated by any other nominee in the solution space. Decision maker (DM) discovers the most advantageous solution within the non-dominated set. Optimization techniques applied on MOPs may be divided into four categories according to the DM strategy [**6**]. Those are no preference method, a priori method, a posteriori method and progressive method. DM does not exist in no preference method so that many assumptions must be generated. A priori strategy transforms the problem into a one-objective problem and solves it like a single objective optimization problem. Posteriori method finds a Pareto optimal set and the DM chooses the most beneficial one among them. In progressive methods, DM conducts the optimization process by giving preferences throughout the process. Posteriori methods include evolutionary algorithms(EAs) and swarm based approaches which are very popular in the operations research domain.

MOPs have a complex solution space due to the multiple objectives those must be improved in an accordance and many constraints that must be ensured. Applying multiple neighborhood search techniques with different characteristics, instead of solely one heuristic method, increase the exploration proficiency in the solution space. However, it is not a common practice to exploit multiple heuristics and its usage is limited with hyper heuristics (HH) [**116**]- [**118**].

Exploiting multiple heuristics for creating neighbour solutions is proposed in the previous chapter(called dynamic neighbor generation (DNG)) [**119**], but its performance on benchmark problems is not overseen. In this part of the study, we aim at making a comprehensive assessment of the performance of DNG framework on a set of multi-objective benchmark problems. In line with this purpose, we have extended a problem test suite [**120**]. MMBO, SPEA2 and PSA are implemented on the benchmark problems in the test suite for the first time. DNG version of NSGA-II, MMBO, SPEA2 and PSA are also adapted and implemented. Additionally, we have implemented multi-objective hyper-heuristic evolutionary algorithm (MHypEA) on the test suite. Since we presented an alternative framework (DNG) that exploits multiple neighboring operations, we wanted to add a known technique (MHypEA) that exploits many heuristics by its nature, to strengthen the comparison.

The contributions of this chapter is three-fold : (i) proposing the improved DNG versions of NSGA-II, MMBO, SPEA and PSA, (ii) implementing NSGA-II, MMBO, SPEA and PSA and their DNG versions to the test suite, implementing MHypEA to the test suite (iii) presenting an extensive discussion as a result of the experiments.

The remainder of the chapter is organized as follows: solution methodologies are described in Section 5.1. Experimental setup implicating performance metric and parameter fine tuning experiments are presented in Section 5.2. Experimental results together with their discussions are illustrated in Section 5.3. Finally, concluding remarks and suggestions for future research are presented in Section 5.4.

5.1 Solution Techniques

The solution techniques introduced are described in this section. Common parts in the implementation are also clarified including the mutation techniques and the crossover operator. It is worth to note that; the terms *solution* and *individual* refer to the same concept and are used interchangeably throughout the chapter.

5.1.1 Dynamic neighbour generation

In this chapter, we have integrated DNG framework with NSGA-II, MMBO, SPEA2 and PSA algorithms. All improved variants are described in the subsequent parts of this section. The fast and elitist non-dominated sorting genetic algorithm with DNG is detailly described in Section 4.3.3 and multi objective migrating birds optimization with DNG is explained in Section 4.3.4.

5.1.2 Strength Pareto evolutionary algorithm 2 with DNG

Strength Pareto evolutionary algorithm is one of the most popular algorithms in multi-objective optimization domain [**67**]. The most distinctive feature of this algorithm that distinguishes it from other evolutionary techniques is the fine grained fitness assignment procedure. It ranks the solutions according to the dominance and also according to the normalized euclidean distances among the solutions. For better explanation, the main parameters and concepts are listed as follows:

- N is the population size
- N' is the archive size
- T is the maximum generation number
- t is the current generation
- $t+1$ is the next generation
- $P(t)$ is the current population
- $P(t+1)$ is the population at time $t+1$
- $A(t)$ is the current archive
- $A(t+1)$ is the archive at time $t+1$

- *i* is an individual of the population
- *R(i)* is the raw fitness value of individual *i*
- *D(i)* stands for the normalized distance for individual *i*
- *F(i)* is the fitness of individual *i*

In this paragraph, the fine grained fitness assignment strategy is clarified. Fitness of a solution *i* is *F(i) = R(i) + D(i)*. *R(i)* stands for a raw fitness value as an integer that keeps the number of times the solution is dominated. *R(i)* is *zero* for a non-dominated solution. For calculating *D(i)*, firstly the distances of solution *i* to all other solutions are calculated and sorted from the smallest one to the highest one. Then *D(i)* is determined as the distance of the k_{th} element in the sorted list where $k=\lfloor\sqrt{N+N'}\rfloor$. The lowest *F(i)* value is preferred that indicates the least dominated solution diversity is provided by *D(i)* calculation. The fine grained fitness assignment strategy inclines choosing the least-dominated and most diversified solution.

Initialize the population size N and archive size N'
Generate a permutation R of heuristics in the system.
T is the maximum generation number, $t= 0$
Generate an initial population $P(0)$ and create the empty archive $A(0) =\emptyset$
Calculate fitness of solutions in $P(t)$ using thefine-grained fitness assignment strategy and copy them to archive $A(t)$
While ($t\leq T$) **do**
  $M(t)$: Select parents by binary tournament selection
  $P(t+1)$: Apply crossover operator to $M(t)$
  $P(t+1)$: Apply the next heuristic in R as mutation operators to $M(t)$
  Calculate fitness of solutions usingfine-grained fitness assignment strategy. Find thenumber of non-dominated solutions N_{NDS}
  if ($N_{NDS}=N$) **then**
    Copy all N_{NDS} to $A(t+1)$
  else if ($N_{NDS}<N$)
    Copy all non-dominated solutions to $A(t+1)$ and dominated solutions using the fitness till the archive is full
  Else
    Eliminate non-dominated solutions one by one using the truncation operator
  end if
  $t=t+1$
end while

Figure 5.1. The pseudo code of SPEA2-DNG.

The pseudo code of the SPEA2-DNG is given in Figure 5.1. Parameters are initialized at the beginning. Initial population is generated in a random manner and the fitness of the population is calculated using fine grained fitness assignment strategy. Population at time *t+1* (*P(t+1)*) is constructed by crossover and DNG mechanisms. The crossover is handled for evolving the solution and DNG is operated as the mutation mechanism. *P(t+1)* is sorted according to fine-grained fitness assignment strategy. The non-dominated solutions are transferred into the next generation if the number of non-dominated solutions does not

exceed the population size *N*. The rest of the generation is completed according to the fitness values.

5.1.3 Pareto simulated annealing with DNG

Simulated annealing (SA) was firstly proposed for single objective optimization problems [**85**] based on evolving and resulting a single solution. However a single solution is not proper outcome for multi-objective optimization problems. Researchers have proposed many SA variations for MOPs up to now [**121**], [**122**]. Most of them comprise an archive where non-dominated solutions are stored.

SA inspired from the annealing process of metal work. The major parameters in SA are as follows:

- *T* is the starting temperature.
- *R* is the number of iterations at each temperature.
- *a* is the cooling rate.
- *b* is the increment variable of *R*.

Generate a random initial solution and indicate it as current solution (*cs*), initialize temperature *T*
Generate a permutation *P* of heuristics in the system
Initialize an empty Archive *A* that stores non-dominated solutions
Initialize $Delta_i$ where is the number of objectives of the problem
Best solution(*bs*) = *cs*
While termination condition is not satisfied
  For *R* times
    Obtain a neighbor solution, *ns* by applying the next heuristic in *P* to*cs*
    $Delta_i$ is of *ns* –*cs* for each objective function *i*
    Update archive *A*
    if *ns* dominated *cs*
      cs = *ns*
    else if random() < $e^{\prod_{i=0}^{k}\left(\frac{|Delta_i|}{T}\right)}$
      cs = *ns*
  end for
  T = *T* / *a*
  R = *R* * *b*
end while

Figure 5.2. The pseudo-code of PSA-DNG.

The larger *T* value, encourages more inferior solutions. After a predetermined number of *R* iterations, *T* is set to *T=T/a*. When *a* is large, the temperature decrease ratio is large and the acceptance of inferior solutions become less likely at a greater rate. When temperature becomes less than a predefined value, SA stops (the stopping condition). Number of

iterations at each temperature is limited by R and greater values of R correspond to slower cooling, that is, more neighbour solution trial is occurring when there is a greater likelihood of inferior exchanges being accepted. In our implementation, R value changes and increases by an initially specified b value. The probability of accepting the worse outcomes decreases as the temperature decrease.

The pseudo-code of PSA-DNG is given in Figure 5.2. The parameters, a permutation of heuristics and an empty archive are initialized at the beginning. A neighbour solution is produced by exploiting the next operator in the permutation P.

5.1.4 Multi-objective hyper-heuristic evolutionary algorithm

Hyper heuristics (HH) is a common way of exploiting multiple heuristic operators. HH was firstly proposed for SOPs as a single solution improvement technique and then adapted to MOPs as population based approaches by the time. HH is known as a heuristic to manage a set of (meta)heuristics. The main two mechanism in HH are heuristic selection mechanism and move acceptance mechanism. Move acceptance mechanism of HH is consolidated with the non-dominated sorting procedures in MOPs.

We have implemented multi-objective hyper heuristics evolutionary algorithm (MHypEA) which is described in [**100**] to add a multi-heuristic approach other than DNG framework to our experiments. The pseudo code of the algorithm is given in Figure 5.3. The parent population is initialized by a predetermined size N. During the run, a low-level heuristic is selected based on the selection mechanism. We preferred using random permutation heuristic selection strategy as it overlaps the heuristic selection mechanism in DNG. Random permutation firstly constructs a permutation of low-level heuristics and applies each one in order according to the permutation. The parent population and the offspring population are combined and sorted by non-dominated sorting by taking crowding distance into consideration.

```
Initialize parent population size N
Evaluate the fitness of parent population
While (not termination-condition) do
    Select a low-level heuristic based on the selection mechanism
    Apply the selected low-level heuristic on the parent population and obtain offspring population
    Evaluate the fitness of offspring population
    Combine parent and offspring populations
    Perform non-dominated sorting on the combined population and select the individuals
(for the next generation) starting from the first front by taking crowding distance into
consideration
end while
```

Figure 5.3. The pseudo code of the MHypEA.

5.1.5 Neighbour generation techniques

In this subsection, aforementioned mutation and crossover techniques that are used to evolve the solutions are described here. The mutation techniques are exploited as heuristic operators in MMBO, MMBO-DNG, PSA and PSA-DNG. Mutation operator and heuristic operator definitions have been used to name the same concepts throughout the chapter.

