Hyperheuristics in Logistics

Kassem Danach

N^o d'ordre: 315

Centrale Lille

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

en

Automatique, Génie Informatique, Traitement du Signal et des Images

par

Kassem Danach

DOCTORAT DELIVRE PAR CENTRALE LILLE

Hyperheuristiques pour des problèmes d'optimisation en logistique

Hyperheuristics in Logistics

Soutenue le 21 decembre 2016 devant le jury d'examen:

President:	Pr. Laetitia Jourdan	Université de Lille 1, France
Rapporteurs:	Pr. Adnan Yassine	Université du Havre, France
	Dr. Reza Abdi	University of Bradford, United Kingdom
Examinateurs:	Pr. Saïd Hanafi	Université de Valenciennes, France
	Dr. Abbas Tarhini	Lebanese American University, Lebanon
	Dr. Rahimeh Neamatian Monemin	University Road, United Kingdom
Directeur de thèse:	Pr. Frédéric Semet	Ecole Centrale de Lille, France
Co-encadrant:	Dr. Shahin Gelareh	Université de l' Artois, France
Invited Professor:	Dr. Wissam Khalil	Université Libanais, Lebanon

Thèse préparée dans le Laboratoire CRYStAL

École Doctorale SPI 072 (EC Lille)

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Frédéric Semet, Dr. Shahin Gelareh and Dr. Wissam Khalil for their continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

I would also like to thank my parents for their wise counsel and sympathetic ear.

I would like express appreciation to my beloved wife Jomana Al-haj Hasan who spent sleepless nights with and was always my support in the moments when there was no one to answer my queries.

Finally, there are my children, who have given me much happiness and keep me hopping. My son, Mahdi, has grown up watching me study and juggle with family and work. Abbass, the little one, who always try to do everything to make his presence felt. I hope I have been a good father and that I have not lost too much during the tenure of my study.

Resumé

Le succès dans l'utilisation de méthodes exactes dans l'optimisation combinatoire à grande échelle est encore limité à certains problèmes, et peut-être des classes spécifiques d'instances de problèmes. Le moyen alternatif consiste soit à utiliser des métaheuristiques ou des mathématiques qui utilisent des méthodes exactes à certains égards. Le concept d'hyperheuristique (HH) est une généralisation de celle des métaheuristiques. Dans le contexte de l'optimisation combinatoire, nous nous intéressons à l'heuristique pour choisir l'heuristique. Les deux catégories hyperheuristiques principales de la classification précédente sont: 1) la sélection heuristique, qui considère une méthode pour sélectionner des heuristiques à partir d'un ensemble d'heuristiques existantes, et 2) la génération heuristique qui consiste à générer de nouvelles heuristiques à partir des composantes des heuristiques existantes.

Dans cette thèse, nous nous concentrons sur l'optimisation hyperheuristique des problèmes logistiques. Nous présentons une revue de littérature détaillée sur les termes origine, concept, domaine d'application, etc. Ensuite, deux problèmes logistiques ont été choisis pour lesquels nous avons proposé HH. Sur la base de la structure générale d'une solution réalisable et en exploitant les informations cachées dans les données d'entrée, nous définissons un ensemble d'heuristiques possibles pour chaque problème. Nous nous concentrons ensuite sur la proposition d'un cadre hyperheuristique qui effectue une recherche dans l'espace des algorithmes heuristiques et apprend comment changer l'heuristique en place d'une manière systématique le long du processus de telle sorte qu'une bonne séquence

d'heuristiques produit des solutions de haute qualité. Notre cadre hyperheuristique est équipé d'un mécanisme d'apprentissage qui apprend l'environnement et guide la transition d'une heuristique historique à une autre jusqu'à ce que l'algorithme global se termine.

Le premier problème abordé est le problème d'ordonnancement de la plate-forme de workover (WRSP), qui consiste à trouver le meilleur horaire pour un certain nombre de plates-formes de workover pour minimiser la perte de production, associée à un grand nombre de puits en attente de maintenance. Un algorithme d'hyperheuristique de sélection est proposé, qui est guidé par un mécanisme d'apprentissage conduisant à un choix approprié de mouvements dans l'espace des heuristiques qui sont appliquées pour résoudre le problème. Nos expériences numériques sont menées sur des exemples d'une étude de cas de Petrobras, la Société nationale brésilienne du pétrole, et ont été comparées avec une méthode exacte prouvant son efficacité.

Le deuxième problème est une variante du problème de routage d'emplacement de concentrateur, qui cherche à diviser les nœuds en moyeux et rayons. Deux HH différentes ont été appliquées à deux variantes de ce problème, qui sont dérivées d'un problème d'acheminement de localisation de concentrateur de capacité d'allocation unique et diffèrent principalement dans la définition de la capacité. Dans le premier, le nombre de rayons pouvant être attribué à chaque hub est limité; Tandis que dans la seconde, c'est le volume de l'écoulement circulant sur la voie du rayon-niveau. De plus, cinq relaxations lagrangiennes (LR) ont été proposées pour le premier problème afin d'utiliser certains résultats pendant le processus de

HH. Les résultats de calcul prouvent l'efficacité de HH et la pertinence d'inclure l'information LR. Enfin, nous comparons les performances de plusieurs HH proposées dans la littérature pour le problème précédemment abordé, avec différentes méthodes de sélection heuristique telles que la sélection aléatoire, la fonction de choix, Q-Learning et la colonie de fourmis.

Contents

List of Tables

List of Figures

Chapter 1

Hyperheuristic: A General Overview

1.1 Introduction

The term *Optimization*, according to Merriam Webster[1], is a process or methodology of making something as fully perfect, or effective as possible. We seek a state from among the set of possible states with the most cost effective or highest achievable performance under given constraints. In mathematical optimization with one single objective, we search a maximum or a minimum value of an objective function of several variables over a set of feasible solutions defined by a set of constraints. When both the objective function and the search space could be described by means of linear terms and linear system, respectively, linear programming based

[1] www.merriam-webster.com/dictionary/mW

techniques are used to solve such models and find the optimum (optima).

A combinatorial problem (COP) is defined as a problem of finding an optimal solution from a finite set or possibly countable infinite set of searching area (Schrijver, 2005). Usually, COPs involve grouping, ordering, or assignment of a discrete finite set of objects with some predefined conditions. Many problems in the real-life from telecommunications, transportation, logistics and energy sectors to modeling of social networks, bio-informatics and even archeology and beyond are modeled as COPs. Some classical and well-studied problems of COPs that often appear as subproblem in other more complex problems includes: knapsack problems (Martello and Toth, 1990), traveling salesman problems (Applegate et al., 2011), vehicle routing problems (Toth and Vigo, 2014) and variants of facility location problems (Drezner and Hamacher, 2001) among many others.

In this thesis, we focus on a couple of problems arising in logistics and network design for management of the flow of resources between the point of origin and the point of consumption in order to meet some requirements. Such problems, like many other problems in logistics, are categorized as COPs and they also include continuous decision variables. These problems are often modeled as mixed integer

programming (MIP) models which can be expressed as follows:

$$\begin{aligned} \text{Objective :} \quad & \min c^T x + d^T y & \\ \text{Constraints :} \quad & Ax + By = b & \text{linear constraints} \\ & x \in \mathbb{R}^{m_1} l \leq x \leq u & \text{bound constraints} \\ & y \in \mathbb{Z}^{m_2} & \text{integrality constraints} \end{aligned}$$

The optimization techniques proposed in the literature, are classified in two main categories: exact method and non-exact method(variants of heuristic approaches). Exact optimization methods provide provably optimal solutions, when exists, while the non-exact methods (which might also be based on MIP models and mathematical programming). Yet, there is a serious bottleneck when dealing with exact methods including general-purpose solvers: Small instances of real-life problems can be often solved to optimality in reasonable times when a good description of polytope (in terms of tightness of LP relaxation, additional valid inequalities are available etc) is available while moderate size instances may be solved using efficient and specialized methods in reasonable times. However, the larger ones are very less likely to be solved to optimality and often even finding a feasible solution for such problems may become very difficult task for exact approaches. At this point, non-exact methods such as those combining heuristic techniques with mathematical programming techniques such as Lagrangian heuristics, LP-based heuristic and primal-dual heuristic. or the different approaches such as specialize ad-hoc heuristics, evolutionary or metaheuristic approaches that are developed to

find very good feasible solutions become inevitable. Sometimes in exact methods, an indication of optimality can be provided, the non-exact often do not provide any indication.

In the remainder of this chapter, we elaborate further on heuristic optimization and explain some differences among different techniques and paradigms. We then focus on the context of hyperheuristic, the paradigm, general framework, steps, and different components.

1.2 Heuristics, Metaheuristics and Matheuristics

Heuristics are often problem-dependent techniques that are usually adapted to the problem at hand and they try to take full advantage of the particularities (structure or other kind of information) of the problem. However, since they are often too biased, they usually get trapped in a local optimum and thus fail, in general, to obtain the global optimum solution, and no guarantee if it get the optimal solution or not.

Metaheuristic is a class of approximate methods developed in the early 1980s. Metaheuristics are designed to solve complex optimization problems; in fact, the classical heuristics were not always effective and efficient, as they were time consuming or there were some limitation to help them escape from a local optima.

Metaheuristics are problem-independent techniques, pattern or paradigms. For example Simulated Annealing (SA) (Kirkpatrick, 1984) imitates the cooling of material in a heat bath and maps the objective function optimal of the optimization problem on the optimal energy level function of the hot material such that annealing is the process in which the temperature of a molten substance is slowly reduced until the material crystallises to give a large single crystal.

Metaheuristic, can be defined as an iterative generation of a process, which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space (Osman and Laporte, 1996).

As such, the metaheuristics are independent of any specificity of the problem and, therefore, can be used as general paradigm. In general, they are less greedy approaches in characteristic. In fact, they may even accept a temporary deterioration of the solution (e.g. SA), which allows them to explore more thoroughly the solution space and thus to get a hopefully better solution (that sometimes will coincide with the global optimum). Although a metaheuristic is a problem-independent technique, it is nonetheless necessary to do some fine-tuning of its associated parameters in order to adapt the technique to the problem at hand.

To illustrate the concept of metaheuristic, we should first distinguish between heuristic algorithms and metaheuristic algorithm. The main objective of heuristics is to obtaining a solution at an acceptable cost for a wide range of problems. Heuristics do not have an approximation guarantee on the quality of the obtained

solutions. On the other hand, Metaheuristics are general-purpose algorithms that can be regarded as upper level general methodologies employed as guiding strategy for the heuristics.

In metaheuristics approach, two strategies must be taken in account (Talbi, 2009):

1. *Diversification*: that explores the search space to avoid getting stuck in the same or similar areas of feasible space.

2. *Intensification*: that emphasizes on concentrating search in the promising regions previously found, in order to exploiting the potentials.

Several families of metaheuristic algorithms are proposed in the literature such as simulated annealing (SA) (Aarts and Korst, 1988), tabu search (TS) (Glover, 1989), genetic algorithm (GA) (Davis, 1991), GRASP (Dyall et al., 1989), variable neighborhood search (VNS) (Mladenović and Hansen, 1997a), adaptive large neighborhood search (ALNS) (Ahuja et al., 2000), path relinking (Glover, 1998), particle swarm optimization (PSO) (Eberhart et al., 1995), scatter search (SS) (Glover, 1998), ant colony (Dorigo et al., 2000), among others.

Metaheuristic algorithms can be categorized with respect to different characteristics as follows. The *deterministic* ones offer the same results for several runs while the *non-deterministic* ones may report different results per different invocations.

One can distinguishes among metaheuristic algorithms by the algorithm that use single solution (*trajectory methods*) at each iteration of the search such as local search, SA, TS, VNS, ALNS, GRASP, path relinking, etc, and those which use multiple solutions (*population-based methods*) and contain more than one (partial) solution such as GA, Ant colony, PSO, scatter search, etc.

One can also distinguishes between the metaheuristics that make use of some kind of long-term or short term memories and the memory-less ones. Memory based methods are the methods where the search moves are recorded and the future moves should use those information to avoid cycling, previous solution, worse solution etc. such as Tabu Search, PSO, Scatter, Path relinking, Genetic Algorithm, Ant Colony, etc., while others such as GRASP, VNS, etc. are originally memory-less.