5.1.5.1 Uniform mutation

Uniform mutation selects a random gene from the population and replaces the value of the chosen gene g_k with a uniform random value between the specified upper and lower bounds (g_k^u and g_k^l) .

5.1.5.2 Non uniform mutation

For each individual G_i^t in a population of the generation *t*, a mutated individual G_i^{t+1} is created through non-uniform mutation as follows: if G_i^t $\{g_1 , g_2 ,\dots g_m\}$ is a population and the element g'_k is selected for the mutation, the result is the vector G_i^{t+1} $\{g'_1, g'_2,\dots g'_m\}$. g'_k is computed as follows:

$$g'_k = \begin{cases} g_k + \Delta(t, g_k^u - g_k), if\ random\ \alpha\ is\ 0 \\ g_k - \Delta(t, g_k - g_k^l), if\ random\ \alpha\ is\ 1 \end{cases}$$

where g_k^u and g_k^l represents the prespecified lower bound and upper bound values respectively. The function $\Delta(t,y)$ returns a value in range $[0,y]$. As *t* increases, $\Delta(t,y)$ approximates to zero. In this way, the operator firtsly search the solution space uniformly for small *t* values. In lates stages when *t* increases operator locally searchs the solution space. The probability of generating a new number close to its successor rather than a random choice is increased by this strategy. $\Delta(t,y)$ is calculated as:

$$\Delta(t, y) = y.(1 - r^{(1-t/T)^b})$$

where *r* is a uniform random number from *[0,1]*, *T* is the maximum generation number, and *b* is a system parameter determining the degree of dependency on the iteration number [**123**].

5.1.5.3 Polynomial mutation

Polynomial mutation is based on polynomial probability distribution [**124**]. Gene g_k at

time *(t+1)* is calculated as:.

$$g_k\ (t+1) = g_k\ (t) + \left(\ g_k^u - g_k^l\right) * \gamma_i$$

where g_k^u is the upper bound and g_k^l is the lower bound of gene g_k . The parameter γ_i is calculated from the polynomial probability distribution, n_m is the polynomial distribution index and *r* is a random number created within [0, 1].

$$\gamma_i \quad = \begin{cases} 2r_i^{\left(\frac{1}{n_m}+1\right)} - 1, r < 0.5 \\ 1 - \left[2\left(1 - r_i\ \right)\right]^{\left(\frac{1}{n_m}+1\right)}, r \geq 0.5 \end{cases}$$

5.1.5.4 Simulated binary crossover

Simulated binary crossover (SBX) creates two offsprings by handling two parent solutions [**125**], [**126**]. The difference between offspring and parent depends on crossover index η, which is any non-negative real number. A large value of η gives a higher probability for creating "near-parent" solutions and a small value of η allows distant solutions to be selected as offspring. The two children composed are symmetric regarding the parent solutions. The children have a spread which is proportional to that of the parent solutions for a fixed η.

5.1.5.5 Binary tournament selection

Binary tournament selection method is used for creating the mating pool. Two solutions are randomly selected from the population. If the ranks of the solutions are different, the one with the lower rank is chosen. Otherwise the one with the higher crowing distance is chosen [**127**].

5.2 Experimental Setup

The performance measurement indicator, the tackled test problems and parameter fine tuning experiment details are described in this section.

The implementations are done based on a test suite proposed [**128**]. Test suite was designed on a variety of programming languages in an adaptive manner, so that one can practically integrate new features to the software. All algorithms are coded in Java programming language on Eclipse IDE running on Intel i7-5500U CPU, and 16 GB memory computer operated by Windows 10 operating system. Regarding the measurement of the performance of the algorithms, in order to detect robustness and

obtain a unbiased set of results, we repeated each test 100 times. That is, each figure given in the tables is the mean of 100 runs of the algorithms. HV and IGD indicators are utilized for measuring the performances (please see Section 4.4.1 for detailed information about the metrics). In this work the number of points in approximated *POF* input files was 1000, which is assumed to be a high value.

5.2.1 Test problems

In order to compare the performance of the proposed algorithms, five multi-objective test problems are used from the test suite proposed by [**120**]. The characteristics of test problems are given in Table 5.1, where abbreviations and the names of the problems are presented. In the table, *M* is the number of objectives and *D* is the number of decision variables. The approximated Pareto front distribution of five tackled problems are illustrated in Figure 5.4.

Table 5.1. Test problem characteristics list.

Problem	Name	M	D	Variables	Pareto Front
RE 2-4-1	Four bar truss design [**129**]	2	4	Continuous	Convex
RE 2-3-2	Reinforced concrete beam design [**130**]	2	3	Mixed	Mixed
RE 2-4-3	Pressure vessel design [**131**]	2	4	Mixed	Mixed, Disconnected
RE 2-2-4	Hatch cover design [**130**]	2	2	Continuous	Convex
RE 2-3-5	Coil compression spring design [**132**]	2	3	Mixed	Mixed, Disconnected

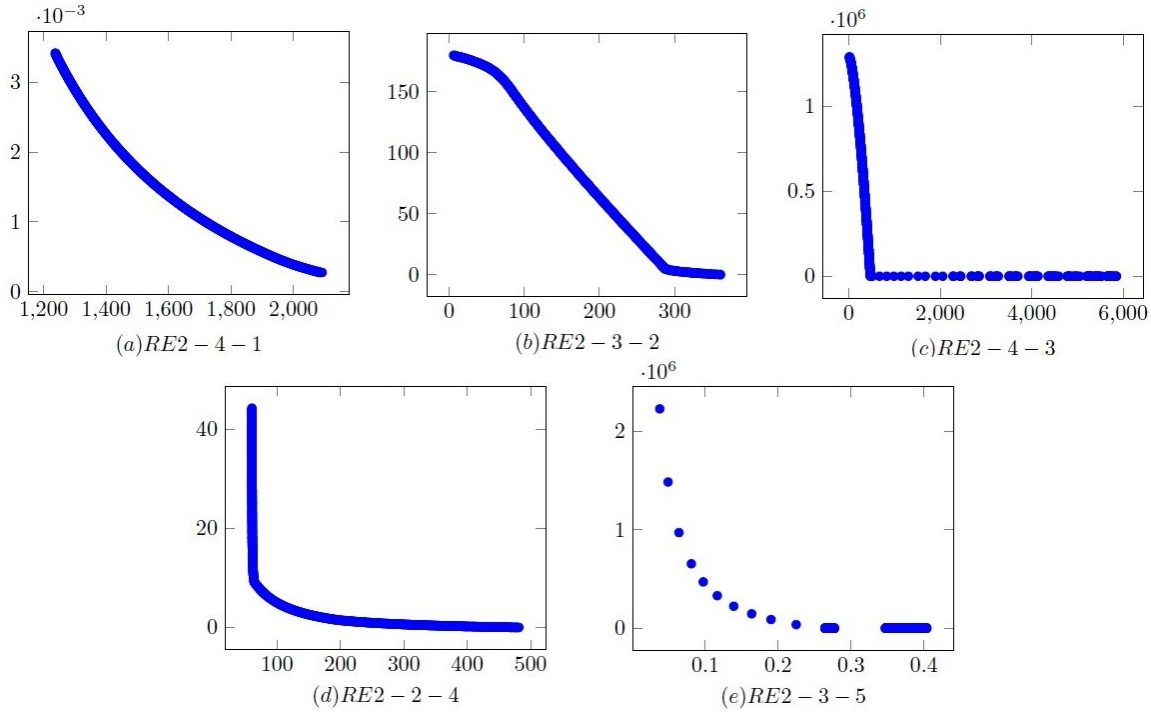

Figure 5.4. Pareto optimal front distribution of test problems. The x and y axis represent objective function 1 and objective function 2, respectively.

5.2.2 Parameter tuning

In order to get the best performance from the algorithms, we need to find out the best performing values of their parameters. These best performing values are discovered with fine tuning experiments. The number of solutions that algorithms can generate while surfing in the solution space is limited to 20,000. The experiments are repeated ten times and results are acquired from the average performances in non-dominated solution sets of these ten runs. Parameter fine tuning results are presented in Table 5.2. Adjusted parameters may be listed as follows:

- *N* is the population size,
- *pCx* is the crossover probability,
- *pM* is the mutation probability,
- *nob* is the number of birds in the flock,
- *non* is the number of neighbours,
- *nof* is the number of flakes,
- *olf* is the overlap factor,
- *T* is the starting temperature,
- *R* is the number of iterations at each temperature,
- *a* is the cooling rate,
- *b* is the increment variable of *R*.

Table 5.2. Parameter fine tuning results for all algorithms.

	NSGA-II	**NSGA-II-DNG**		**MBO**	**MBO-DNG**		**MHypEA**
Param	**Values**	**Values**	**Param**	**Values**	**Values**	**Param**	**Values**
N	**100**, 150, 200	**100**, 150, 200	**nob**	5, 21, 51,**101**	5, 21, 51,**101**	N	**100**, 150, 200
pCx	**0.5**, 0.7, 0.9	**0.5**, 0.7, 0.9	**non**	3, 5, **7**, 10	3, 5, **7**, 10		
pM	0.1, **0.3**, 0.5	0.1, 0.3,**0.5**	**nof**	5, **10**	5, **10**		
			olf	**1**,2,3	**1**,2,3		

	PSA	**PSA-DNG**		**SPEA2**	**SPEA2-DNG**	
Param	**Values**	**Values**	**Param**	**Values**	**Values**	
T	**400**,100,010,000	**400**,100,010,000	**N**	50,80,**100**	50,80,**100**	
R	**20**, 50, 100	**20**, 50, 100	**A**	50,80,**100**	50,80,**100**	
a	1.2, 1.5, **1.9**	1.2, 1.5, **1.9**	**pCx**	**0.6**, 0.7, 0.9	**0.6**, 0.7, 0.9	
b	1.1, **1.5**, 1.9	1.1, **1.5**, 1.9	**pM**	0.1, **0.3**, 0.5	0.1, 0.3, **0.5**	

NSGA-II, SPEA2 and PSA enforce a single mutation technique and MMBO performs a

unique heuristic technique for improving the solutions. Since we have three different heuristic techniques, we have conducted an experiment to understand which of the three operators in the system is compatible with which algorithm. In order to assess a fair comparison among all applied techniques, we set the stopping criterion of the algorithms based on the number of instances (*noi*) created while each algorithm is running. This experiment is done on each the benchmark instances, while *noi* is set to 20,000 and each case is repeated 10 times. Heuristics are listed in the Table 5.3 from the best performing one to the worst performing one. Uniform Mutation performs best for NSGA-II and MMBO, polynomial mutation is best for SPEA2 and PSA.