Matheuristic optimization (Bartolini and Mingozzi, 2009) algorithms are a combination between (meta)heuristic approaches and different techniques of mathematical programming. Whether mathematical programming is in charge of solving the problem at hand and (meta)heuristics exploit the primal/dual information to generate better primal bounds or mathematical programming techniques are employed within (meta)heuristics to solve a sub-problem after some variable being fixed (for example a capacitated flow problem, or a linear part of an initially non-linear model where part of variables are fixed by the (meta)heuristic) we are

talking about the matheuristics. Such techniques become very popular as MIP solvers or customized MIP codes have become more effective as primary solvers or as sub-procedures which is due to the advancements that were achieved in the research on mathematical programming, and in particular on discrete optimization.

1.3 Hyperheuristics

Hyperheuristics (HH) aims to solve hard computational search problems by automating the design of heuristic methods. The term itself appeared for the first time in 1997, in a study about automated theorem proving (Denzinger et al., 1996). It was given to a protocol that combines several artificial intelligence methods. It was then more adequately used in the year 2000 in connection with combinatorial optimization (Cowling et al., 2001): *heuristics to choose heuristics*. In this context, a hyperheuristic is a high-level approach that solves hard computational search problems, given a particular problem instance and a number of low-level heuristics, by selecting and applying an appropriate low-level heuristic at each decision point. Even though the term is relatively new, the idea itself dates back to the 1960s (Crowston et al., 1963). A number of researchers developed the automation of the design process of heuristic methods during the 1990s (Fang et al., 1993).
Hyperheuristics however go a step beyond metaheuristics. The particularity of hyperheuristics is that their search space is not the usual space of the solutions but is rather the space of heuristics or metaheuristics.

The concept of *Hyperheuristic* may be seen a generalization of that of (Meta)-heuristics and facilitates classifying a large body of literature of heuristics and metaheuristics that was rather difficult to classify before this. Many definition of the term was found in the literature. Özcan et al. (2010) defines hyperheuristics as methodologies for searching the feasible space, which is generated by a set of low level heuristics while Topcuoglu et al. (2014) defines it as methods for automation of the process of selecting and generating multiple low-level heuristics.

When we are dealing with certain variants of optimization problems, hyperheuristics are most useful since they only require general knowledge of the problem domain. Cowling et al. (2001) defined the term hyperheuristic as *heuristics to choose heuristics*. Ochoa et al. (2009) stated that in hyperheuristics research, the importance of increasing the efficiency of the solution method is more important than bettering the solution itself. They also stated that what distinguishes metaheuristics from hyperheuristics is that the latter has a search space of heuristics not of problem solutions, which was previously mentioned by (Ross, 2005).
Recently, the definition of hyperheuristics has been recently extended to refer to the *search method or learning mechanism to select or generate heuristic to solve combinatorial problem*. The main objective is to design a generic method which can apply in several problem domains. This method should produce a solution of high quality but the emphasis, in this method, is on the use of low level heuristics that are easy to implement.
The hyperheuristic framework requires a set of predefined materials. The hyper-

heuristic framework is provided with a set of preexisting simple heuristics, and the challenge is to select or generate the heuristic or the operator that is somehow the most suitable for the current problem state. This process continues until the termination criteria is met.Figure 1.1 depicts a generic hyperheuristic process, which represents the strategy plan in the high level and the major components such as the termination criteria, Low-level heuristics (LLH)/operator selection methods, the acceptance criteria etc. Until the termination criteria is met, HH selects the corresponding LLH (or operator) from an *a priori* known set according a predefined selection rule and apply it subsequently. The resulted solution at each iteration may or may not be accepted which depends on the predefined acceptance criteria.

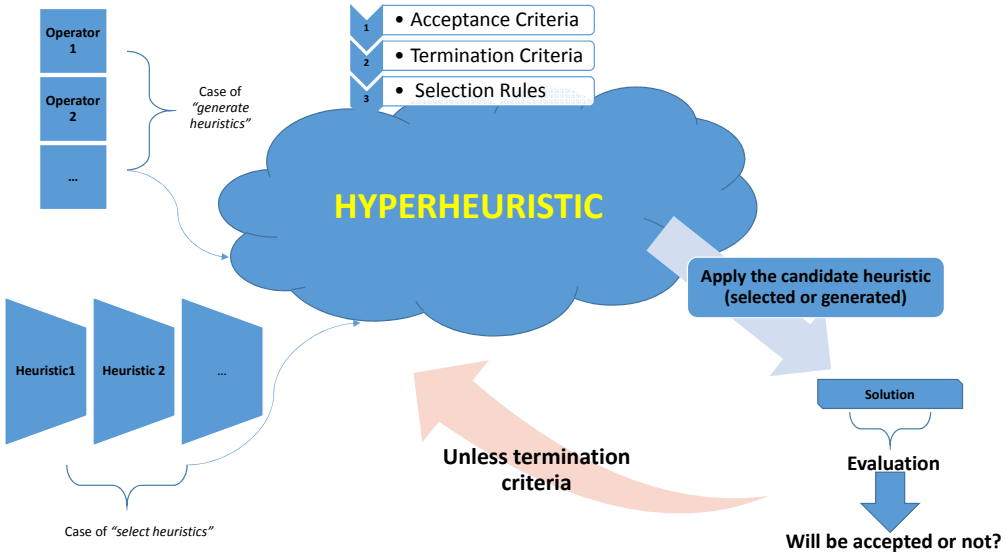

Figure 1.1: The general scheme of hyperheuristic framework.

1.3.1 Hyperheuristic Classification

A variety of hyperheuristic approaches was defined in the literature. In this subsection, we present an overview of the different categorizations of hyperheuristic with respect to I) the nature of heuristic search space, II) if it is constructing a solution from scratch or starting from a feasible solution in order to improve it, and III) and the use of learning mechanisms during the search process. Interested readers are

referred to see (Burke et al., 2010, 2009a; Bader-El-Den and Poli, 2008) for further details.

Nature of heuristic search space: While in the beginning the hyperheuristics were to uses (select from among) the pool of existing low-level heuristics, in the late of 2000's genetic programming started being used for hyperheuristics. This has introduced a new way in which the low-level heuristics were not only *selected* but also being *generated from existing heuristic components*. Such an emerging perspective has led to a revised definition: *as an automated methodology to select or generate heuristic in order to solve hard problems* (Burke et al., 2010). Hence, the HH are now belong to either heuristic selection category or the heuristic generation ones. We are not aware of any work that try to propose any hybrid method.

Akin to the heuristic selection HH class of hyperheuristics, the other category requires a set of heuristics that are suitable to the problem at hand. The difference is that these heuristics will not be supplied directly to the framework, they will be fragmented into their basic building-blocks so that new heuristics can be generated from them, all in the hopes of achieving a good feasible solution. This class of hyperheuristics led researchers to distinguish between *reusable* and *disposable* heuristics. The latter refers to heuristics that were built to be used on a specific problem and cannot be used on others.

Solution Target: construction vs. perturbation: If no initial feasible solution is at hand we talk about the Selection based *construction* hyperheuristics which aim at select and use construction heuristics in a smart and efficient way that it builds a good solution from the scratch progressively. It starts with an empty solution and uses the *hopefully* most suitable heuristic, from a pool of pre-existing problem specific construction heuristics, to solve the current problem instance. This process terminates when it reaches the complete feasible solution. This highlights the importance of choosing the best heuristic at each problem instance, because the sequence is not infinite.

On the other hand, when a feasible solution is indeed at hand, one talks about the selection based *perturbation* hyperheuristics that require a complete solution to start with. This solution could be generated randomly or using a construction heuristic (not directly related to the HH). Afterwards, the higher-level selection hyperheuristic is supplied with a pool of neighborhood structures and/or simple local search routines. It selects and applies those heuristics to the starting solution in an iterative manner until a stopping condition has been met, unlike the constructive class where the process is ended after a certain number of choices.

Selection Methods: with learning or without: Researchers also classified hyperheuristics based on whether they include learning mechanisms or not (Soubeiga, 2003). Hyperheuristics with learning are methods that record the historical performance of the heuristics available in the search pool, and use learning mechanisms to

study those records and manage the selection process of the heuristics based on it. The Hyperheuristics without learning, select the heuristics from the search pool in a predetermined manner. In Chapter 4, we will elaborate further on different selection methods with and without learning.

The 2-dimension hyperheuristic classification by (Burke et al., 2010), shown in Figure 1.2, presents a classification of hyperheuristics according the nature of heuristic search space and the source of feedback during the process. The first dimension distinguishes between heuristic selection and heuristic generation, which could both be constructive or local search hyperheuristics using construction and perturbation low-level heuristics, respectively. The second dimension however, points out towards another classification, which is orthogonal to the first one. It distinguishes between new categories according to the source of feedback during learning.

The first class is referred to as *online learning* hyperheuristics, where the learning mechanism is active during the solution process of an instance of the problem. The higher-level strategy selects or generates the appropriate low-level heuristic according to their real-time performances.

The second class are the *off-line learning* hyperheuristics, which use a set of training instances in order to extract hidden knowledge in the form of programs or rules, all in the hopes of generalizing the process to the stage where it becomes

fully capable of solving unseen instances.

Apart from the aforementioned classes, there are *non-learning* hyperheuristics that are the one that do not employ any learning mechanisms of any sort.

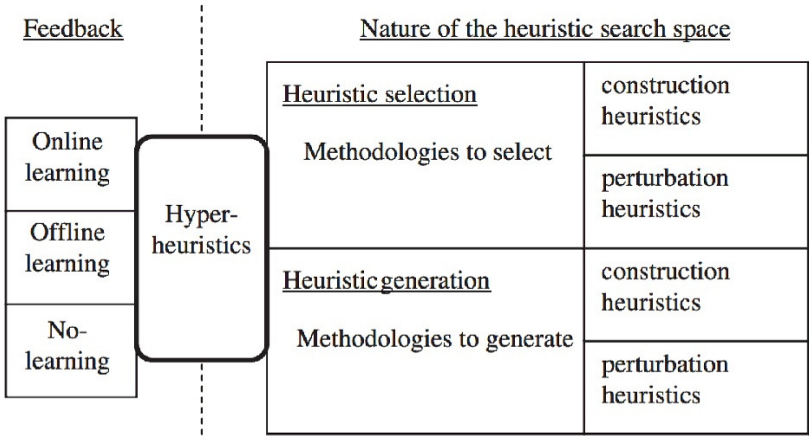

Figure 1.2: A classification of hyperheuristic approaches (Burke et al., 2010)

.

There are also other classification schemes such as the one in (Chakhlevitch and Cowling, 2008a) that will be more elaborated in chapter 4.

1.3.2 Move Acceptance Criteria

The decision of whether or not accepting a new solution during the search, which known as *move acceptance strategy*, is regarded as a crucially important decision in

the recent studies such the selection methods. The acceptance strategies themselves are divided into two categories: Deterministic and non-deterministic.

Deterministic methods make the same acceptance decision for the same candidate situations and no random elements exists in this category. Few deterministic methods, which are used in (Cowling et al., 2001, 2002a), are summarized in the following:

i) *All Moves(AM)*: deals with accepting the candidate solution regardless of its quality.

ii) *Only Improvements (OI)*: accepts only the improving solutions.

iii) *Improving and Equal (IE)*: the same of *Only Improvements* except that accepting candidates with the same quality as of the current but significantly different from the incumbent, as a kind of diversification.

Cowling et al. (2002a) present a variety of hyperheuristic in order to compare the performance of different combinations of heuristics selection methods and acceptance strategies. Two acceptance methods were tested: AM and OI. The choice function hyperheuristic with AM acceptance criteria is shown to be the better one in their experimental results. To the best of our knowledge the AM is only considered in (Nareyek, 2004) as the acceptance criteria with reinforcement learning technique.

On the other hand, non-deterministic methods might make different decision for

the same input at different decision point. Therefore, non-deterministic methods require additional parameter such as the time stamp. Several non-deterministic acceptance strategies are proposed in the literature, some of which are listed in the sequel:

1. *Exponential Monte Carlo (MC)* (Ayob and Kendall, 2003a), allows the non-improving solution to be accepted with a probability $e^{-\delta}$, where δ is the difference between the objective function value of the current solution and that of the candidate. The probability of a non-improving solution to be accepted decreases as the δ increases.

2. *Simulated Annealing (SA)* (Bai and Kendall, 2005), accepts any worsening solution with a probability of $e^{-\delta/t}$. The probability of a non-improving solution being accepted decreases as δ/t increases, where t is the time stamp.