Table 5.3. Best performing heuristics for algorithms.

Algorithm	Neighbour Generation Techniques (ordered from best to worst)
NSGA-II	**Uniform Mutation**, Polynomial Mutation, Non-Uniform Mutation
MMBO	**Uniform Mutation**, Polynomial Mutation, Non-Uniform Mutation
SPEA2	**Polynomial Mutation**, Uniform Mutation, Non-Uniform Mutation
PSA	**Polynomial Mutation**, Uniform Mutation, Non-Uniform Mutation

5.3 Results and Discussion

In this section, we present and analyze the results of the computational experiments in terms of IGD and HV indicators. NSGA-II, NSGA-II-DNG, MMBO, MMBO-DNG, SPEA2, SPEA2-DNG, PSA, PSA-DNG, MHypEA are implemented to be used in experiments. The aim is to make an extensive comparative analysis.

We have conducted computational experiments to trace the change by the increase of search opportunity. We limited the number of evaluations with 20,000 due to the quick approximation capability of applied algorithms to the Pareto optimal front. One run takes 112.07 milliseconds on the average for all algorithms. The outcomes remained stationary when the evolution number is more than 20,000.

The IGD results are utilized to trace the change with respect to the increasing evaluations. Solution techniques are tested for *noi* 500, 800, 1000, 5000, 10000, 20000. Average IGD values obtained from the 100 runs are demonstrated in Table 5.4. We should remind that, smaller IGD value demonstrates results those are closer to Pareto optimal front, therefore a more thriving performance. Each problem have distinctive characteristics so that it would be better to evaluate the results separately for each of them.

Table 5.4. Average IGD values for noi=500, 800, 1000, 5000, 10000, 20000.

Benchmark	Algorithm	500	800	1,000	5,000	10,000	20,000
RE 2-4-1	**NSGA-II**	**0.0535**	**0.05051**	0.0503	0.04802	0.04808	0.04811
	NSGA-II-DNG	0.0545	0.05173	0.05148	0.04856	0.04807	0.04801
	MMBO	0.0563	0.05183	**0.05021**	**0.0479**	**0.04784**	**0.04784**
	MMBO-DNG	0.0562	0.05205	0.05114	0.04804	0.04788	0.04788
	SPEA2	0.0578	0.05254	0.0524	0.04944	0.04909	0.04845
	SPEA2-DNG	0.0567	0.05552	0.05186	0.04868	0.04818	0.04804
	PSA	0.1006	0.09731	0.09591	0.09492	0.085	0.08226
	PSA-DNG	0.1195	0.10973	0.10954	0.10384	0.09086	0.07612
	MHypEA	0.0502	0.04826	0.0483	0.048	0.04798	0.04794
RE 2-3-2	**NSGA-II**	0.0019	0.00119	0.00089	0.00054	0.00073	0.00078
	NSGA-II-DNG	**0.0013**	**0.00086**	**0.00069**	0.00072	0.00053	0.00084
	MMBO	0.0018	0.0011	0.00092	**0.00023**	0.00033	0.00025
	MMBO-DNG	0.0016	0.00104	0.00073	0.00026	**0.00023**	**0.00022**
	SPEA2	0.0018	0.00092	0.00073	0.00062	0.00078	0.00068
	SPEA2-DNG	0.0019	0.00124	0.00083	0.00035	0.00034	0.00032
	PSA	0.009	0.00892	0.00101	0.00082	0.00067	0.00059
	PSA-DNG	0.0041	0.00401	0.00158	0.00058	0.00046	0.00042
	MHypEA	0.0022	0.0017	0.0011	0.00067	0.00066	0.00064
RE 2-4-3	**NSGA-II**	0.0078	0.00421	0.00498	0.00147	0.00163	0.00187
	NSGA-II-DNG	**0.0062**	**0.00296**	**0.00197**	0.0005	0.00053	0.00049
	MMBO	0.0103	0.00557	0.00376	0.00116	0.00101	0.00082
	MMBO-DNG	0.0097	0.00415	0.00374	**0.00027**	**0.00026**	**0.00025**
	SPEA2	0.0084	0.00792	0.00735	0.00663	0.00504	0.00437
	SPEA2-DNG	0.0091	0.00806	0.00765	0.00655	0.00443	0.00435
	PSA	0.018	0.01374	0.01238	0.00661	0.00558	0.00399
	PSA-DNG	0.0105	0.00655	0.00537	0.0036	0.00338	0.0026
	MHypEA	0.0156	0.0089	0.01172	0.0006	0.00059	0.00055
RE 2-2-4	**NSGA-II**	0.0101	0.01066	0.00826	0.0005	0.00049	0.0005
	NSGA-II-DNG	**0.0065**	0.00699	**0.00579**	0.00051	0.00051	0.00049
	MMBO	0.0084	**0.00582**	0.00668	**0.00025**	0.00024	0.00024
	MMBO-DNG	0.0085	0.00677	0.00559	0.00026	**0.00024**	**0.00023**
	SPEA2	0.0093	0.0092	0.00917	0.00081	0.0008	0.00079
	SPEA2-DNG	0.0092	0.00918	0.00898	0.00085	0.00084	0.00074
	PSA	0.0089	0.00826	0.00687	0.00543	0.00052	0.00058
	PSA-DNG	0.0057	0.00338	0.00233	0.00093	0.00033	0.00029
	MHypEA	0.0059	0.00418	0.00237	0.00053	0.00053	0.00053
RE 2-3-5	**NSGA-II**	**0.0019**	**0.00124**	0.00054	0.00086	0.00074	0.0001
	NSGA-II-DNG	0.0031	0.00195	**0.00047**	0.0001	0.00009	0.00009
	MMBO	0.0059	0.00353	0.00105	0.00017	0.000032	0.000032
	MMBO-DNG	0.0054	0.0027	0.00184	**0.00003**	**0.000031**	**0.000031**
	SPEA2	0.0054	0.00178	0.00153	0.00062	0.00058	0.00084
	SPEA2-DNG	0.0073	0.00159	0.00138	0.00038	0.00036	0.00027
	PSA	0.0235	0.00509	0.00256	0.00122	0.00083	0.00064
	PSA-DNG	0.0154	0.00993	0.00651	0.00076	0.00006	0.0002
	MHypEA	0.0061	0.00171	0.00309	0.00013	0.00011	0.00011

Problem RE-2-4-1: NSGA-II achieved the best performance for *noi* equal to 500 and 800, while MMBO is the best for the upcoming cases. DNG versions of the algorithms achieved 2.11% better IGD results than their original versions on the average.

Problem RE-2-3-2: NSGA-II performs best for small iteration numbers, while MMBO-

DNG produces best results for *noi* =10000 and *noi* =20000.

Problem RE-2-4-3: Multi-mutation version of NSGA-II outperforms for *noi* equal to 500-1000. Multi-heuristic MMBO performs best for *noi* is equal to 5000-20000. For this problem, the multi heuristic/mutation versions obtained better results for each experimental case than the single-operator versions.

Problem RE-2-2-4: NSGA-II is best when *noi* for the least number of evaluations (*noi* = 500). MMBO or MMBO-DNG performs best for the higher evolution values. Both algorithms MMBO and MMBO-DNG produce very close IGD values even one is better than the other in each when *noi* =10000 and *noi* =20000. When *noi* is equal to 10000 MMBO-DNG is the best and 51.42% better than NSGA-II. MMBO is the best for *noi* = 20000 and 50.04% better to approximate Pareto optimal front than NSGA-II.

Problem RE-2-5-3} NSGA-II is best when *noi*= 500 and *noi*=800. NSGA-II-DNG is best for *noi* is equal to 1000. MMBO-DNG performs best for the higher evolution values. Multi heuristic/mutation versions produced better IGD values in each case for *noi* =5000,10000 and 20000.

The results are visualized in Figure 5.5. In the experiments, the starting random seeds of all algorithms are set to be the same, so all algorithms start from the same initial point. PSA and PSA-DNG prominently demonstrate worse performance than the other algorithms for *RE-2-4-1*. It is also observed from the figure that SA moves away from the Pareto optimal for small *noi* values at the beginning of the experience. This deviation may be expounded as follows: Temperature *T* in PSA and PSA-DNG is high when *noi* is equal to a small number such as 500, 800 or 1000 hence the probability of accepting worse results is high. For higher *noi* values *T* decreases and probability of accepting worse results decreases as a result.

As a summary, we observe that NSGA-II and NSGA-II-DNG present good performance when *noi* $\leq$ 1000. The good performance of NSGA-II and NSAGii-DNG is expected because of its widely acceptance by the optimization community. However, when for *noi* $\geq$ 5000 we can see that MMBO and MMBO-DNG become the favorite player in the game. With this result, we can say that the promising performance of MBO is also shown in multi-objective optimization problems. Furthermore, our contribution with this chapter, that is the DNG framework, is shown to put a great performance by obtaining the best result in almost two thirds of the tests, which we believe to be an important contribution to the multi-objective optimization research field.