3. *Exponential Monte Carlo With Counter (EMCQ)* (Ayob and Kendall, 2003a), is a variant of simulated annealing, but the probability of accepting a candidate solution follows the equation $\exp(\frac{-\theta}{\tau})$, where $\theta = \delta t$, $\tau = p(Q)$, t is the time. θ and τ were defined in a manner to ensure that the probability of accepting worsening solution decrease as t increase and δ decrease. Q is a parameter which it is represented as a counter of consecutive non-improved iterations. With the smaller values of Q the probability of accepting non-improving solution increases which ensures some kind of diversification.

4. *Record to Record Travel (RRT)*, (Dueck, 1993), which allows worsening

solutions to be accepted with respect to a threshold value, which depends on the objective function deviation from the current best solution.

5. *Great Deluge*, (Dueck, 1993), which is similar to the RRT while the threshold τ, presented in the following, decreases in time linearly. *maxIter* represents the maximum number of iterations or the time limit, t is the elapsed time or iterations, and ΔR is an expected range between the initial objective function value and the best one.

$$\mathcal{T} = f_{opt} + \Delta R(1 - \frac{t}{maxIter}) \tag{1.1}$$

6. *Naive Acceptance*, (Cowling et al., 2001), where non-improving solutions are accepted with a probability of 0.5.

7. *Adaptive Acceptance*, which authorizes worsening moves being accepted with a probability of $1 - \frac{1}{C},\ C > 0$. C is considered as a counter, which increases to N consecutive operation without improving the solution. C is reset to 1 each time an improvement in the solution is observed.

8. *Late Acceptance (LA)*, is a generic optimisation method (Burke and Bykov, 2008) which is an extension of simple hill-climbing and requires a unique parameter with a memory based approach. Such in hill-climbing, accepted new solution should be better than the incumbent one, LA accepts a solution if it is of better quality than the solution n iterations previously, where n is the size of a memory of previously seen solutions.

The non-deterministic acceptance criteria have received a lot of attentions so far. Ayob and Kendall (2003b) proposed three move acceptance strategies based on Monte Carlo move acceptance method, which are based on the change of the fitness value, time and the number of consecutive non-improving moves. The heuristic selection method was a simple random method, forms with one of the three move acceptance methods a hyperheuristic approach, which aims to solve the component placement problem. Authors demonstrate that the best move acceptance strategies for this problem within the used heuristic selection method, is exponential Monte Carlo with counter method.

Other SA-based acceptance method were also applied in the literature. Bai and Kendall (2005) demonstrates good performance of a simple random hyperheuristic with SA acceptance criteria in the problem of shelf space allocation. Reheating schema was embedded into the SA move acceptance, and applied to the problem of travelling tournament problem in (Anagnostopoulos et al., 2006). Moreover, a hyperheuristic, consists of SA with reheating and a reinforcement learning method, was applied in (Bai et al., 2007) for nurse roistering, course timetabling and 1D bin packing problem. It must be noted that SA differs from EMCQ by using a cooling schedule.

In (Dueck, 1993), simple random heuristic selection was combined with four acceptance strategies including monte carlo, AM, OI and RRT, in order to compare their performances. RRT move acceptance provides the better solution among the three others.

Another performance study for several heuristic selection methods and acceptance strategies was employed by (Bilgin et al., 2007) for the problem of exam timetabling. The hyperheuristic comprised of a choice function heuristic selection method and SA acceptance mechanism, is demonstrated as the best method. While, simple random with great deluge hyperheuristic has the second place with respect to the performance.

The great deluge, which is a threshold based acceptance method, is proposed by (Dueck, 1993). It was applied by (Kendall and Mohamad, 2004) with simple random selection method to a mobile telecommunication network problem. The great deluge with greedy heuristic selection was also applied in order to solve a problem of job shop scheduling in (Mcmullan, 2007).

Three variants of great deluge move acceptance strategies with a reinforcement learning was presented in (Sin and Kham, 2012) in order to solve a problem of exam timetabling problem.

Özcan et al. (2009) applied late acceptance strategy with several heuristic selection method for solving exam timetabling problem. Simple random selection method, which select one of four perturbation low-level heuristics, with late acceptance method is shown as the best proposed approach compared to the performance of using tabu search or choice function selection methods.

1.3.3 Termination Criteria

Termination criteria deal with defining a special condition(s) as search stopping criteria. The termination condition tries to avoid useless computations and also

avoid early termination. The most known proposed termination criteria include time limit, number of iterations, number of non-improvement steps, performance changes of each existing heuristic etc.

The different (combination of) termination criteria in different contexts were proposed in literature, e.g. (Özcan et al., 2010), (Burke et al., 2008a), (Raghavjee and Pillay, 2015), (Shmygelska and Hoos, 2005) and (GiriRajkumar et al., 2010) among others.

1.4 State of the art

The hyperheuristic approach has been successfully applied to solve many combinatorial problems in Scheduling (see e.g. (Ahmed et al., 2015; Zheng et al., 2015; Aron et al., 2015), Routing (Danach et al., 2015a,b; Monemi et al., 2015; Marshall et al., 2014; Garrido and Riff, 2010; Garrido and Castro, 2012)), Bin Packing (see (Sim et al., 2015; Beyaz et al., 2015; Ross et al., 2002)), Telecommunications (see (Kendall and Mohamad, 2004; Keles et al., 2010; Segura et al., 2011)) and Constraint Satisfaction Terashima-Marín et al. (2008); Ortiz-Bayliss et al. (2010) from among others. The interested readers are referred to (Chakhlevitch and Cowling, 2008b) for a classification and review of the recent developments in hyperheuristics including real-world applications. The authors identified three distinct attributes that define a hyperheuristic method. According to the authors, a hyperheuristic is 1) a high level heuristic that manages lower level ones, 2) its goal is to find a

good solution method instead of finding a good solution, and 3) it makes use of problem-specific information uniquely. The authors emphasize on the importance of the last attribute.

Burke et al. (2009a) elaborates on some of the possible methodologies that use sets of promising heuristic components to generate new heuristics. These methodologies are highly influenced by Genetic Programming techniques. Many points were highlighted in this chapter, the authors described the steps to properly apply this approach along with some case studies. Furthermore, some of the issues faced by this type of HH are discussed and a brief literature review is presented. Kendall et al. (2002a) presents an approach to solve three personnel scheduling problems by ranking heuristics using a performance-rating function. Burke et al. (2003) combined Tabu-search and HH has been proposed as a hybrid method that was then applied on eleven university course timetabling problems and on variants of a nurse scheduling problem. Burke et al. (2002, 2006); Petrovic and Qu (2002) reported the effectiveness of case based reasoning when employed in timetabling problems as a heuristic selection methodology.

The following approaches are based on the work by Ross et al. (2002) on uni-dimensional bin packing and are categorized as evolutionary approaches in generating HH for solving the 2D-Regular Cutting Stock Problems: in (Terashima-Marín et al., 2005a), HH was generated by using the XCS Classifier System; in (Terashima-Marín et al., 2005b), the authors made use of a GA with integer and fixed-length representation in order to produce HH. The results achieved by both of

these HH were significantly superior to the ones delivered by ordinary heuristics, proving efficiency of HH for many different problem instances.

In (Ross et al., 2002), the authors focused on learning methods that could be applied to many problem instances rather than learning and improving individual solutions. This method selects a heuristic at each problem state, in order to solve the problem. The selected heuristic is the most appropriate one and has the biggest chance of solving that specific state. The method proceeds progressively in this manner; selecting different heuristics for different states of problems, until eventually, the problem reaches a solved state. This is consistent with the attributes that make every HH (Chakhlevitch and Cowling, 2008b).

In (Wilson et al., 1998), an accuracy-based Learning Classifier System (XCS) was employed to learn a set of rules that allows the method to associate characteristics of the current problem state with eight different heuristics. This application on the one-dimensional bin packing problem was the first attempt at using such HH model.

In (Schulenburg et al., 2002), improvements were made over the initial method. A new heuristic that randomly selects heuristics was introduced to the method in order to compare results. The HH method gave results that greatly supersede those achieved by the proposed heuristics. During the learning process of HH, two different reward systems were applied to the process, creating and evolving individual processes. This work gave much importance to those individual processes

and focused mainly on them. The authors continued to analyze and compare the performance of the HH method to that of single heuristics.

The promising results led to further examine the idea on the other problem domains. A HH method was proposed by (Cowling et al., 2001, 2002a,b,c) with an even higher level of abstraction than that of metaheuristic local search methods. It selects different neighborhoods according to a choice function of which the goal is to determine the most appropriate neighborhood for the current problem. Previously, several time consuming trial and error experimentations were carried out to select the correct neighborhoods. This in turn, highlights the importance of this HH method in reducing the computational time, and perform results quality. Cowling et al. (2001, 2002a,b,c) also stressed the importance of choice functions, and their integral role in the success of HH.

Burke and Newall (2002) proposed a HH approach aiming at improving an initial heuristic ordering in examination timetabling problems. The adaptive heuristic functions works as follows: First, it schedules the exams in an order specified by the original heuristic in order to create an initial solution. An exam is prompted up the order in a later construction in the case where the use of this ordering hinders the process of acceptably scheduling an exam. The termination criteria of this process are either achieving the goal, which is effectively ordering the exams in a manner that they all can be acceptably scheduled, or reaching a pre-determined time limit. The process will continue until at least one of them is met. The results obtained from the experiments prove that the quality of the solution method

achieved by this process is greatly better than the quality achieved by the original heuristic. According to the authors, the method can still find acceptable results relatively quickly even if the initial heuristic was poor.

Cowling et al. (2002c) studied a *trainer scheduling* using a genetic algorithm based hyperheuristic (hyper-GA) in order to schedule several geographically distributed training staff and courses. The aim of the hyper-GA is to evolve a good-quality heuristic for each given instance of the problem and use this to find a solution by applying a suitable ordering from a set of low-level heuristics.

Since the user only supplies a number of low-level heuristics and an objective function, the proposed hyperheuristic can be re-implemented for a different type of problem. The method's results appear much better than those of conventional genetic and memetic algorithm methods, and it is expected to be robust across a wide range of problem instances.

1.5 Existing and the Proposed Framework

The hyperheuristic system is consist of two levels separated by the domain barrier: Hyper Level and Base Level. A set of predefined heuristics, a specific fitness function and search space was encapsulated in the base level (Swan et al., 2013).

The main decision of the hyper level is to decide which base level heuristics must solve the defined problem (Ryser-Welch and Miller, 2014). The hyperheuristic architecture represented by the two level concept, answers not only the question

about the degree of generality, but also paves the way of plug and play concept in heuristic domain. Many hyperheuristic frameworks were proposed in the literature such as in the following:

1. *SATzilla*: is proposed by (Nudelman et al., 2004) for SAT solvers, is focused on the concept of algorithm portfolio, which aims at predicting the executing time of algorithms in order to solve the problem with a reduced time. It uses an off-line learning in order to develops heuristics portfolios. A Matlab code of this framework has been developed.

2. *Single Neighborhood-based Algorithm Portfolio in Python* (Snappy): adopts the algorithm portfolio. The main different issue with SATzilla is using on-line learning methods in order to improve its own performances (Samulowitz et al., 2013).

3. *Hyflex*: is a library proposed by (Burke et al., 2009b), implemented in java, which includes a set of methods, the communication protocols between the solver and the problem domain. It is an efficient tools to build new cross-domain hyperheuristic.

4. *parHyFlex*: is a parallel implementation of Hyflex framework allowing to run a hyperheuristic in a parallel setting (Van Onsem and Demoen, 2013).

5. *Generic Intelligent Hyperheuristic (GIHH)*: is an improved version of Hyflex, proposed as a generic framework for online selective hyperheuristic, which equipped with several online learning methods (Misir, 2012).

Our proposed heuristic selection hyperheuristic framework, shown in Figure 1.3, ic comprised of two main classes: `HyperLevel` and `BaseLevel`. The first one represents the hyperheuristic process methods and attributes, while the second one represents the problem properties.

Heuristics, which are predefined for the given problem, are categorized by type using `HeuristicTypes` abstract class. Hyperheuristic can use any acceptance criteria in the set of All Moves, Only Improvements, Improving and Equal, Exponential Monte Carlo (MC), Simulated Annealing (SA), Exponential Monte Carlo With Counter (EMCQ), Record to Record Travel, Naive Acceptance, Late Acceptance etc. In addition, concerning the termination criteria, three possible methods are defined which are the time limit, a defined number of non improvement consecutive iterations and termination based on the heuristics performance. Finally, the hyperheuristic system can opt for any selection method among the aforementioned ones (with/without learning). We elaborate further on the selection methods in 4.