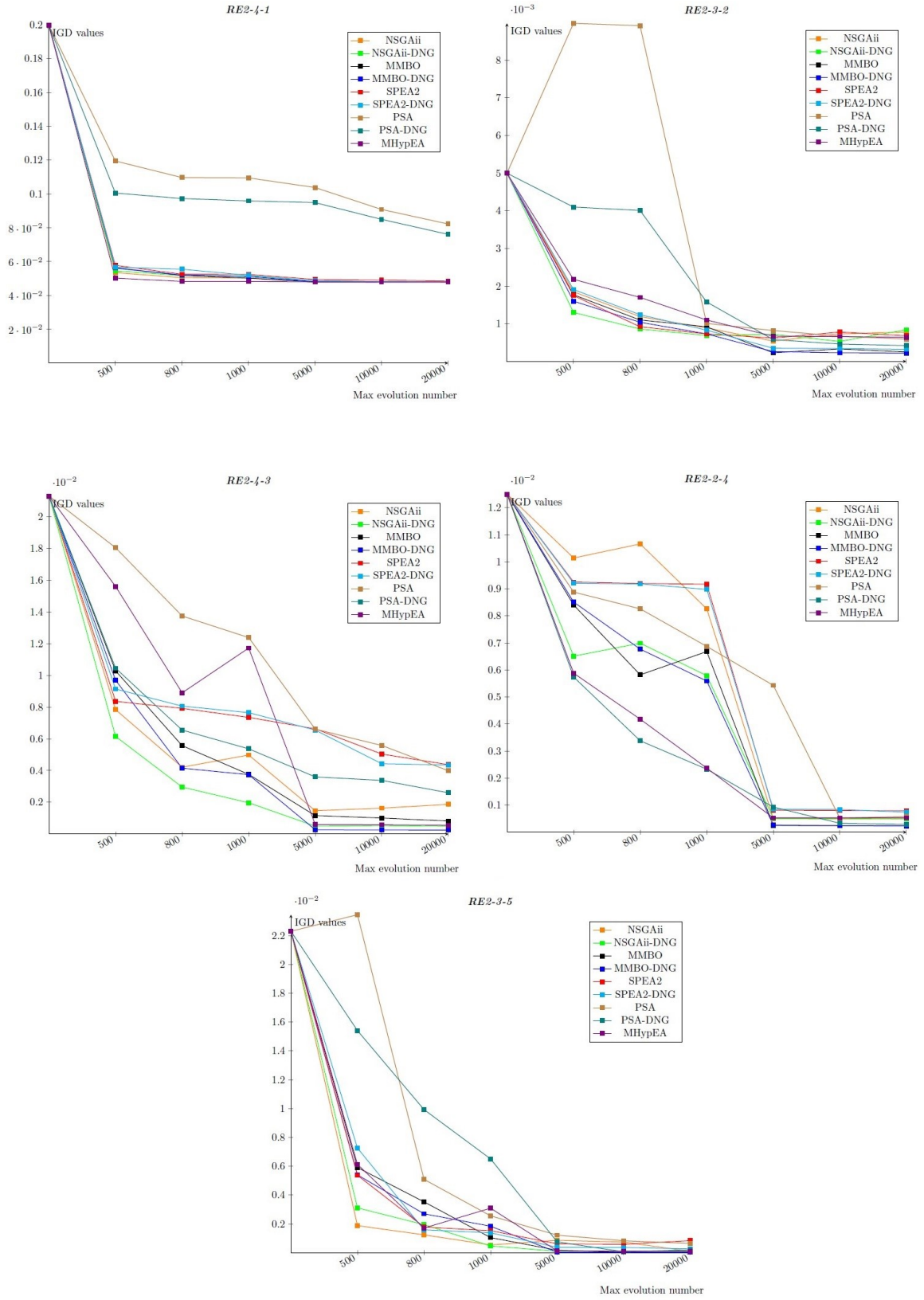

Figure 5.5. IGD change of each algorithm for increasing evolution numbers.

Additionally, the results are analyzed in terms of HV indicator. The average HV values of 100 runs for all compared algorithms when *noi* is equal to 20000 are given in Table 5.5. It is worth to remind that higher HV indicates a better performance. Nearly all algorithms are

successful in catching the best HV value for the first problem (*RE 2-4-1*). MMBO-DNG gains the best HV results for all test problems.

Table 5.5. Average HV values when noi is equal to 20000.

Algorithm/Problem	RE 2-4-1	RE 2-3-2	RE 2-4-3	RE 2-2-4	RE 2-3-5
NSGA-II	**3.E+04**	5.E+04	7.E+09	3.E+04	5.E+07
NSGA-II-DNG	**3.E+04**	5.E+04	7.E+09	3.E+04	5.E+07
MMBO	**3.E+04**	7.E+04	2.E+11	1.E+04	3.E+08
MMBO-DNG	**3.E+04**	**3.E+06**	**3.E+11**	**1.E+05**	**4.E+08**
SPEA2	2.E+04	3.E+04	2.E+08	1.E+04	4.E+07
SPEA2-DNG	**3.E+04**	1.E+05	1.E+09	1.E+04	4.E+07
PSA	2.E+04	3.E+05	7.E+09	3.E+04	3.E+07
PSA-DNG	2.E+04	3.E+05	7.E+09	7.E+04	5.E+07
MHypEA	**3.E+04**	5.E+04	8.E+09	3.E+04	5.E+07

The DNG versions of the algorithms are compared with their original models with the goal of presenting our contributions more strikingly. The results obtained from this pairwise comparison in terms of IGD and HV indicators are presented in Table 5.6. The success of multi-heuristic based DNG variants may differ due to differences in problem structures. As a result, in terms of IGD indicator, the DNG versions of the algorithms are better than their original versions by 24.39%, on the average. As well as, in terms of HV indicator the DNG variants are 28.35% better than their original versions on the average.

Table 5.6. Percentage deviations of algorithms' IGD and HV results from original versions vs DNG versions.

IGD						
Algorithm/Problem	**RE 2-4-1**	**RE 2-3-2**	**RE 2-4-3**	**RE 2-2-4**	**RE 2-3-5**	**Average**
NSGA-II vs NSGA-II-DNG	0.21	-7.69	73.8	2	10	15.66
MMBO vs MMBO-DNG	-0.08	12	69.51	4.17	3.13	17.74
SPEA2 vs SPEA2-DNG	0.85	52.94	0.46	6.33	67.86	25.69
PSA vs PSA-DNG	7.46	29.16	34.73	50.85	70.05	38.45
Average	2.11	21.6	44.62	15.84	37.76	24.39
HV						
Algorithm/Problem	**RE 2-4-1**	**RE 2-3-2**	**RE 2-4-3**	**RE 2-2-4**	**RE 2-3-5**	**Average**
NSGA-II vs NSGA-II-DNG	0	2.1	1.19	0.01	0	0.66
MMBO vs MMBO-DNG	2.4	97.83	32.54	89.85	30.51	50.62
SPEA2 vs SPEA2-DNG	10.85	8.92	6.95	61.85	43.79	26.47
PSA vs PSA-DNG	14.34	71.7	83.66	6.57	1.99	35.65
Average	6.9	45.14	31.09	39.57	19.07	28.35

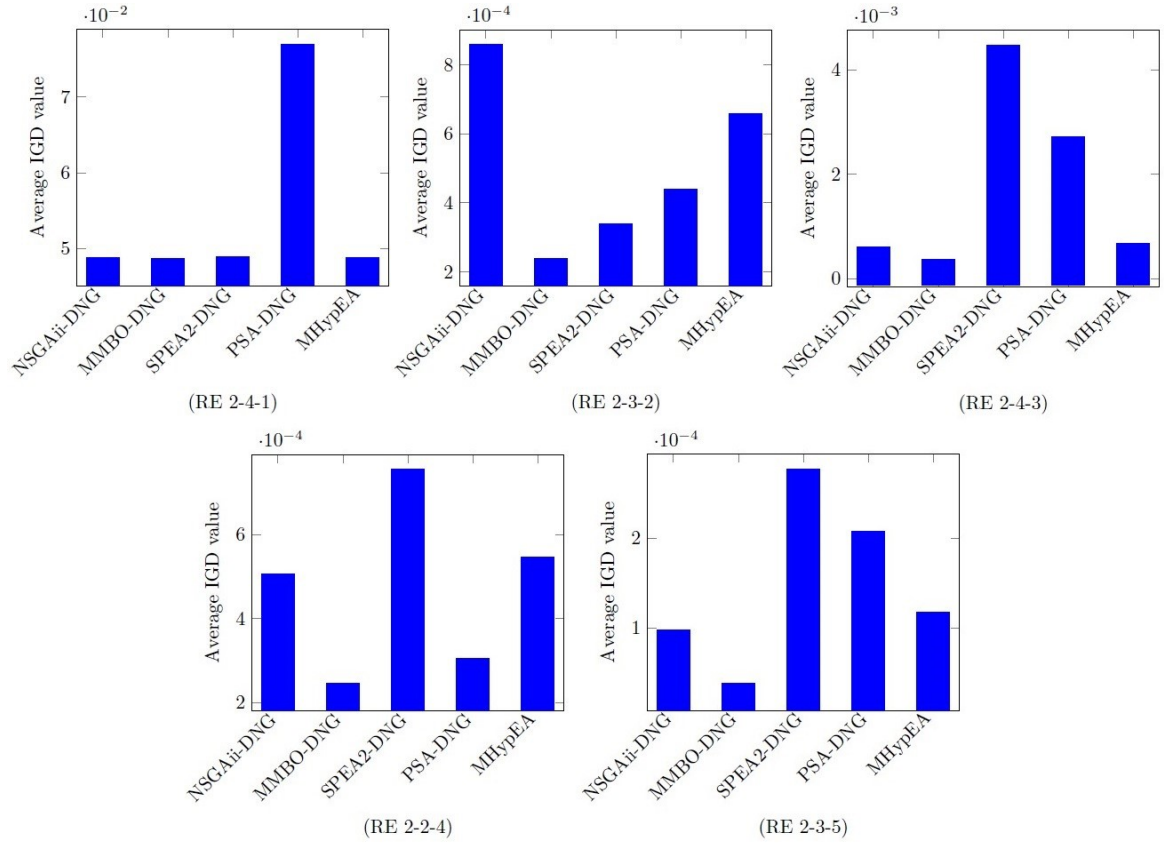

Figure 5.6. Average IGD values for MMBO-DNG and MHypEA for all test problems.

The MHypEA is a state-of-the-art algorithm that has been commonly used in the literature and creates neighbour solutions by more than one heuristics. In this respect, it is also logical to compare the performance of MHypEA with that of the DNG variants proposed in this chapter. The average IGD and HV values for MHypEA and for each DNG variant are visualized in Figure 5.6 and Figure 5.7. Let us remind that smaller IGD outcome indicates a better result contrarily higher HV outcome inticates a better result. The results show that MHypEA is capable of approximating the best value set, moreover it is better than some DNG variants for some problems. However, MMBO-DNG is more capable to approximate Pareto optimal front than MHypEA for all cases. MHypEA and DNG-based algorithms in our experiments are similar as they have the ability to search the solution space using multiple heuristics. However, DNG can easily be integrated into any meta-heuristic algorithm, further enhancing the already strong nature of the algorithms it is integrated with.