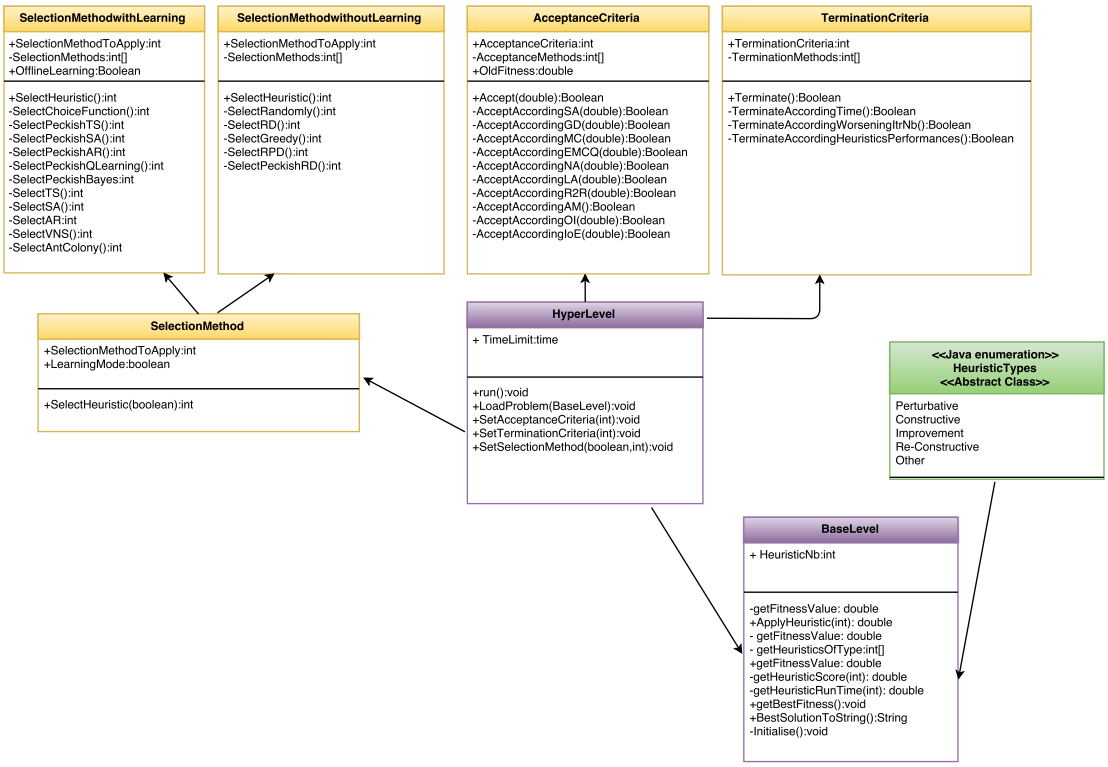

Figure 1.3: The Proposed Hyperheuristic Framework.

1.6 Contributions and Overview

This thesis is organized as follows: in chapter 2 we deal with the Workover Rig Problem. We review the state of the arts and propose a new mathematical model and several valid inequalities. We then propose a selective hyperheuristic application with two learning methods applied in real instances of the problem.

The following contributions summarize the results of this chapter.

1. Monemi, R. N., Danach, K., Khalil, W., Gelareh, S., Lima, F. C., & Aloise,

D. J. (2015). Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. Expert Systems with Applications, 42(9), 4493-4505

2. Danach, K. M., Khalil, W., Junior, F., & Gelareh, S. (2014). Routing parallel heterogeneous machines in maintenance planning:A hyperheuristic approach. ICCSA, (p. 441). Le Havre, France.

chapter 3 is composed of two parts. The first part introduces a new variant of hub location routing problem which is referred to as p-Hub Location Routing Problem. After a through literature review, we propose a 3-index design variable model for this problem. We then propose a Lagrangian relaxation for a 2-index version of the problem. We then propose a HH that exploits information from the Lagrangian Multipliers to solve the problem. Two learning mechanisms were employed within our HH. The second part, examines the effectiveness of the aforementioned HH (together with its learning mechanisms) on instances of another variant of Hub Location Routing Problem proposed in Rodríguez-Martín et al. (2014).

The following contribution is the outcome of this chapter:

1. Danach, K., Khalil, W., Gelareh, S., Semet, F., & Junior, F. (2015). Capacitated Single location P-Hub Location Routing : Hyperheuristic Approach. ROADEF. Marseille, France.

In chapter 4 we study the impact and effectiveness of several heuristic selection methods on the overall performance of HHs proposed for our problems in the previous chapter. This includes the approaches including learning mechanisms and

without it.

The following contribution is the outcome of this chapter:

1. Danach, K., Gelareh, S., Khalil, W., & Semet, F. (2016). Capacitated Single Allocation p-Hub Location Problem:Hyperheuristic Approaches with different Selection Methods. ROADEF. Compiegne, France.

During this thesis, we have also produced other contributions that make use of the knowledge of HH we have obtained during this thesis.

1. Danach, K., Khalil, W., & Gelareh, S. (2015). Multiple Strings Planing Problem in Maritime Service Network: Hyperheuristic Approach. TAEECE. Beirut, Lebanon.

2. Danach, K., Haj Hassan, J., Khalil, W., Gelareh, S., & Kalakish, A. (2015). Routing Heterogeneous Mobile Hospital With Different Patients Priorities: Hyperheuristic Approach. DICTAP. Beirut, Lebanon.

Chapter 2

Workover Rig Problem

2.1 Introduction

One of the most important natural resources of the world since late XIX century is oil, which shapes our lives in many ways, not only by being the main energy source of our era, but also its uses on plastics, road construction, pharmaceutical drugs, etc. The process of finding, drilling, producing, transporting and refining oil provides a wide range of research fields, from geology to biochemistry and so on.

Many land (onshore) oil fields are composed of many wells, which are distributed geographically. Occasionally, failures happen on these wells, requiring an intervention inside them to return to their original condition. Such operation normally includes substituting the production equipments (cleaning) or stimulating the reservoir itself (stimulation), to name a few. Those interventions require the use of *workover rigs*, big structures that can be dismounted, transported and

mounted from one well to another, providing safety and accuracy conditions to the intervention. Renting of workover rigs come at great cost, thus having them at standby availability is expensive. This chapter boards the problem of prioritizing onshore interventions using workover rigs to minimize production loss associated with the wells awaiting service. The problem in study here can be classified like a particular case of machine scheduling problem.

A classical problem of machine scheduling represents a set of tasks (or jobs) to be processed, where each task consists of a sequence of operations to be performed using a given number of machines. The processing of an operation requires the use of a specific machine for a particular processing time, and each operation must be executed in the order given by the sequence. Each machine must process only one operation at a time. The objective is to arrange the wells so that the global performance measures can be optimized.

A vast body of literature is dedicated to the classical problems of scheduling (job-shop and flow-shop), but in specific applications, the quantity of publications is rather limited. Two well-known samples of historical papers for the classical problems are related to the problems with 10 wells and 10 workover rigs proposed by Muth J.F. (1963), which was only solved 26 years later by Carlier and Pinson (1989). In this problem, each task has to be processed on each of the given workover rigs exactly once —classical job-shop scheduling problem.

Here we are concerned by a particular case of machine scheduling, an application to the problem of *Workover Rigs Scheduling (WRS)* for maintenance services in oil wells of onshore fields. The problem consists in finding the best schedule for

a small number of workover rigs to minimize the production loss, associated with a large number of wells waiting for service.

2.2 Literature review

Smith (1956) showed that if the problem has only one rig and no time windows, the optimal sequencing is obtained, independent of the quantity of wells, by sorting the wells in an increasing order of value $\frac{P_i}{Et_i}$, where P_i is the rate of daily production loss of well i and Et_i is the estimated maintenance service time of well i.

Barnes et al. (1977) provided lower bounds for workover rigs problem. The authors consider m rigs and n wells, and show that a lower bound can be obtained as $Max\{B(1), B(n)\}$ where $B(n)$ is the total production loss with n rigs and $B(1) = \frac{1}{2m}[(m-1)B(n)+2B(1)]$ is the total production loss with only one single rig.

Noronha et al. (2001) presented a greedy heuristic algorithm for the workover rigs problem. The authors consider priorities for the wells as $G_{ij} = \frac{P_i}{T_{ij}}$, where, P_i is the daily rate of production loss of well i, and T_{ij} is the estimated maintenance service time of the well i by the rig j. In their greedy approach, the authors consider also the environmental risks corresponding to the service. The proposed algorithm was later used a constructive phase of a GRASP metaheuristic.

Aloise et al. (2006) proposed a variable neighborhood search (VNS) metaheuristic. In the VNS algorithm the authors, have used the constructive heuristic $H1$, proposed by Noronha et al. (2001), which adds one well at-a-time to the

routes computed for the workover rigs. The local search procedure proposed for the VNS is based on a swap neighborhood defined by all solutions, which can be obtained by the exchange of a pair of wells from the current solution. The numerical experiments were performed with real-life instances showing a loss reduction of 16.4% on average. This VNS metaheuristic approach is currently being used as an operational scheduling tool at Petrobras S. A (Brazilian National Petroleum Corporation).

Mattos Ribeiro et al. (2011) proposed a simulated annealing (SA) for a variant of the WRS where the travel time is not considered. The authors have used CPLEX 12.1 (IBM, 2009) to solve instances with up to 50 wells. They have also reported that the proposed SA presents a low deviation (the worst case, 0.037%) from optimality and takes, approximately, 10 seconds for solving real-life instances composed of 25, 50, 75, 100 and 125 wells, with 2, 4, 6, 8 and 10 rigs.

In Duhamel et al. (2012), three mixed integer linear models are proposed. The first model improves an existing scheduling-based formulation. The second one, uses an open vehicle routing approach and the third one is an extended model for which a column generation strategy is developed. The models were tested using CPLEX 12.0 under default parameters and the instances were composed with up to 60 wells, the number of rigs varies from 2 to 5 and the time horizon is set to 15 days. The authors report optimal values for medium-size instances of WRS.

Ribeiro et al. (2012a) presented the WRS as a workover rig routing problem, a particular case of vehicle routing problem with time windows, in context of the operations of onshore oil fields. The authors have proposed three metaheuristics

for the problem: an iterated local search, a clustering search, and an Adaptive Large Neighborhood Search (ALNS). They have carried out experiments with 50, 100 and 500 wells, and 5 and 10 rigs, testing a total of 60 instances. The authors reported a superior performance of ALNS on larger instances.

Mattos Ribeiro et al. (2012), in this work the authors propose a branch, price and cut algorithm as the first exact algorithm for the WRS, which is modeled as a workover rig routing problem. The computational experiments relies on a set of 40 instances (with 100 and 200 wells, 5 and 10 rigs, and 200 to 300 units of time for the horizon). For the larger instances (200 wells), 12 of the 40 instances could not be solved, in particular, all instances with 200 wells,10 rigs, and horizon time of 300 hours were unsolvable.

Ribeiro et al. (2012b) look at the problem as a routing problem and proposes a branch, price and cut algorithm for solving instances of this problem up to 200 wells and 10 rigs. Recently, Ribeiro et al. (2014) proposed three different heuristics such as branch-price-and-cut (BPC) heuristic version of Ribeiro et al. (2012b), an adaptive large neighborhood search (ALNS), and a hybrid genetic algorithm (HGA). They managed to solve up to 10 rigs and 300 wells.

2.2.1 Objective and contribution

We propose a new model, which is based on an arc-time-indexed formulation inspired by the work in Pessoa et al. (2010). We also propose several classes of valid inequalities in order for tightening the MIP polytope.

The work was motivated by the industrial application and the need for an

efficient and scalable solution framework that can exploit the knowledge hidden in all the heuristics proposed for the problem at hand.

Here, a heuristic selection type of hyperheuristics is proposed, which is a self-adaptive mechanism in the sense that the selection of heuristic algorithm (chosen from a pool of constructive, improvement and destructive ones) iteratively applied to the problem is based on a proposed learning method. Our main goal is to show that the self-adaptive nature of the learning mechanism controlling the heuristic selection type hyperheuristic allows a very efficient exploration of neighborhoods using several heuristics. This helps us to identify the classes of heuristic, among those applied here, which fit best for solving instances of WRS.