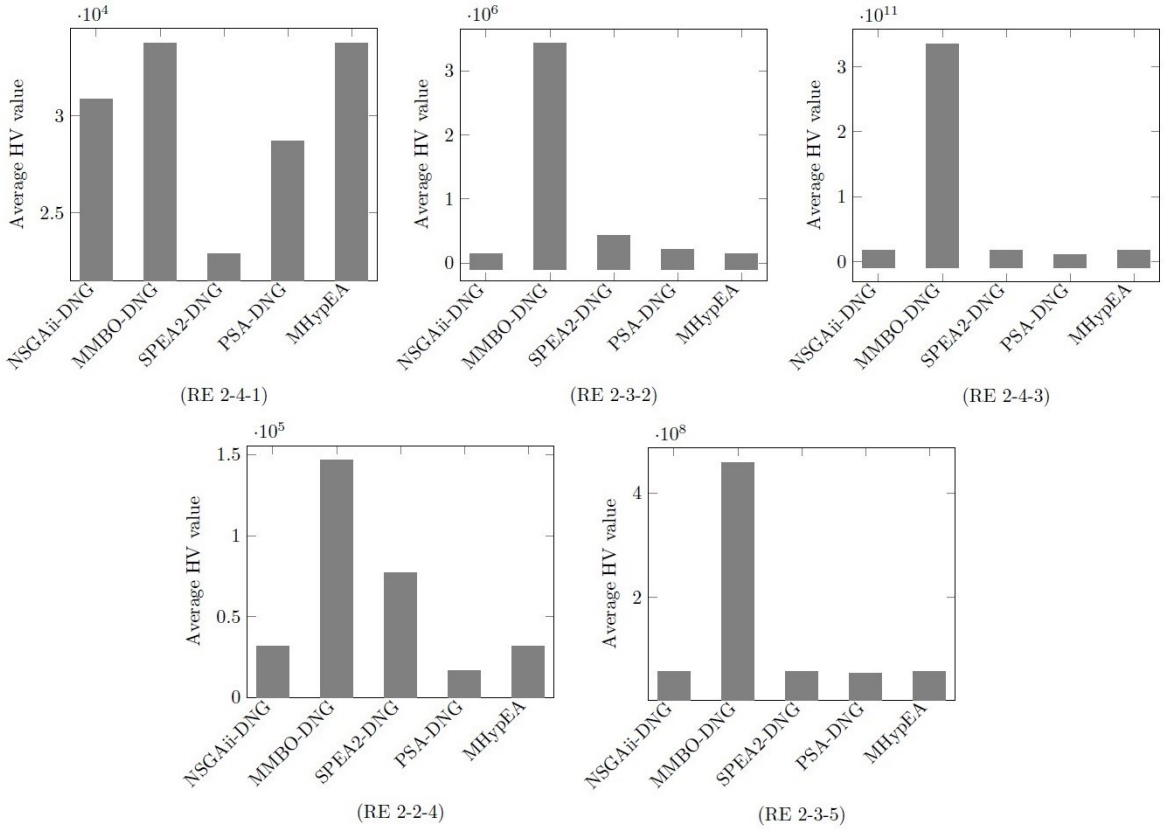

Figure 5.7. Average HV values of DNG variants and MHypEA for all test problems.

In order to validate our results statistically, we present the results of standard deviation and one way variance analysis, in addition to the average values. Firstly, the standard deviations of IGD indicator are presented in Table 5.7 to demonstrate the robustness of the algorithms. It is observed that MMBO is more robust than its rivals. Although all algorithms are robust enough, we can say that the original variants produce slightly better standard deviation results than their DNG variants. This can be explained by the design of the algorithms as follows. The DNG variants wander within the solution space with different mutation operators whereas the original algorithms exploit a single mutation operator which inherently makes them more robust.

To analyze whether the differences of nine compared algorithms are statistically significant or not, the one-way analysis of variance (ANOVA) test is conducted at an α= 0.05 significance level for IGD and HV results. Then we calculated the sum of squares (*SS*), the mean squares (*MS*), the *F* statistic (*F*), and the probability that the *F* statistic is greater than the critical *F* (*P-value*). The ANOVA results for all test problems on adopted metrics are listed in Table 5.8. All p-values are close to zero which shows the model is much better than the 0.95 confidence level. This demonstrates that the differences between

group means are statistically significant.

Table 5.7. Average standard deviation of IGD results.

Algorithm	RE 2-4-1	RE 2-3-2	RE 2-4-3	RE 2-2-4	RE 2-3-5	Average
NSGA-II	0.00142	0.00026	0.00166	0.00127	0.00102	0.00113
NSGA-II-DNG	0.00139	0.0004	0.00252	0.00234	0.00091	0.00151
MMBO	0.00217	0.00016	0.00095	0.00097	0.0005	0.00095
MMBO-DNG	0.00192	0.00085	0.00095	0.00139	0.00068	0.00116
SPEA2	0.00197	0.00017	0.00145	0.00109	0.00085	0.0011
SPEA2-DNG	0.00253	0.00036	0.00151	0.00117	0.00083	0.00128
PSA	0.0033	0.00359	0.00315	0.00285	0.00262	0.0031
PSA-DNG	0.00335	0.00442	0.00418	0.00282	0.00398	0.00375
MHypEA	0.00055	0.00043	0.00285	0.00132	0.00118	0.00127

Table 5.8. The ANOVA results for IGD and HV metrics and for each test problem.

Metric	Problem	SS	MS	F	P-value
IGD	**RE 2-4-1**	0.0367	0.0046	12.81	1.00E-11
	RE 2-3-2	0.0003	0	7.69	1.00E-07
	RE 2-4-3	0.0008	0.0001	124.9	3.00E-42
	RE 2-2-4	0.0011	0.0001	37.43	4.00E-24
	RE 2-3-5	0	0	39.56	7.00E-25
HV	**RE 2-4-1**	2.00E+10	3.00E+09	295.03	2.00E-244
	RE 2-3-2	2.00E+10	3.00E+09	295.03	2.00E-244
	RE 2-4-3	1.00E+25	1.00E+24	2027.82	3.00E-221
	RE 2-2-4	1.00E+12	2.00E+11	587.64	5.00E-121
	RE 2-3-5	2.00E+19	2.00E+18	703.13	2.00E-221

Additionally we have conducted post-hoc Tukey HSD test to analyze pairwise statistical significance. The results of Tukey HSD test is reported in Table 5.9. “s+” shows the approach in “Algorithm 1” column performs statistically better than the approach in “Algorithm 2” column, s− shows the vice versa. “+” shows the approach in “Algorithm 1” column performs slightly better than the approach in “Algorithm 2” column and “-” shows the vice versa. Results show that differences between the algorithms are statistically significant.

Table 5.9. Pair-wise comparison of algorithms for HV and IGD indicators due to Tukey HSD.

		IGD					HV				
Algorithm 1	**Algorithm 2**	**RE 2-4-1**	**RE 2-3-2**	**RE 2-4-3**	**RE 2-2-4**	**RE 2-3-5**	**RE 2-4-1**	**RE 2-3-2**	**RE 2-4-3**	**RE 2-2-4**	**RE 2-3-5**
NSGAii	**MMBO-DNG**	s-	s-	s-	s-	s-	s-	s-	s-	s-	s-
	NSGAii-DNG	s-	s-	s-	s-	-	-	s-	s-	-	-
	MMBO	s-	s-	s-	s-	-	-	s-	s-	s-	s-
	SPEA2	s-	s-	s-	s-	s-	s-	s-	s-	s-	s-
	SPEA2-DNG	s-	s-	s-	s-	-	s-	s-	s-	s-	s-
	PSA	s+	+	s+	s+	s+	s+	s+	s+	s+	s+
	PSA-DNG	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	MHypEA	s-	s-	s-	s-	-	-	s-	s-	s-	s-
NSGAii-DNG	**MMBO-DNG**	s-	s-	s-	s-	-	s-	s-	s-	s-	s-
	MMBO	s+	s+	s-	s+	s+	s+	s+	s-	s+	s+
	SPEA2	s+	s+	s-	s+	s+	s+	s+	s-	s+	s+
	SPEA2-DNG	+	s+	s-	s+	+	s+	s+	s-	s+	s+
	PSA	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	PSA-DNG	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	MHypEA	s-	s+	s+	+	-	-	s+	s+	+	s-
MMBO	**MMBO-DNG**	s-	s-	s-	-	-	s-	s-	s-	-	s-
	SPEA2	s+	s+	s-	s-	s+	s+	s+	s-	s-	s+
	SPEA2-DNG	s+	s-	s+	s+	-	s+	s-	s+	s+	s-
	PSA	+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	PSA-DNG	s+	s+	s+	s+	+	s+	s+	s+	s+	s+
	MHypEA	s+	s+	s+	s-	s+	s+	+	-	s-	s+
MMBO-DNG	**SPEA2**	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	SPEA2-DNG	+	s+	s+	s+	+	s+	s+	s+	+	s+
	PSA	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	PSA-DNG	+	s+	s+	s+	s+	s+	s+	s+	s+	s+
	MHypEA	s+	s+	s+	s+	+	s+	s+	s+	s+	s+
SPEA2	**SPEA2-DNG**	s-	s-	s-	s-	-	-	s-	s-	s-	s-
	PSA	s+	s+	s+	s+	s+	s+	s+	+	s+	s+
	PSA-DNG	s+	s+	s+	s+	+	+	+	+	s+	s+
	MHypEA	+	s+	s+	s-	s+	s+	s+	s+	s-	s+
SPEA2-DNG	**PSA**	s+	s+	s+	s+	+	s+	s+	s+	s+	s+
	PSA-DNG	+	s+	+	s+	s+	s+	+	s+	+	s+
	MHypEA	s+	s+	s+	s+	s+	+	+	s+	s+	s+
PSA	**PSA-DNG**	-	s-	s-	s-	-	s-	s-	s-	s-	s-
	MHypEA	s-	s-	s-	s-	s-	s-	s-	s-	s-	s-
PSA-DNG	**MHypEA**	s-	s-	s-	s+	+	-	s-	s-	s+	s+

5.4 Summary

In this chapter, we have presented an extensive comparison for analyzing the achievement of DNG framework. DNG is a framework that allows to use more than one heuristics while searching the solution space and can be easily integrated with other meta-heuristics. A set of state-of-the-art multi-objective evolutionary algorithms and their DNG versions are implemented on a test suite. Those are NSGA-II, NSGA-II-DNG, MMBO,MMBO-DNG, SPEA2, SPEA2-DNG, PSA, PSA-DNG. Additionally, we have added MHypEA to solution suite for revealing a more significant evaluation.

We conducted experimental studies on a set of benchmark test problems. Experimental results indicated that DNG framework improved the original performances of algorithms by 24.39% on the average in terms of IGD indicator and 28.35% on the average in terms of HV indicator. MMBO-DNG has the highest ability to converge to POF among the algorithms participating in the experiment. MMBO-DNG outperformes MHypEA for each problem in terms of average IGD values and it is 49.7% better on the average. The results are also validated through statistical analyses and tests. As a result, it is shown that DNG framework is an important contribution to the multi-objective optimization research field.

6. SUMMARY, CONCLUSIONS, AND FUTURE WORK

In this chapter, a general summary and conclusions of the work done in this thesis are presented. Also, possible directions for future work is proposed in detail.

In this thesis we focused the workforce scheduling problems and the vehicle routing problems in both single objective optimization domain and the multi-objective optimization domain.

In the first part, we presented the mathematical model of a shift scheduling problem of a manufacturing company including the fairness perspective (FOSSP). Besides, a novel algorithm is designed by embedding hyper heuristic improvements in the migrating bird optimization algorithm (HHMBO). The model of FOSSP is firstly verified with a solver where the solver is capable on catching global optima with large execution time even for small instances. Therefore, exploitation of heuristic based techniques was an apparent need. In line with this need, we applied HHMBO together with simulated annealing (SA), hyper heuristics (HH) and classical migrating bird optimization (MBO). We have conducted computational experiments in order to assess the performance of the algorithms on synthetic data. Results show the superiority of HHMBO-RP_OI and it is observed that it outperforms its rivals by 9.3% on the average for FOSSP. Besides, it is seen that HHMBO-RP_OI definitely offers better solutions in large problem instances. We believe that this good performance is due to integrating the exploitation capability of HH with MBO's exploration capability.

We have also applied HHMBO on QAP to justify its superiority over the other algorithms. This experiment was significant to understand whether the achievement of the algorithm is specific to the particular problem. We conducted computational experiments on 41 QAPLIB instances with assorted densities. According to the results of this experiment, AS_OI variant of the new hydrid algorithm could achieve up to 14.6% better results than MBO.

In the second part, we have introduced the mathematical model of a new problem to the literature, which integrates the features of shift assignment problem and the capacitated vehicle routing problem (VRPSA). We have applied the state of the art evolutionary algorithms and a popular swarm algorithm to solve the proposed problem; those are NSGA, NSGA-II and MMBO. In addition to all, we have introduced, three novel

algorithms based on dynamic neighbor generation model are also presented; those are NSGA-DNG, NSGA-II-DNG, MMBO-DNG.

MMBO-DNG produced the best average experimental HV and IGD outcomes and dynamic neighbor generation versions of each algorithm beated its original version for larger noi values. The outcomes uncover that MMBO-DNG definitely offers better solutions in large-sized problem instances. We believe that this good performance is due to integrating the exploitation capability of applying dynamic neighbor generation with MBO's exploration capability.

In the following part, we have presented an extensive comparison for analyzing the achievement of DNG framework on a set of multi-objective test problems. A set of state-of-the-art multi-objective evolutionary algorithms and their DNG versions are implemented on a test suite. Those are NSGA-II, NSGA-II-DNG, MMBO, MMBO-DNG, SPEA2, SPEA2-DNG, PSA, PSA-DNG. Additionally, we have added MHypEA to solution suite for revealing a more significant evaluation. Experimental results indicated that DNG framework improved the original performances of algorithms by 24.39% on the average in terms of IGD indicator. As a result, it is shown that DNG framework is an important contribution to the multi-objective optimization research field.

There are several interesting directions for future research. HHMBO may be applied on different type of optimization problems to highlight its wide applicability and generality. An example may be the test suites given in [**133**]. Furthermore, a comparative study may be conducted on swarm based hybrid algorithms. A multi-objective variation of HHMBO may be developed and may be applied on a set of multi-objective problems. An open route VRPSA may be introduced to the literature. Additionally, we plan applying DNG on dynamic MOPs or it may be integrated with parallel multi-objective optimization algorithms. We aim to improve the heuristic selection part of DNG. Also an investigation may be conducted for observing the achievement of DNG on other MOPs.

REFERENCES

[1] Andreas T. Ernst, Houyuan Jiang, Mohan Krishnamoorthy, Bowie Owens, and David Sier, "An annotated bibliography of personnel scheduling and rostering," *Annals of Operations Research*, vol. 127, pp. 21–144, 2004.

[2] Claudio Ciancio, Demetrio Laganà, Roberto Musmanno, and Francesco Santoro, "An integrated algorithm for shift scheduling problems for local public transport companies," *Omega*, vol. 75, pp. 139–153, 2018.

[3] Albert H. Schrotenboer, Michiel A. J. uit het Broek, Bolor Jargalsaikhan, and Kees Jan Roodbergen, "Coordinating technician allocation and maintenance routing for offshore wind farms," *Computers & Operations Research*, vol. 98, pp. 185–197, 2018.

[4] Christian Fikar and Patrick Hirsch, "Home health care routing and scheduling: A review," *Computers & Operations Research*, vol. 77, pp. 86–95, 2017.

[5] Yann Collette and Patrick Siarry, *Multiobjective optimization: principles and case studies*.: Springer Science & Business Media, 2004.

[6] Seyedali Mirjalili, Pradeep Jangir, Seyedeh Zahra Mirjalili, Shahrzad Saremi, and Indrajit N. Trivedi, "Optimization of problems with multiple objectives using the multi-verse optimization algorithm," *Knowledge-Based Systems*, vol. 134, pp. 50–71, 2017.

[7] Tzung-Pei Hong, Hong-Shung Wang, and Wei-Chou Chen, "Simultaneously applying multiple mutation operators in genetic algorithms," *Journal of heuristics*, vol. 6, pp. 439–455, 2000.

[8] Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck, "Personnel scheduling: A literature review," *European journal of operational research*, vol. 226, pp. 367–385, 2013.

[9] Andreas T. Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European journal of operational research*, vol. 153, pp. 3–27, 2004.

[10] Patrick De Causmaecker, Peter Demeester, G. Vanden Berghe, and Bart Verbeke, "Analysis of real-world personnel scheduling problems," in *Proceedings of the 5th international conference on practice and theory of automated timetabling, Pittsburgh*, 2004, pp. 183–197.

[11] Mieke Defraeye and Inneke Van Nieuwenhuyse, "Staffing and scheduling under nonstationary demand for service: A literature review," *Omega*, vol. 58, pp. 4–25, 2016.

[12] Melanie Erhard, Jan Schoenfelder, Andreas Fügener, and Jens O. Brunner, "State of the art in physician scheduling," *European Journal of Operational Research*, vol. 265, pp. 1–18, 2018.

[13] Peter Brucker, Rong Qu, and Edmund Burke, "Personnel scheduling: Models and complexity," *European Journal of Operational Research*, vol. 210, pp. 467–473, 2011.

[14] Kenneth R. Baker, "Workforce allocation in cyclical scheduling problems: A survey," *Journal of the Operational Research Society*, vol. 27, pp. 155–167, 1976.

[15] Hesham K. Alfares, "Survey, categorization, and comparison of recent tour scheduling literature," *Annals of Operations Research*, vol. 127, pp. 145–175, 2004.

[16] Jens O. Brunner and Raik Stolletz, "Stabilized branch and price with dynamic parameter updating for discontinuous tour scheduling," *Computers & Operations Research*, vol. 44, pp. 137–145, 2014.

[17] Júlíus Atlason, Marina A. Epelman, and Shane G. Henderson, "Call center staffing with simulation and cutting plane methods," *Annals of Operations Research*, vol.

127, pp. 333–358, 2004.

[18] James G. Morris and Michael J. Showalter, "Simple approaches to shift, days-off and tour scheduling problems," *Management Science*, vol. 29, pp. 942–950, 1983.

[19] Fred F. Easton and Donald F. Rossin, "A stochastic goal program for employee scheduling," *Decision Sciences*, vol. 27, pp. 541–568, 1996.

[20] Raik Stolletz, "Operational workforce planning for check-in counters at airports," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, pp. 414–425, 2010.

[21] Fred F. Easton and Donald F. Rossin, "Equivalent alternate solutions for the tour scheduling problem," *Decision Sciences*, vol. 22, pp. 985–1007, 1991.

[22] Hesham K. Alfares, "An efficient two-phase algorithm for cyclic days-off scheduling," *Computers & Operations Research*, vol. 25, pp. 913–923, 1998.

[23] Douglas S. Altner, Erica K. Mason, and Les D. Servi, "Two-stage stochastic days-off scheduling of multi-skilled analysts with training options," *Journal of Combinatorial Optimization*, vol. 38, pp. 111–129, 2019.

[24] Hoong Chuin Lau, "On the complexity of manpower shift scheduling," *Computers & Operations Research*, vol. 23, pp. 93–102, 1996.

[25] Alkin Yurtkuran, Betul Yagmahan, and Erdal Emel, "A novel artificial bee colony algorithm for the workforce scheduling and balancing problem in sub-assembly lines with limited buffers," *Applied Soft Computing*, vol. 73, pp. 767–782, 2018.

[26] Maik Günther and Volker Nissen, *Application of particle swarm optimization to the british telecom workforce scheduling problem*., 2013, vol. 4.

[27] Mukund Nilakantan Janardhanan, Zixiang Li, and Peter Nielsen, "Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem," *Soft Computing*, vol. 23, pp. 11263–11276, 2019.

[28] Haneen Algethami and Dario Landa-Silva, "Diversity-based adaptive genetic algorithm for a Workforce Scheduling and Routing Problem," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1771–1778.

[29] Haneen Algethami, Anna Martínez-Gavara, and Dario Landa-Silva, "Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem," *Journal of Heuristics*, vol. 25, pp. 753–792, 2019.

[30] Kanchana Sethanan and Rapeepan Pitakaso, "Improved differential evolution algorithms for solving generalized assignment problem," *Expert Systems with Applications*, vol. 45, pp. 450–459, 2016.

[31] Shengchao Zhou et al., "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE transactions on cybernetics*, 2019.

[32] Fuqing Zhao, Lexi Zhao, Ling Wang, and Houbin Song, "An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion," *Expert Systems with Applications*, vol. 160, p. 113678, 2020.

[33] Wei-Li Liu et al., "Coordinated charging scheduling of electric vehicles: A mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, pp. 5094–5109, 2019.

[34] Edmund K. Burke et al., "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, pp. 1695–1724, 2013.

[35] Stefania Pan et al., "A hybrid heuristic for the multi-activity tour scheduling problem," *Electronic Notes in Discrete Mathematics*, vol. 69, pp. 333–340, 2018.

[36] Noberto A. Hernández-Leandro, Vincent Boyer, M. Angélica Salazar-Aguilar, and Louis-Martin Rousseau, "A matheuristic based on Lagrangian relaxation for the multi-activity shift scheduling problem," *European Journal of Operational Research*, vol. 272, pp. 859–867, 2019.

[37] Fuqing Zhao et al., "A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion," *Expert Systems with Applications*, vol. 146, p. 113166, 2020.

[38] Shahriar Asta, Ender Özcan, and Tim Curtois, "A tensor based hyper-heuristic for nurse rostering," *Knowledge-based systems*, vol. 98, pp. 185–199, 2016.

[39] Hong Li, Ying Wang, Shi Li, and Sujian Li, "A column generation based hyper-heuristic to the bus driver scheduling problem," *Discrete Dynamics in Nature and Society*, vol. 2015, 2015.