While we only focus on the HH heuristic in this thesis, however, in order to evaluate the performance and measure the quality of solutions reported by our solution framework, the outcome of the HH has been injected into an exact solution method which is a branch-and-price algorithm. The best solution reported by the HH constitutes the initial columns of a branch-and-price algorithm and helps accelerating convergence. With a fast convergence or within a given time limit when the branch-and-price decide to terminate we are able to show whether the solution reported by the HH is optimal or will have an indication of distance from optimality. We must emphasize that this branch-and-price have been developed by a different group of researchers and is independent of the work in this thesis. Details of this approach can be found in (Monemi et al., 2015).

2.3 Problem Description

The problem is described as following: *A set of wells requiring maintenance,* $\mathcal{J} = \{1, \ldots, n\}$*, scattered within a geographical area.* $[d_{ij}]_{|\mathcal{J}|\times|\mathcal{J}|}$ *represents the distance in term of travel time between every ordered pair of wells* $(i, j) \in \mathcal{J} \times \mathcal{J}$*. A set of workover rigs (i.e. mobile maintenance heterogeneous workover rigs) are available to serve the wells upon need and there is a service capacity* q_r *associated to each workover rig* $r \in \mathcal{R}$ *such that* $q_1 \leq \cdots \leq q_{|\mathcal{R}|}$*. Every workover rig* r *can offer all the services offered by rig* r'*,* $q_{r'} \leq q_r$*. To every well* $j \in \mathcal{J}$ *a required service level* l_j *is associated and there is a smallest* $\hat{r}$ *for which every rig* $r : r \geq \hat{r}$ *can serve the well* j*. At time* 0*, all the rigs are at their initial locations and the production is already interrupted (or significantly deteriorated) at all the wells requiring maintenance. Duration of maintenance on well* $i \in \mathcal{J}$ *is* p_i *and production revenue per time unit has a monetary value of* $g_i, i \in \mathcal{J}$*. The objective is to minimize the total lost production revenue that is to minimize the total completion time of maintenance activities.*

2.4 Mathematical Model

Our modeling framework relies on a set of assumptions as in the following: i) A *field of work* is comprised of a set of wells and a set of workover rigs within this field dedicated to serve wells inside it. Normally, the area of this field as well as the travel time between every pair of wells is limited. This suggests that a workover rigs does not need to travel a very long distance between pairs of well.

ii) The process takes place on a discrete-time planning horizon. Moreover, we assume that dismounting (equivalently mounting) of all workover rigs are equal and equivalent to one unit of time, δt, iii) the dismounting (equivalently mounting) time is already included in the processing time of every workover rig, and, iv) without loss of generality, we assume that the rigs are heterogeneous meaning that no two machines have the same compatibility list (otherwise, the subproblems per those similar machines will collapse to one as explained in Pessoa et al. (2010)).

The necessary parameters and variables are listed in Table 3.4:

Table 2.1: Model Parameters and variables.

Parameters:
$\mathcal{J}$: the set of well to be serviced, $\mathcal{R}$: the set of workover rigs to service the wells, $\mathcal{T}$: the time horizon periods, $1, \ldots, T$, p_i: the process time of task i, d_{ij}: the travel distance between the location of task i and the location of task j, 0: the dummy task, which is the *first* and *last* task on every machine.
Variables:
x_{ijrt}: 1, if task i is finished and task j is started at period t on machine r, 0, otherwise.

We define $\mathcal{J}^+ = \mathcal{J} \cup \{0\}$. To this end, we use *workover rig (WOR)* and *machine*, alternatively. The *well* and *tasks/job* are also used alternatively.

2.4.1 Workover Rig Scheduling (WRS) Problem

In our modeling approach at $t = 0$, every machine is processing dummy task 0.

(WRS)

$$min \sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{J}}\sum_{j\in\mathcal{J}^+} g_i(t+p_j)x_{ijrt} \tag{2.1}$$

$s.\,t.$

$$\sum_{j\in\mathcal{J}} x_{0jr\ p_0+d_{0j}} + x_{00r\ p_0} = 1, \qquad \forall r \in \mathcal{R}, \tag{2.2}$$

$$\sum_{i\in\mathcal{J}}\sum_{t=p_i+1}^{|\mathcal{T}|} x_{i0rt} + x_{00r\ p_0} = 1, \qquad \forall r \in \mathcal{R}, \tag{2.3}$$

$$\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{J}^+:j\neq i}\sum_{t=p_i}^{|\mathcal{T}|} x_{ijrt} = 1, \qquad j \in \mathcal{J}, \tag{2.4}$$

$$\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{J}^+:j\neq i}\sum_{t=p_j}^{|\mathcal{T}|} x_{jirt} = 1, \qquad j \in \mathcal{J}, \tag{2.5}$$

$$x_{ijrt} \leq \sum_{\substack{l\in\mathcal{J}\\ t+d_{jl}+p_j\leq|\mathcal{T}|}} x_{jlr\ t+d_{jl}+p_j}, \\ + x_{j\ 0\ r\ t+p_j} \qquad \forall t \in \mathcal{T}, r \in \mathcal{R}, i \in \mathcal{J}^+, j \in \mathcal{J} : j \neq i, \tag{2.6}$$

$$x_{ijrt} \in \{0,1\}^{|\mathcal{J}^+|\times|\mathcal{J}^+|\times|\mathcal{T}|\times|\mathcal{R}|}. \tag{2.7}$$

The objective function (2.1) accounts for the minimizing the lost production revenue.

Constraints (2.2) ensure that for every machine, either it start working on a task j at $p_0 + d_{0j}$ (assuming that $|\mathcal{R}| \leq |\mathcal{J}|$) or $x_{00r0} = 1$ meaning that the task 0 is being treated after the task 0 and terminated at time p_0. If a real job $i \neq 0$ started, then such a first task on every machine r does not start before $p_0 + d_{0i}$, which accounts for the process time of dummy job 0 plus travel time from the initial location (where the dummy job takes place) to i.

The last job on every machine is actually the dummy task 0, which is executed after the last real task i. Such a task does not occur during $[0, p_0 + d_{oi}]$, evidently. This is ensured in constraints (2.3).

Constraints (2.4) ensure that the real task j will start at some point in time after another task $i \in \mathcal{J}^+$ on one of the available workover rigs. However, this cannot start earlier than $p_0 + d_{0i}$. Analogously, constraints (2.5) ensure that a real task j is followed by a task $i \in \mathcal{J}^+$ on the same machine and this cannot occur within $[0, p_0 + d_{oi}]$.

For task j executed after a real task i on machine k and time t there must be a consecutive task $l \in \mathcal{J}^+$, which starts at $t + p_j + d_{jl} \leq |\mathcal{T}|$ (starts at $t + p_j + d_{jl} : l = 0$). This has been ensured by constraints (2.6).

2.4.2 Valid inequalities

Constraints (2.2)-(2.6) describe the polytope of the problem including all the feasible solutions given the fact that all the variables have positive cost in the objective function. However, there are some other constraints which, can be added to the model as following:

a) Every task i is excused at some point in time on one machine:

$$\sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{J}^+}\sum_{r\in\mathcal{R}} x_{ijrt} = 1, \qquad \forall i \in \mathcal{J}^+ \tag{2.8}$$

b) For two distinct well i, j on the same machine k, either of them is executed before the other one:

$$\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}(x_{ijrt} + x_{jirt}) = 1, \qquad i, j \in \mathcal{J} \tag{2.9}$$

c) The total number of variables $x_{ijrt},\ i, j \neq i \in \mathcal{J}^+,\ r \in \mathcal{R}, t \in \mathcal{T}$ taking 1 in any feasible solution is constrained as following:

$$\sum_{i,j\neq i\in\mathcal{J}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}} x_{ijrt} = |\mathcal{J}|, \tag{2.10}$$

$$\sum_{i,j\neq i\in\mathcal{J}^+}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}} x_{ijrt} = |\mathcal{J}| + 2|\mathcal{R}|. \tag{2.11}$$

d) a real task $i \in \mathcal{J}$ must be followed (precede) by a another task $j \in \mathcal{J}^+ : j \neq i$:

$$\sum_{t=p_0}^{|\mathcal{T}|} \sum_{j \in |\mathcal{J}^+|} x_{ijrt} \leq \sum_{t=p_0}^{|\mathcal{T}|} \sum_{j \in |\mathcal{J}^+|} x_{jirt}, \qquad \forall r \in \mathcal{R}, i \in \mathcal{J}, \tag{2.12}$$

e) There is no 3-cycle in the order of jobs on a give machine:

$$\sum_{t=p_0}^{|\mathcal{T}|} (x_{ijrt} + x_{jlrt} + x_{lirt}) \leq 2, \qquad \forall r \in \mathcal{R}, \underset{j \neq i, l \neq i, l \neq j}{\{i, j, l\}} \in \mathcal{J}, \tag{2.13}$$

f) On the same machine no two wells can be executed at the same time:

$$\sum_{i \in \mathcal{J}^+} \sum_{j \in \mathcal{J}^+ : j \neq i} x_{ijrt} \leq 1, \qquad \forall r \in \mathcal{R}, t \in \{p_0, \ldots, , |\mathcal{T}|\} \tag{2.14}$$

$$\sum_{i \in \mathcal{J}^+} \sum_{j \in \mathcal{J}^+ : j \neq i} x_{jirt} \leq 1, \qquad \forall r \in \mathcal{R}, t \in \{p_0, \ldots, , |\mathcal{T}|\} \tag{2.15}$$

g) On every rig r either at some point $t \in \mathcal{T}$ a task $j \in \mathcal{J}$ is started (finished) as the first (last) task or $x_{00r0} = 1$:

$$\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} x_{0jrt} + x_{00r0} = 1, \qquad \forall r \in \mathcal{R}, \tag{2.16}$$

$$\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} x_{j0rt} + x_{00r0} = 1, \qquad \forall r \in \mathcal{R}, \tag{2.17}$$

The aforementioned constraints are particularly useful when due to some relaxations or reduced cost updates, the pricing problem or Lagrangian subproblem objective has variables with negative cost. There, these constraints serve to avoid

too many variables take 1 and will tighten the pricing problem/Lagrangian relaxation polytop.

2.4.3 Preprocessing

Some of the variables can be set to zero in advance. The total number of variables eliminated in this way depends on the instance of problem being solved.

Lemma 1. *None of the wells $i \in \mathcal{J}$ can receive service during* $[0, \min_{j \in \mathcal{J}}\{p_0 + d_{0j}\} - 1]$ *on any machine.*

Analogously, we have:

Lemma 2. *None of the wells $i \in \mathcal{J}$ can receive service during* $[\mathcal{T} - \min_{j \in \mathcal{J}}\{p_j + d_{j0}\} + 1, \mathcal{T}]$ *on any machine.*

variable fixing by task-machine feasibility

As stated in the problem description, the workover rigs of larger size can serve those wells of equal size or smaller while the inverse does not hold.

Let $\mathcal{R}(j)$ represents the set of all workover rigs that can serve task j. The following constraint ensures the feasibility of task assignment.

$$x_{ijrt} = 0 \qquad \forall i, j \in \mathcal{J} : j \neq i, r \notin \mathcal{R}(i) \vee r \notin \mathcal{R}(j) \tag{2.18}$$

2.4.4 Illustrative example

We have considered $|\mathcal{R}| = 3$, $|\mathcal{J}^+| = 7$ and a time horizon $|\mathcal{T}| = 16$. We have run the model in CPLEX 12.60 for a `TiLim` = 1000 seconds. $g_i = 1,\ \forall i \in \mathcal{J}^+$, $d_{ij} = \lfloor \frac{(i \times j + (n-i)(n-j))}{10} \rfloor,\ \forall i, j \in \mathcal{J}^+$ and $p_i = i,\ \forall \in \mathcal{J}^+$.

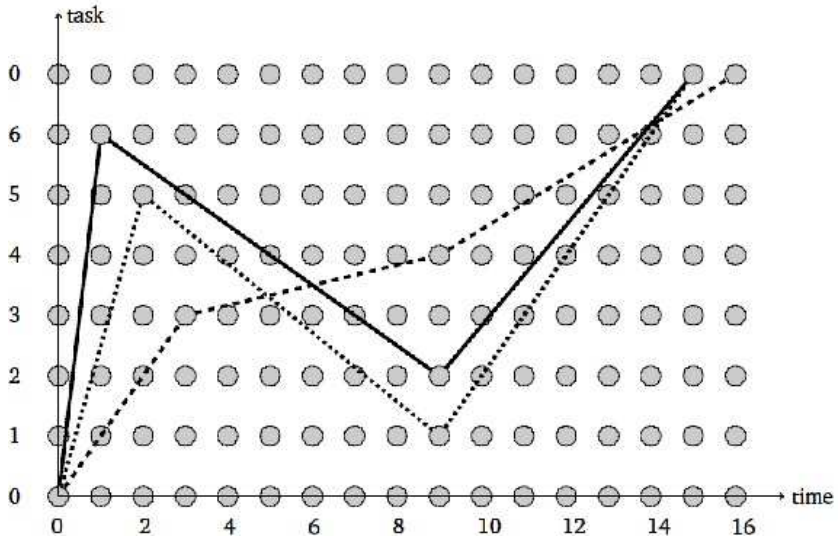

Figure 2.1: Illustrative example with $|\mathcal{R}| = 3$, $|\mathcal{J}^+| = 7$ and $|\mathcal{T}| = 16$. Every machine starts by processing job 0 at $t = 0$ and terminates at the same job as the job processed on it.