[40] Edmund K. Burke, Graham Kendall, and Eric Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of heuristics*, vol. 9, pp. 451–470, 2003.

[41] Fernando Garza-Santisteban et al., "A simulated annealing hyper-heuristic for job shop scheduling problems," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 57–64.

[42] Morteza Alinia Ahandani, Mohammad Taghi Vakil Baghmisheh, Mohammad Ali Badamchi Zadeh, and Sehraneh Ghaemi, "Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem," *Swarm and Evolutionary Computation*, vol. 7, pp. 21–34, 2012.

[43] Lin Chen, Hong Zheng, Dan Zheng, and Dongni Li, "An ant colony optimization-based hyper-heuristic with genetic programming approach for a hybrid flow shop scheduling problem," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 814–821.

[44] Venkatesh Pandiri and Alok Singh, "A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem," *Information Sciences*, vol. 463, pp. 261–281, 2018.

[45] Ekrem Duman, Mitat Uysal, and Ali Fuat Alkaya, "Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem," *Information Sciences*, vol. 217, pp. 65–77, 2012.

[46] George B. Dantzig and John H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, pp. 80–91, 1959.

[47] Rajeev Goel and Raman Maini, "Vehicle routing problem and its solution methodologies: a survey," *International Journal of Logistics Systems and Management*, vol. 28, pp. 419–435, 2017.

[48] A.A.N. Perwira Redi, Meilinda F. N. Maghfiroh, and F. Yu Vincent, "An improved variable neighborhood search for the open vehicle routing problem with time windows," in *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, 2013, pp. 1641–1645.

[49] Erbao Cao, Mingyong Lai, and Hongming Yang, "Open vehicle routing problem with demand uncertainty and its robust strategies," *Expert Systems with Applications*, vol. 41, pp. 3569–3575, 2014.

[50] Dimitrios Sariklis and Susan Powell, "A heuristic method for the open vehicle routing problem," *Journal of the Operational Research Society*, vol. 51, pp. 564–573, 2000.

[51] S. A. MirHassani and N. Abolghasemi, "A particle swarm optimization algorithm for open vehicle routing problem," *Expert Systems with Applications*, vol. 38, pp. 11547–11551, 2011.

[52] Shiwei Yu, Chang Ding, and Kejun Zhu, "A hybrid GA–TS algorithm for open vehicle routing optimization of coal mines material," *Expert Systems with Applications*, vol. 38, pp. 10568–10573, 2011.

[53] Christos D. Tarantilis, George Ioannou, Chris T. Kiranoudis, and Gregory P. Prastacos, "Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm," *Journal of the Operational Research Society*, vol. 56, pp. 588–596, 2005.

[54] Majid Salari, Paolo Toth, and Andrea Tramontani, "An ILP improvement procedure for the open vehicle routing problem," *Computers & Operations Research*, vol. 37, pp. 2106–2120, 2010.

[55] Eduardo Uchoa et al., "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, pp. 845–858, 2017.

[56] Boon Ean Teoh, Sivalinga Govinda Ponnambalam, and Ganesan Kanagaraj, "Differential evolution algorithm with local search for capacitated vehicle routing problem," *International Journal of Bio-Inspired Computation*, vol. 7, pp. 321–342, 2015.

[57] Kangzhou Wang, Shulin Lan, and Yingxue Zhao, "A genetic-algorithm-based approach to the two-echelon capacitated vehicle routing problem with stochastic demands in logistics service," *Journal of the Operational Research Society*, vol. 68, pp. 1409–1421, 2017.

[58] Lijun Wei, Zhenzhen Zhang, Defu Zhang, and Stephen C. H. Leung, "A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints," *European Journal of Operational Research*, vol.

265, pp. 843–859, 2018.

[59] Jose Bernal, John Willmer Escobar, Juan Camilo Paz, Rodrigo Linfati, and Gustavo Gatica, "A probabilistic granular tabu search for the distance constrained capacitated vehicle routing problem," *International Journal of Industrial and Systems Engineering*, vol. 29, pp. 453–477, 2018.

[60] Eliana M. Toro, John F. Franco, Mauricio Granada Echeverri, and Frederico Gadelha Guimarães, "A multi-objective model for the green capacitated location-routing problem considering environmental impact," *Computers & Industrial Engineering*, vol. 110, pp. 114–125, 2017.

[61] Navid Zarbakhshnia, Hamed Soleimani, Mark Goh, and Seyyedeh Sara Razavi, "A novel multi-objective model for green forward and reverse logistics network design," *Journal of Cleaner Production*, vol. 208, pp. 1304–1316, 2019.

[62] Maxim A. Dulebenets, "A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping," *International Journal of Production Economics*, vol. 196, pp. 293–318, 2018.

[63] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization*.: Springer, 2005, pp. 105–145.

[64] Hui Li and Qingfu Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE transactions on evolutionary computation*, vol. 13, pp. 284–302, 2008.

[65] Nidamarthi Srinivas and Kalyanmoy Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, pp. 221–248, 1994.

[66] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, pp. 182–197, 2002.

[67] Eckart Zitzler, Marco Laumanns, and Lothar Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[68] Zeng Wang, Weidong Liu, Minglang Yang, and Dongji Han, "A multi-objective evolutionary algorithm model for product form design based on improved SPEA2," *Applied Sciences*, vol. 9, p. 2944, 2019.

[69] Fei He, Kang Shen, Li Guan, and Mingming Jiang, "Research on energy-saving scheduling of a forging stock charging furnace based on an improved SPEA2 algorithm," *Sustainability*, vol. 9, p. 2154, 2017.

[70] Xi Liu and Dan Zhang, "An Improved SPEA2 algorithm with local search for multi-objective investment decision-making," *Applied Sciences*, vol. 9, p. 1675, 2019.

[71] Praveen Kumar Tripathi, Sanghamitra Bandyopadhyay, and Sankar Kumar Pal, "Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients," *Information sciences*, vol. 177, pp. 5033–5049, 2007.

[72] Xin-She Yang, "Bat algorithm for multi-objective optimisation," *International Journal of Bio-Inspired Computation*, vol. 3, pp. 267–274, 2011.

[73] Chao Lu, Liang Gao, Xinyu Li, and Shengqiang Xiao, "A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry," *Engineering Applications of Artificial Intelligence*, vol. 57, pp. 61–79, 2017.

[74] Zikai Zhang, Qiuhua Tang, Dayong Han, and Zixiang Li, "Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment," *Neural Computing and Applications*, vol. 31, pp. 7501–7515,

2019.

[75] Xin-She Yang and Suash Deb, "Multiobjective cuckoo search for design optimization," *Computers & Operations Research*, vol. 40, pp. 1616–1624, 2013.

[76] Piotr Czyzżak and Adrezej Jaszkiewicz, "Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 7, pp. 34–47, 1998.

[77] J. Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu, "Workforce scheduling and routing problems: literature survey and computational study," *Annals of Operations Research*, vol. 239, pp. 39–67, 2016.

[78] Haneen Algethami, Rodrigo Lankaites Pinheiro, and Dario Landa-Silva, "A genetic algorithm for a workforce scheduling and routing problem," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 927–934.

[79] Philippe De Bruecker, Jeroen Beliën, Liesje De Boeck, Simon De Jaeger, and Erik Demeulemeester, "A model enhancement approach for optimizing the integrated shift scheduling and vehicle routing problem in waste collection," *European Journal of Operational Research*, vol. 266, pp. 278–290, 2018.

[80] Wasakorn Laesanklang, Dario Landa-Silva, and J. Arturo Castillo-Salazar, "Mixed integer programming with decomposition to solve a workforce scheduling and routing problem," 2015.

[81] Ashlea Bennett Milburn and Jessica Spicer, "Multi-objective home health nurse routing with remote monitoring devices," *International Journal of Planning and Scheduling*, vol. 1, pp. 242–263, 2013.

[82] Gözde Alp and Ali F. Alkaya, "Improving the Quality of Personnel Scheduling by Incorporating Fairness," *International Journal of Modeling and Optimization*, vol. 9,

pp. 97–101, 2019.

[83] Wai Yuen Szeto and Hong K. Lo, "Transportation network improvement and tolling strategies: the issue of intergeneration equity," *Transportation Research Part A: Policy and Practice*, vol. 40, pp. 227–243, 2006.

[84] Wooseung Jang, Huay H. Lim, Thomas J. Crowe, Gail Raskin, and Thomas E. Perkins, "The Missouri Lottery optimizes its scheduling and routing to improve efficiency and balance," *Interfaces*, vol. 36, pp. 302–313, 2006.

[85] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, pp. 671–680, 1983.

[86] Dindar Öz and Işıl Öz, "Scalable parallel implementation of migrating birds optimization for the multi-objective task allocation problem," *The Journal of Supercomputing*, vol. 77, pp. 2689–2712, 2021.

[87] Yuyan Han, Jun-Qing Li, Dunwei Gong, and Hongyan Sang, "Multi-objective migrating birds optimization algorithm for stochastic lot-streaming flow shop scheduling with blocking," *IEEE Access*, vol. 7, pp. 5946–5962, 2018.

[88] Tao Meng, Quan-Ke Pan, Jun-Qing Li, and Hong-Yan Sang, "An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem," *Swarm and Evolutionary Computation*, vol. 38, pp. 64–78, 2018.

[89] Aymen Sioud and Caroline Gagné, "Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times," *European Journal of Operational Research*, vol. 264, pp. 66–73, 2018.

[90] Ali Fuat Alkaya, Ramazan Algin, Yusuf Sahin, Mustafa Agaoglu, and Vural Aksakalli, "Performance of migrating birds optimization algorithm on continuous functions," in *International Conference in Swarm Intelligence*, 2014, pp. 452–459.

[91] Rainer E. Burkard, Stefan E. Karisch, and Franz Rendl, "QAPLIB–a quadratic assignment problem library," *Journal of Global optimization*, vol. 10, pp. 391–403, 1997.

[92] Juan A. Dıaz and Elena Fernández, "A tabu search heuristic for the generalized assignment problem," *European Journal of Operational Research*, vol. 132, pp. 22–38, 2001.

[93] Zhifeng Zhang, Yusheng Sun, Yadong Cui, and Haodong Zhu, "Modified Migrating Birds Optimization for Solving the Low-carbon Scheduling Problem," *International Journal of Grid and High Performance Computing (IJGHPC)*, vol. 12, pp. 63–75, 2020.

[94] Haitao Yuan, Jing Bi, MengChu Zhou, Jia Zhang, and Wei Zhang, "Profit-Maximized Task Offloading with Simulated-annealing-based Migrating Birds Optimization in Hybrid Cloud-Edge Systems," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 1218–1223.