2.5 Hyperheuristic for WRS Problem

The mathematical model of WRS becomes intractable even with small $|\mathcal{T}|$, few wells and rigs. Therefore, in order to solve more realistic size instances, we have to resort to other techniques, which are efficient and provide good approximation of optimal solutions. From among such techniques, we have chosen to use the concept of Hyperheuristic (Burke et al., 2009a, 2008b, 2013)

The low-level heuristics are categorized as follows:

1. *constructive* ones, which produce complete feasible solutions from the scratch,

2. *improvement* methods that accept a feasible solution and try to improve it within a predefined neighborhood,

3. *perturbation* procedures that try to inject some noises to the process in order to produce solutions, which might help in finding better solutions and possibly escaping from local optima, and finally

4. *reconstructive* mechanisms to (randomly or deterministically) destroy and reconstruct part of solutions again, hoping to jump to some unexplored part of the search space that might involve better solutions.

Low Level Heuristic

In the following, we briefly explain how each of these low-level heuristics performs.

1) *constructive heuristic*: we construct an initial feasible solution, step-by-step, according to a set of predefined rules without any effort to improve this solution. i) *C1* construct a solution by exploiting the instance information. Here, we sort wells based on the decreasing production parameter. Subsequently, we

start from the wells with highest production and allocated each well i to the compatible rig r that is less utilized among others (see Algorithm 4).

2) *improvement heuristics*:

The improvement algorithm starts from a feasible solution and improve it by applying successive changes within a given neighborhood. Our neighborhoods are characterized by the following moves:

i) *better-sequence-on-the-same-rig*: For a given rig, we reorder (one task per time) the sequence of allocated wells to be served by this rig (see Algorithm 2) and among the improving solution found, we move to the best found feasible solution in a greedy manner.

ii) *inset/drop-between-two-different-rigs*: aiming at making a balanced utilization and fair distribution of tasks among rigs, for two *randomly* chosen rigs $r_i, r_j \neq i$, we consider three moves: 1) move one tasks from rig r_i to rig $r_j \neq r_i$ such that the difference between the objective functions of rig r_i and rig r_j, i.e. $\Delta = (OF(r_i) - OF(r_j))$, being minimized. 2) removing a well from the list of wells being served by a given rig and insert it in a proper place within the sequence of wells being served by the rig with the least objective function.

Algorithm 1: Constructive Heuristic 1 (C1)

1: **procedure** CONSTRUCTIVE1($\mathcal{J}, \mathcal{R}$) ▷ INPUT: sets of wells and rigs

2: $S_s \leftarrow \emptyset, \quad \forall s = 1, \ldots, m, \quad (m = |\mathcal{R}|)$

3: $i \leftarrow 0$

4: $M \leftarrow 0$

5: $p_{iM} \leftarrow \infty$

6: **for** $i = 1, \ldots, n \quad (n = |\mathcal{J}|)$ **do**

7: **for** $r = 1, \ldots, m$ **do**

8: **if** $(p_{ir} < p_{iM})$ **then** ▷ p_{ir} is the processing time of well i by rig r

9: $M \leftarrow r$

10: **end if**

11: $\mathcal{S}_M \leftarrow [S_M, \mathcal{J}_i]$

12: **end for**

13: **end for**

14: **return** $\mathcal{S}$ ▷ OUTPUT: sequence of wells associated to each rig r

15: **end procedure**

Algorithm 2: Order Improvement Operator

1: **procedure** ORDERIMPROVE($\mathcal{S}$) ▷ INPUT: sequences associating wells to rigs

2: $L \leftarrow \emptyset$

3: $G \leftarrow \emptyset$

4: $Mid \leftarrow \emptyset$

5: $p \leftarrow$ *SelectPivot*($\mathcal{S}$)

6: **for** $s = 1, \ldots, m \quad (m = |\mathcal{S}|)$ **do**

7: **if** $(Lv_s \leq Lv_p)$ **then** ▷ Lv_s is the production loss value of sequence s

8: *Append*(L, s)

9: **else**

10: *Append*(G, s)

11: **end if**

12: **end for**

13: $\mathcal{S} \leftarrow$ *Concatenate*(ORDERIMPROVE(L), $Mid(p)$,ORDERIMPROVE(G))

14: **return** $\mathcal{S}$ ▷ OUTPUT: Ordered sequences by best objective function value

15: **end procedure**

3) *Perturbation heuristics*:

The perturbation phase assures a *diversification* strategy during the search; it tries to explore the search space via randomized efforts to escape search from local optimum. In this study, we implement different perturbation heuristics as the following:

1. *Mutation-like*: it resembles the genetic operator used to maintain diversified population from one generation to another. This operator corresponds to a perturbation in the configuration of a chromosome (a sequence of wells on every rig), that prevents the risk of premature convergence and allows the exploration of other areas in the search space. The operator is applied with two chromosomes that are selected randomly in the current population.

2. *Crossover*: In this case, the swapping of 'genetic material' is made with the

rigs (part of the chromosome), while in the mutation operator is made with two wells, one in each chromosome.

3. *SequenceRandomSwap*: This operator is a kind of mutation-like operator. But here, the perturbation is made in the same chromosome, which is carried out randomly.

4) *Destroy-and-reconstruct heuristic*:

In order to better exploit the search space (diversify the search process), we define an operator that destroys and reconstructs part of a solution, which is randomly chosen. Thus the operator *Reconstructive1* destroys a part of the current solution and then reconstruct it by using constructive algorithms presented previously.

2.5.1 Reinforcement learning

A reinforcement learning method in an *selection-type* hyperheuristic, selects a heuristic that has the maximal utility value (Burke et al., 2008b). Thus, it is a mechanism that chooses corresponding actions given some information about its performance, and update this performance at each time it is applied. The process takes into consideration when to diversify and when to intensify in the search process. In chapter 4, we will elaborate more the concept, techniques, and methods of reinforcement learning.

Here, we employ two different methods of learning: 1) the built-in method of generic intelligent hyperheuristic - GIHH (Misir, 2012) [1] and 2) our improved

[1]The code is publicly available at https://code.google.com/p/generic-intelligent-hyperheuristic/

method called *Alternative Learning Method (A.L.M.)*.

Both methods rely on using Hyperheuristics Flexible framework (Hyflex), which is an well-defined library incorporating a set of methods that assure the communication between the problem domain and the solver components.

In GIHH, The number of new best solutions and time spent by each heuristic are determinant to update the probability vector in the selection operation. In addition, an acceptance criterion is defined to balance intensification and diversification processes. In general, diversification is occurred at the beginning of the search, followed by intensification towards the end.

Our proposed hyperheuristic, is based on constructing a feasible solution and consequently applying of heuristics that are chosen following a specific criteria. Then we design a tabu list that prevents some heuristics, in particular conditions, to be applied, in order to guide the search to apply heuristic series to reach the optimal solution.

We define a so-called *heuristic weight* variable as the following:

$$\Psi_i = \sum \Delta f = \sum (f^{in} - f^{out})/f^{in}$$

A Heuristic H_i weight is equal to the negative sum (for minimization case) of the objective function values taken by the application of H_i divided by the total number of times that H_i has been applied during the search.

Our tabu list T_l is designed as the following: Given our time limit, we initially set $T_l = \emptyset$ and at each iteration, T_l is updated as follows: if the CPU time spend

is less than the half of the time limit, we put all heuristics that, in mean more frequently used than the other my suggestion. Also, all heuristics with too small weights in T_l. In case of the consumed time is greater than the half of the time limit, T_l will contain all heuristics expect those having the best improvement and those having their weight greater than the mean of heuristics weights. The Tabu list is updating with change of quality of each heuristic - calculation of Ψ_i. To simplify our tabu strategy, we represent it as a learning method in Algorithm 3.

Algorithm 3 Alternative Learning Method

```
1:  procedure LEARNINGMETHOD(t)                          ▷ INPUT: time limit
2:      α ← 0.7
3:      P ← Proba(Random)
4:      while (consumed_time < t) do
5:          if (consumed_time < t/2) then
6:              if (h == SearchHightImproveBestHeuris()) then
7:                  return h
8:              end if
9:              if (HeurisWeight[h] > CalcWeightMean()) then
10:                 return h
11:             end if
12:         else
13:             if ((HeurisNbOfCall[h] < CalcCallMean())) then
14:                 return h
15:             end if
16:             if (HeurisWeight[h] > CalcWeightMean()) then
17:                 return h
18:             end if
19:             if (h == SearchHightImproveBestHeuris()) then
20:                 return h
21:             end if
22:         end if
23:     end while                                    ▷ OUTPUT: heuristic selected
24: end procedure
```

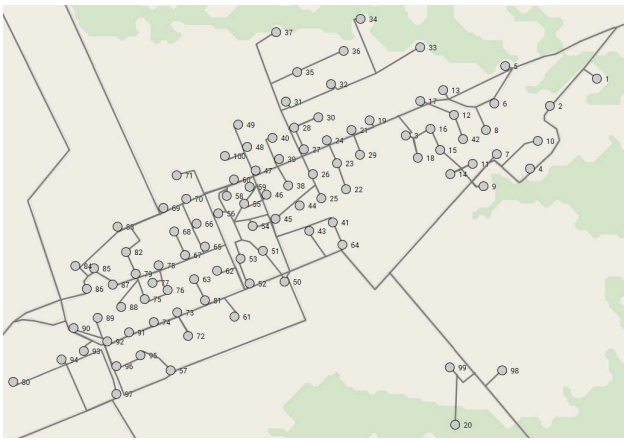

Figure 2.2: The wells are uniformly distributed within a given geographical zone.

2.6 Numerical experiments

Our computational experiments are based on a set of perturbed data from Petrobras and within the Brazilian territory. The data relates to a particular field of operation within which the total number of wells is around 200 wells and are densely distributed in such a moderate size field. A geographical presentation of spatial distribution of wells within this field for an aggregated instance of size $|\mathcal{J}| = 100$ is presented in Figure 2.2.

Our order of business is as follows: we run our hyperheuristic on the instances of the problem. We compare our method against that of GIHH and present an analysis of the results. In addition, HH results are compared with result of an exact method called branch, price and cut algorithm which uses the best-known solution of our hyperheuristic as an initial column in the branch, price and cut algorithm in

Monemi et al. (2015).

All experiments were performed on an Intel 2.54 GHz core i5 CPU and 4 Gb of memory running on Windows 7. All instances are named in a format `instance_i_j` where i indicates the number of wells, and j indicates the number of workovers in the instance.

There are, in total, 37 instances ranging from 10 tasks and 3 workover rigs to 200 tasks and 12 workover rigs. Table 2.2 reports the numerical experiments. The first column reports the instance name, the second one indicates the objective value of the initial solution. The third (resp. fifth) column reports the objective function value of the best solution found when using the learning mechanism of GIHH framework (resp. A.L.M.). The computational times of GIHH and that of A.L.M. are reported in the fourth and sixth columns, respectively.

We further assume $p_0 = 0,\ d_{0j} = 0,\ \forall j$ in our experiments.

Table 2.2: Best solution and execution time results of the two methods.