[95] Sujun Zhang, Xingsheng Gu, and Funa Zhou, "An improved discrete migrating birds optimization algorithm for the no-wait flow shop scheduling problem," *IEEE Access*, vol. 8, pp. 99380–99392, 2020.

[96] Vahit Tongur, Ela Ertunc, and Mevlut Uyan, "Use of the Migrating Birds Optimization (MBO) Algorithm in solving land distribution problem," *Land Use Policy*, vol. 94, p. 104550, 2020.

[97] Biao and Pan, Quan-ke and Gao, Liang and Zhang, Xin-li and Peng, Kun-kun Zhang, "A multi-objective migrating birds optimization algorithm for the hybrid flowshop rescheduling problem," *Soft Computing*, vol. 23, no. 17, pp. 8101-8129, 2019.

[98] Mashael Maashi, Graham Kendall, and Ender Özcan, "Choice function based hyper-heuristics for multi-objective optimization," *Applied Soft Computing*, vol. 28, pp.

312–326, 2015.

[99] A. Charan Kumari, K. Srinivas, and M. P. Gupta, "Software module clustering using a hyper-heuristic based multi-objective genetic algorithm," in *2013 3rd IEEE International Advance Computing Conference (IACC)*, 2013, pp. 813–818.

[100] A. Charan Kumari and K. Srinivas, "Hyper-heuristic approach for multi-objective software module clustering," *Journal of Systems and Software*, vol. 117, pp. 384–401, 2016.

[101] Olacir R. Castro and Aurora Pozo, "A MOPSO based on hyper-heuristic to optimize many-objective problems," in *2014 IEEE Symposium on Swarm Intelligence*, 2014, pp. 1–8.

[102] Edmund K. Burke, J. Dario Landa Silva, and Eric Soubeiga, "Multi-objective hyper-heuristic approaches for space allocation and timetabling," in *Metaheuristics: Progress as Real Problem Solvers*.: Springer, 2005, pp. 129–158.

[103] Juan Carlos Gomez and Hugo Terashima-Marín, "Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems," *Genetic Programming and Evolvable Machines*, vol. 19, pp. 151–181, 2018.

[104] Fangfang Zhang, Yi Mei, and Mengjie Zhang, "Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 1366–1373.

[105] Rongwei Cui, Wei Han, Xichao Su, Yong Zhang, and Fang Guo, "A multi-objective hyper heuristic framework for integrated optimization of carrier-based aircraft flight deck operations scheduling and resource configuration," *Aerospace Science and Technology*, vol. 107, p. 106346, 2020.

[106] Yan Sun, Fuhong Lin, and Haitao Xu, "Multi-objective optimization of resource scheduling in Fog computing using an improved NSGA-II," *Wireless Personal Communications*, vol. 102, pp. 1369–1385, 2018.

[107] A. Sathya Sofia and P. GaneshKumar, "Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II," *Journal of Network and Systems Management*, vol. 26, pp. 463–485, 2018.

[108] Hongfeng Wang, Yaping Fu, Min Huang, George Q. Huang, and Junwei Wang, "A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 113, pp. 185–194, 2017.

[109] Maha Elarbi, Slim Bechikh, Abhishek Gupta, Lamjed Ben Said, and Yew-Soon Ong, "A new decomposition-based NSGA-II for many-objective optimization," *IEEE transactions on systems, man, and cybernetics: systems*, vol. 48, pp. 1191–1210, 2017.

[110] Zhibiao Zhao, Bin Liu, Chunran Zhang, and Haoran Liu, "An improved adaptive NSGA-II with multi-population algorithm," *Applied Intelligence*, vol. 49, pp. 569–580, 2019.

[111] Xingsi Xue, Jie Chen, Junfeng Chen, and Dongxu Chen, "A hybrid nsga-ii for matching biomedical ontology," in *International conference on intelligent information hiding and multimedia signal processing*, 2018, pp. 3–10.

[112] Zixiang Li, Mukund Nilakantan Janardhanan, and Qiuhua Tang, "Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots," *Neural Computing and Applications*, pp. 1–22, 2021.

[113] Wanida Laoraksakiat and Krisada Asawarungsaengkul, "Bi-objective Hybrid Flow Shop Scheduling with Family Setup Times Using Hybrid Genetic and Migrating Birds Optimization Algorithms," *Applied Science and Engineering Progress*, vol.

14, pp. 19–30, 2021.

[114] Nery Riquelme, Christian Von Lücken, and Benjamin Baran, "Performance metrics in multi-objective optimization," in *2015 Latin American Computing Conference (CLEI)*, 2015, pp. 1–11.

[115] Eckart Zitzler and Lothar Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, pp. 257–271, 1999.

[116] Shaolin Wang, Yi Mei, and Mengjie Zhang, "A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[117] Olacir R. Castro, Gian Mauricio Fritsche, and Aurora Pozo, "Evaluating selection methods on hyper-heuristic multi-objective particle swarm optimization," *Journal of Heuristics*, vol. 24, pp. 581–616, 2018.

[118] Giovani Guizzo, Silvia R. Vergilio, Aurora T. R. Pozo, and Gian M. Fritsche, "A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem," *Applied Soft Computing*, vol. 56, pp. 331–344, 2017.

[119] Gözde and Alkaya, Ali Fuat Alp, "An investigation of nature inspired algorithms on a particular vehicle routing problem in the presence of shift assignment", *Computers and Operations Research*, Accepted for publication, 2021.

[120] Ryoji Tanabe and Hisao Ishibuchi, "An easy-to-use real-world multi-objective optimization problem suite," *Applied Soft Computing*, vol. 89, p. 106078, 2020.

[121] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA,"

IEEE transactions on evolutionary computation, vol. 12, pp. 269–283, 2008.

[122] R. K. Suresh and K. M. Mohanasundaram, "Pareto archived simulated annealing for job shop scheduling with multiple objectives," *The International Journal of Advanced Manufacturing Technology*, vol. 29, pp. 184–196, 2006.

[123] Xinchao Zhao, Xiao-Shan Gao, and Ze-Chun Hu, "Evolutionary programming based on non-uniform mutation," *Applied mathematics and computation*, vol. 192, pp. 1–11, 2007.

[124] Tapas Si, N. D. Jana, and Jaya Sil, "Particle swarm optimization with adaptive polynomial mutation," in *2011 World Congress on Information and Communication Technologies*, 2011, pp. 143–147.

[125] Kalyanmoy Deb, Ram Bhushan Agrawal, and others, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, pp. 115–148, 1995.

[126] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the 9th annual conference on genetic and evolutionary computation*, 2007, pp. 1187–1194.

[127] Seyed Habib A. Rahmati, M. Zandieh, and M. Yazdani, "Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 64, pp. 915–932, 2013.

[128] Ryoji Tanabe. (2021) An Easy-to-use Real-world Multi-objective Optimization Problem Suite. [Online]. https://github.com/ryojitanabe/reproblems

[129] FY and Li, XS Cheng, "Generalized center method for multiobjective engineering optimization," *Engineering Optimization, Taylor & Francis*, vol. 31, no. 5, pp. 641-661, 1999.

[130] Hossain M and Hasegawa, Takashi Amir, "Nonlinear mixed-discrete structural optimization," *Journal of Structural Engineering*, vol. 115, no. 3, pp. 626-646, 1989.

[131] BK and Kramer, Steven N Kannan, "An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design," *Journal of Mechanical Design* , 1994.

[132] Jouni and Zelinka, Ivan Lampinen, "Mixed integer-discrete-continuous optimization by differential evolution," in *Proceedings of the 5th international conference on soft computing*, 1999, pp. 71-76.

[133] Guohua and Mallipeddi, Rammohan and Suganthan, Ponnuthurai Nagaratnam Wu, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.

[134] Rainer Storn, "Differential evolution design of an IIR-filter," in *Proceedings of IEEE international conference on evolutionary computation*, 1996, pp. 268–273.

[135] Gilbert Laporte, Yves Nobert, and Martin Desrochers, "Optimal routing under capacity and distance restrictions," *Operations Research*, vol. 33, pp. 1050–1073, 1985.

[136] Fred Glover and Claude McMillan, "The general employee scheduling problem. An integration of MS and AI," *Computers & operations research*, vol. 13, pp. 563–573, 1986.

[137] Augerat., CVRPLIB, 1995, http://vrp.atd-lab.inf.puc-rio.br/index.php/en/.

[138] Philippe Augerat. (1995) CVRPLIB. [Online]. Http://vrp.atd-lab.inf.puc-

rio.br/index.php/en/

RESUME

Name Surname : Gözde Alp

Foreign Languages : English

Education

Degree	Department	University	Graduation Year
Bachelor of Science	Computer Engineering Department	Süleyman Demirel University	2011
Master of Science	Computer Engineering Department	Dokuz Eylül University	2013

Work Experience

2008-2008, Cashier ,HMS-Host Great Lakes Travel Plaza, Cleveland, OHIO

2011-2012, Software Developer, Veri Bilişim Hizmetleri, İzmir

2012-2013, Research Assistant, Dokuz Eylül University, İzmir

2013-2019, Research Assistant, Marmara University, İstanbul

2019-... Research Assistant, Aydın Adnan Menderes University, Aydın

Scientific Studies

1. Alp, G., & Alkaya, A. F. (2021). "An investigation of nature inspired algorithms on a particular vehicle routing problem in the presence of shift assignment", Computers and Operations Research.
2. Alp, G., & Alkaya, A. F. (2021). Hyper Heuristic Based Migrating Birds Optimization Algorithm for a Fairness Oriented Shift Scheduling Problem. Mathematical Problems in Engineering, 9(2021).
3. Alp, G., & Alkaya, A. F. (2020). A Comparison of Swarm Intelligence Algorithms Exploiting a Novel Neighbour Generation Technique. 12th International Conference on Soft Computing and Pattern Recognition.

4. Alp, G., & Alkaya, A. F. (2019). Improving the Quality of Personnel Scheduling by Incorporating Fairness. *International Journal of Modeling and Optimization*, *9*(2).
5. Atay, C. E., & Alp, G. (2016). Modeling and querying multidimensional bitemporal data warehouses. *International Journal of Computer and Communication Engineering*, *5*(2), 110.
6. Alp, G., & Atay, C. E. (2016) ÇİFT ZAMANLI VE YARI YAPILI ÖZEL VERİ AMBARLARI İLE İLİŞKİSEL VERİ TABANI ÜZERİNE BİR KARŞILAŞTIRMA ÇALIŞMASI. *Trakya Üniversitesi Mühendislik Bilimleri Dergisi*, *17*(2), 57-6