Instance Name	Init. Obj.	Best Obj. (GIHH)	CPU Time (GIHH)	Best Obj. A.L.M.	CPU Time A.L.M.
Instance_10_3	38223	21578	0.534	19683	0.117
Instance_15_3	96179	50020	0.431	40054	1.143
Instance_15_5	51415	26055	5.019	28965	1.093
Instance_20_3	165681	90499	0.841	86809	0.241
Instance_20_5	97214	52314	5.206	45985	1.164
Instance_25_3	245356	136401	0.949	124236	1.045
Instance_25_5	214666	102923	5.636	97772	0.638
Instance_25_7	123666	64582	5.11	63494	0.150
Instance_30_3	559221	165326	4.697	162479	0.566
Instance_30_5	231082	126291	0.593	117757	0.220
Instance_30_7	186164	124002	0.367	94619	0.155
Instance_35_3	521748	181936	5.672	180341	1.074
Instance_35_5	396655	172343	0.603	159223	0.304
Instance_35_7	245830	119389	0.549	104206	0.208
Instance_50_5	768691	286110	5.013	276221	0.705
Instance_50_7	549441	221749	5.296	201456	0.556
Instance_50_9	413259	204595	5.561	200710	0.362
Instance_50_10	241935	170762	0.589	154070	0.295
Instance_75_7	1357179	503698	10.079	493423	0.658
Instance_75_9	790796	440775	9.921	445433	0.826
Instance_75_10	849473	386271	10.008	367835	4.203
Instance_100_7	6284802	1748851	10.145	1719752	4.273
Instance_100_9	6302564	2151156	1.471	2020055	4.118
Instance_100_10	5062012	1792773	10.052	1728323	4.276
Instance_100_12	4287143	1519804	9.711	1448339	1.116
Instance_125_7	27388184	5136775	6.914	5010572	4.228
Instance_125_10	21967460	5351825	9.785	5185172	2.503
Instance_125_12	13899045	4166687	1.864	4109715	1.064
Instance_150_7	39577184	7413717	11.093	6959831	3.179
Instance_150_10	29616360	6922348	6.695	6778345	4.183
Instance_150_12	25065540	6590526	9.561	6234537	3.162
Instance_175_7	96465248	17056898	10.187	16443304	5.112
Instance_175_10	75896016	16379125	5.017	15493673	4.186
Instance_175_12	59023100	14735414	10.104	14380174	4.139
Instance_200_7	109649824	18386244	10.017	17943176	4.143
Instance_200_10	87850240	19201584	10.033	18810648	6.427
Instance_200_12	83277888	17716744	4.71	17785952	4.295

We introduce $\nu_a,\ a \in \{GIHH, A.L.M.\}$ defined as in (2.19) to be the accumulated relative improvement of each of the two methods over all the 37 instances.

$$\nu_t = \sum_{i=1}^{37} \frac{InitObj_a - BestObj_a}{InitObj_a} \tag{2.19}$$

For GIHH , ν_{GIHH} is equal to 61.205 while for A.L.M., $\nu_{A.L.M.}$ is equal to 63.497. This confirms that in average our method is superior to that of GIHH.

The solution method is ended in accordance with the termination criteria. The termination condition takes in consideration avoiding useless computation and tries to avoid early termination.

The proposed termination criteria, sets a global time limit:

$$T_l = \max\{10000, 1000\frac{NumberofWells}{NumberofWorkovers}\}$$

seconds for the overall computation time while for every low-level heuristic the termination criteria relies on the number non-improving solutions,

$$L_i = 100\frac{NumberofWells}{NumberofWorkovers}$$

. In addition, we set a third termination criteria to terminate the whole algorithm once $W = 10\frac{NumberofWells}{NumberofWorkovers}$ non-improving iterations is observed.

Figure 2.3 presents the execution time behavior as a function of number of wells. Clearly, almost always, when the number of wells is less than 50, the CPU

time is almost less than 5 seconds and when the number of wells is greater than 50, the consumed time is around 10 seconds using GIHH. However, the result shows that, except for *four* instances, still our computational times are always inferior to those of GIHH. For instances that have more than 50 wells in it, our computational times remains around 7 seconds.

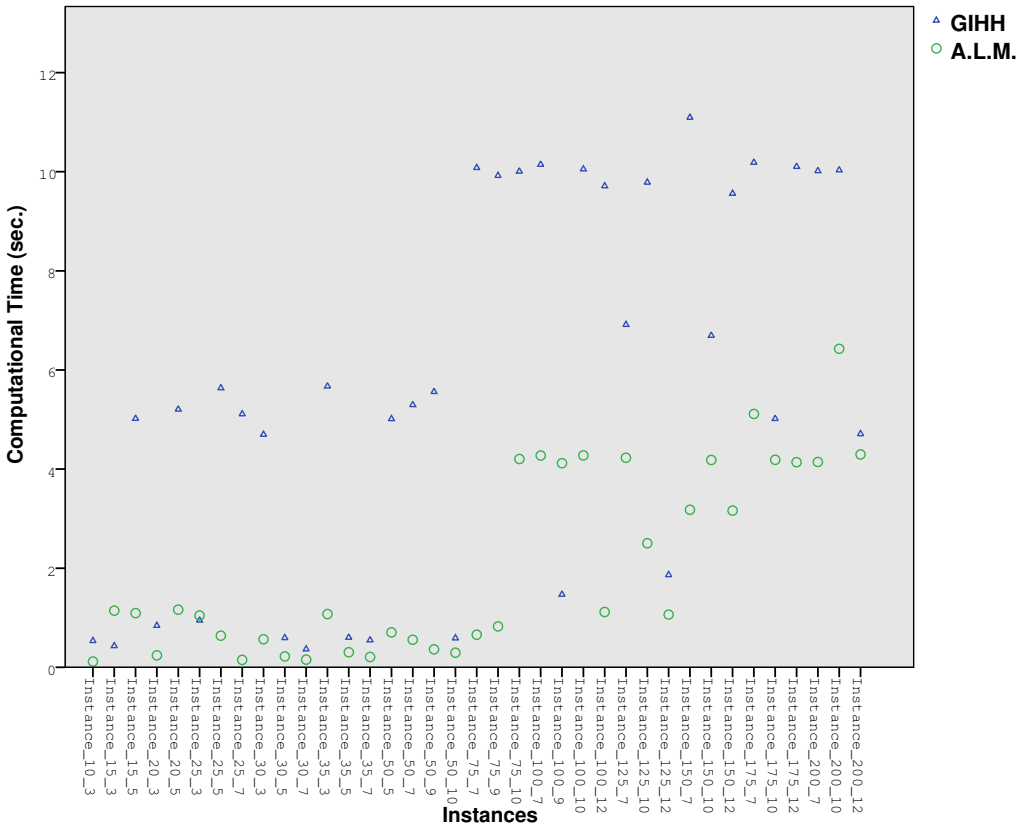

Figure 2.3: Execution time of each method.

Figure 2.4, Figure 2.5 and Figure 2.6 report the statistics of heuristic calls by each proposition for each instance. In Figure 2.4, the largest percentage is

dedicated to the order improvement heuristic in both cases (GIHH and A.L.M.). An absolute majority of the calls is associated with the order improvement heuristic in A.L.M. and a minor part belongs to the reconstructive heuristic.

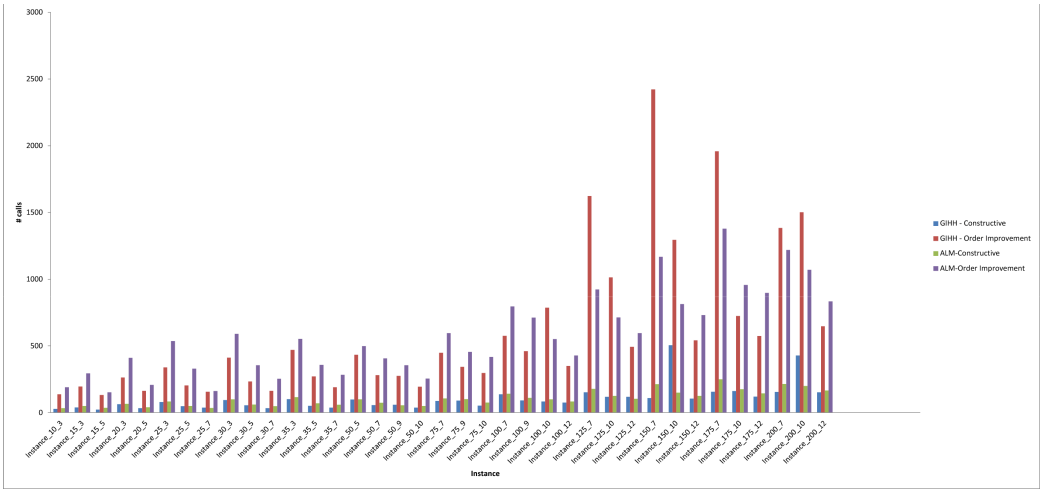

Figure 2.4: The constructive and improvement heuristics calls distribution by each method.

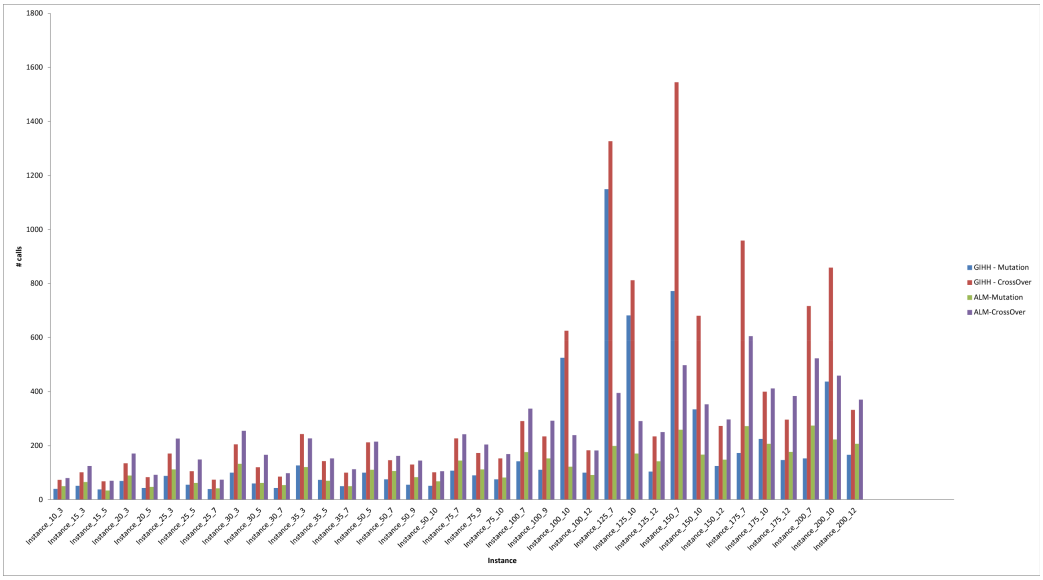

Figure 2.5: the mutation and crossover heuristics calls distribution by each proposition.

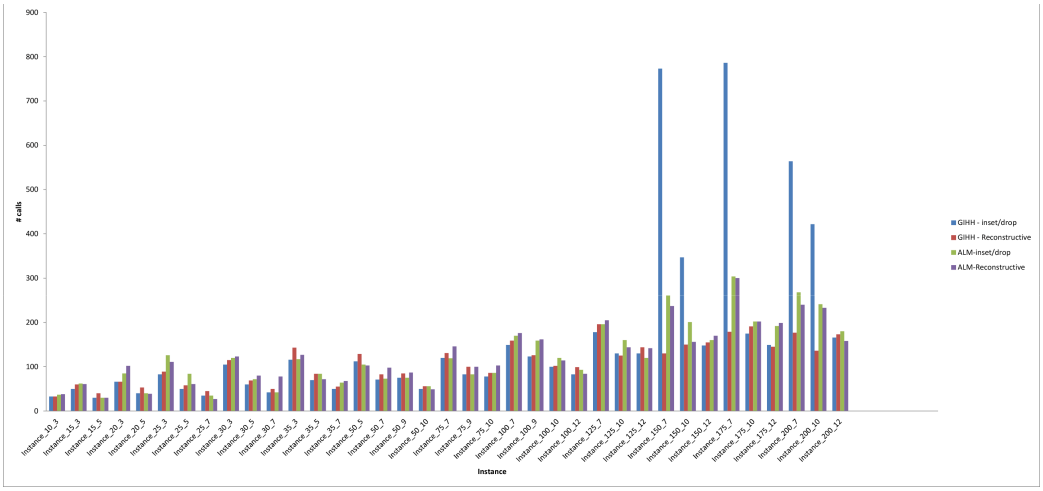

Figure 2.6: insert/drop and reconstructive heuristics calls distribution by each method.

Table 2.3: Computational results of our branch, price and cut.

	min(A.L.M., GIHH)		branch, price and cut(Monemi et al., 2015)				
Instance Name	Best Obj.	CPU (sec.)	#bb.nodes	#cols	$(\frac{f^{A.L.M} - f^{B\&p\&C}}{f^{B\&p\&C}}) \times 100$	CPU (sec.)	Term. Cond.
Instance_10_3	19683	0.117	0	26	0.00	1133.20	opt.
Instance_15_3	40054	1.143	0	31	0.00	1298.46	opt.
Instance_15_5	26055	5.019	0	37	0.00	1413.76	opt.
Instance_20_3	86809	0.241	2	40	0.00	3044.17	opt.
Instance_20_5	45985	1.164	2	53	0.00	3734.80	opt.
Instance_25_3	124236	1.045	2	74	0.00	4331.15	opt.
Instance_25_5	97772	0.638	2	88	0.00	4783.87	opt.
Instance_25_7	63494	0.150	0	105	0.01	5163.40	tlim.
Instance_30_3	162479	0.566	5	110	0.03	6415.23	tlim.
Instance_30_5	117757	0.220	6	156	0.10	6335.19	tlim.
Instance_30_7	94619	0.155	5	201	0.24	7561.65	tlim.
Instance_35_3	180341	1.074	10	226	3.55	7820.65	tlim.
Instance_35_5	159223	0.304	8	269	4.04	8214.07	tlim.
Instance_35_7	104206	0.208	5	318	5.23	9384.43	tlim.
Instance_50_5	276221	0.705	12	470	3.75	9633.20	tlim.
Instance_50_7	201456	0.556	11	512	11.55	10041.21	tlim.
Instance_50_9	200710	0.362	8	556	6.93	9792.10	tlim.
Instance_50_10	154070	0.295	10	601	9.36	9645.43	tlim.

In the Table 2.3 we report our numerical experiments. The first column reports the instance name, followed by the column reporting best objective function of the solutions found by our hyperheuristic and the CPU times in the column immediately next to it.

In column 4 and 5 , reports the number of branching performed and number of columns added before termination Monemi et al. (2015). The relative gap between the solution of our hyperheuristic and that found by the branch, price and cut upon termination was shown in the sixth column. In general, the solutions found by hyperheuristic are practically very good solutions. The CPU time is reported in the next column, which shows that while we can obtain a high quality solution with

our heuristic very quickly, a similar quality solution with branch, price and cut take much longer. This may confirm that the proposed hyperheuristic is a good and reliable alternative for a mathematical programming-based method.

2.7 Summary, conclusion and outlook to future work

Workover rig scheduling problem is an important problem in the highly sensitive petroleum production industry as it seeks to maintain the optimal production level of a company. There is a huge investment involved in acquiring, training, staffing and operations of workover rigs while the oil production itself plays direct role in the economy growth, welfare and the GDP of a country.

We have proposed a new mixed integer linear model for the workover rig scheduling problem that is based on a time-arc indexed formulation. We proposed several valid inequalities. The results of hyperheuristic were used for setting the initial columns of an exact method. We have shown that as long as the optimal solution (or the optimal interval) is known, the solutions reported by the hyperheuristic are very near to optimal solutions. This confirms the particular efficiency of our learning mechanism in controlling the selection of heuristics and allowing the right heuristic to search the right part of the solution space at appropriate iterations.

However, a major limitation of this work is that we have to keep a very small size set of heuristic as the search in the space of heuristics and neighborhoods

becomes very time consuming and deteriorates the efficiency. Moreover, the discretization of planning horizon also needs to be done cleverly in order to avoid unnecessarily enlargement of search space for every heuristic.

The research on this problem is very often limited to modeling it as a routing problem while perhaps scheduling is the dominating part of the problem. The literature is unaware of any multi-agent or distributed optimization techniques applied to this problem given an inherent distributed nature (in larger scale the wells are scattered within dispersed zones with long distances among the zones).

Further research on minimizing workover CO_2 emission, incorporating more real aspects such as workforce scheduling, task time window and ensuring robustness in the solution.

Chapter 3

Capacitated Single Allocation p-Hub Location Routing Problem

3.1 Introduction

The term *hub* is commonly used in different domains such as transportation, logistics and telecommunication systems, among others. In almost all such areas, the hubs serve as consolidation, switching and sorting centers. In transportation and logistics, the hub-and-spoke structures are present in almost all modes of transport (cross-docks in national level, major airports and container ports in continental and inter-continental level). Hubs are facilities at which arriving flows of commodities in smaller volumes originated from the spokes are consolidated, sorted and re-distributed (repartitioned) in a larger volumes on fewer highly utilised links and sent to either the final destinations that are hubs or the hubs where the spoke

destinations are allocated to them. The main motivation for deploying such a flow network structures is in exploiting economies of scale (in terms of time and/or cost) in transporting higher volumes on much more efficient corridors (connecting hubs).

Given a graph $G(V, A)$ where V is the set origins and destinations and A is the set of all possible arcs, Hub Location Problem (HLP) seeks a partition of nodes in V into hub and spokes. Consequently, each spoke is associated to one or more hubs. In classical models, two main classes are distinguished Alumur and Kara (2008): 1) *Single Allocation*: wherein each spoke node is assigned to only one hub, and 2) *Multiple Allocation*: offering the possibility that a spoke be allocate to more than one hub node. Moreover, one also distinguishes between *capacitated* and *uncapacitated* variants depending on where the capacity is being imposed. Some authors also distinguish between the cases where cardinality of the set of selected hub nodes is defined exogenously and when it is an endogenous part of the problem. This topic has been one of the very active areas of research in the past two decades.

We consider the previously defined graph G, a fixed number, p, of hubs node to be installed, Γ, the minimum number of spokes associated to each hub, C, the capacity that can be installed on each feeder route, w_{ij}, the demand to be transferred from i to j, and t_{ij}, the cost between any two node i and j, which is taken in this study as the time needed to travel between node i and j. We also define d_i as

the load of transporter after leaving node i on a spoke-level route (the load at arrival minus the total demand of i from every other node plus the total supply of i to every other node). We seek to construct from the scratch a hub-and-spoke network, which is consist of a complete hub-level subgraph with p hub nodes and p spoke-level directed cyclic routes each of which composed of a hub node and all the spoke nodes allocated to it. The hub-level network is a complete subgraph where each two connected hub have a bidirectional link. The objective function of our problem seeks minimizing the total transit time composed of transportation plus transhipment time in the whole network provided that the total volume of flow on the spoke-level arcs does not violate the capacity of transporter on it, given a homogenous fleet of transporters.

This problem is a special case of the *Bounded Cardinality Capacitated Hub Routing Problem (BCCHRP)* proposed in Gelareh et al. (2015) in which the lower and upper bounds on the number of hub nodes coincide. We refer to this problem as *Capacitated Single Allocation p-Hub Location Routing Problem (CSApHLRP-1)*.

Figure 3.1 represents a solution for an instance with $|V| = n = 10$ nodes and $p = 3$ fully interconnected hub nodes. The nodes 4, 5 and 7 represent hubs while all the remaining nodes considered as spokes. One observes that each hub together with spoke nodes allocated to it forms a directed cycle. In this solution, three cycles are define: i) $4 \rightarrow 0 \rightarrow 2$, ii) $7 \rightarrow 9 \rightarrow 8 \rightarrow 6$, and iii) $5 \rightarrow 3 \rightarrow 1$.

In this work we are dealing with a problem that fits within the category of

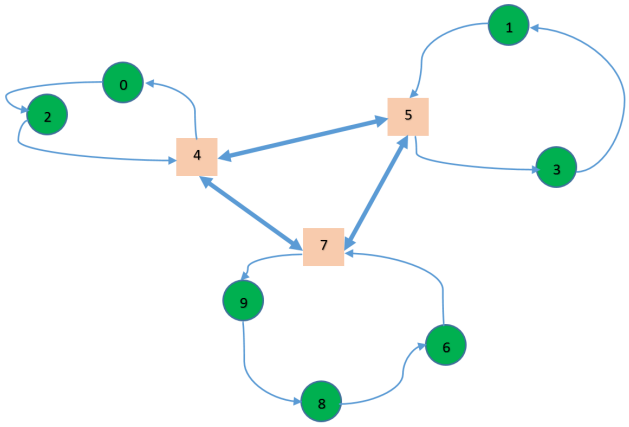

Figure 3.1: A solution of network with 3 hubs and 10 nodes.

capacitated single allocation hub location problems (CSAHLP). In the following, we review some of the related work in the literature with particular emphasis on the single allocation scheme and exogenous number of hubs in modeling part and (Meta)heuristic approaches in solution methods. Ebery et al. (2000) proposed a hybrid method that combines evolutionary algorithm (EA) with a branch-and-bound method (B&B) in order to solve a capacitated single allocation hub location problem. In this work, the proposed EA selects the set of hubs while B&B continues with the allocation part of the problem by allocating the non-hubs to the located hubs.

Correia et al. (2009) proposed a Mixed Integer Programming (MIP) formulation for the problem of single allocation where hub capacity is not an exogenous value but it is a variable with discrete choices of value. Several variants of this problem have been proposed including: capacitated and uncapacitated for single allocation problem, and capacitated flow splitting (Campbell et al., 2007) for

multiple allocation.

A MIP formulation and branch-and-cut algorithm were proposed by Rodríguez-Martín et al. (2014), which consider a single allocation model, the capacity constraint in terms of number of spokes per route, and a fixed number of hubs.

Ebery et al. (2000) proposed a MIP formulation and a heuristic for capacitated multiple allocation hub location problem. Uncapacitated p-Hub problem wherein each spoke should be connected to r hubs, is studied by Peiró et al. (2014). The authors proposed a GRASP method as a solution method.

An Uncapacitated p-Hub Single Allocation Location problem, has been proposed in Ge et al. (2007) that seeks p hubs in the network and single allocation of non-hubs to the hubs in a manner to minimize the total cost of transporting between all origin-destination pairs in the network. In this work, the spokes are simply connected to the hub and there is no particular topology imposed on the spokes allocated to the same hub.

A MIP formulation has been proposed by Gelareh et al. (2013a) for a hub and spoke network design and fleet deployment of liner shipping, in order to minimize weighted sum of transit times and the fixed deployment costs. The proposed MIP is based on 4-index formulation of HLPs and allocated spokes form directed cycles (the so called *string*). On every string, the vessel class or capacity to be installed to accommodated the volume of flow on the string must be determined from a given heterogeneous fleet.

The current work has some similarity with the one studied in Rodríguez-Martín

et al. (2014) but it s distinguished from it, as follows: In Rodríguez-Martín et al. (2014) the objective function is only to minimize the total transportation cost while in this work transshipment cost(time) has also been taken into account. In the former, the spoke-level routes are undirected while we consider that flow on the spoke-level route circulates in only one direction. Finally, in Rodríguez-Martín et al. (2014), the capacity is defined as the maximum number of spokes along a spoke-level route while we define it as the volume of flow circulating on the spoke-level route that needs to respect the capacity of transporter on it.

This work is also distinguished from Gelareh et al. (2015), by the fact that we do not take into account any upper/lower bound on the number of hubs, rather we assume a fixed cardinally, p, as an exogenous parameter.

To the best of our knowledge, so far, the non-exact methods applied to variants of hub location problems were either metaheuristics (population-based or trajectory-based) or classical heuristics.

Kratica et al. (2012) proposed a genetic algorithm (GA) for solving an uncapacitated multiple allocation problem. Binary encoding and genetic operators are used and the run-time is optimized by a caching technique.

Rabbani and Kazemi (2015) proposed a genetic algorithm and a simulated annealing for solving an uncapacitated multiple allocation p-hub center problem wherein the objective function is calculated based on Dijkstra's algorithm.

A heuristic based on (GA) is proposed by Stanimirovic (2008) in order to solve uncapacitated multiple allocation p-hub median problem. Described GA uses

binary representation of the solutions. A mutation operator with frozen bits and a caching technique, all of that contribute to improve the solution quality.

Randall (2008) proposed an ant colony metaheuristic optimization method. Four variations exploring different construction modelling choices are developed to solve the capacitated single allocation hub location problem.

Multiple Ant Colony Optimization (MACO) algorithm is proposed by Ting and Chen (2013) to solve a capacitated location routing problem, which is decomposed into two subproblems: 1) facility location problem and 2) multiple depot vehicle routing problem. The two subproblems are treated together within MACO where collaboration of colonies is operated by interchanging information through pheromone renewing the chosen location and costumer assignment.

In order to avoid local optima in p-hub location problem, Klincewicz (1992) proposed two metaheuristic: tabu search and GRASP algorithm. The considered objective was to minimize the total costs of sending traffic over links. In these two methods, local search procedure is based on the 2-exchange neighborhood.

A p-median Problem was treated by Rolland et al. (1997) that proposed a tabu search algorithm which uses short term and long term memory, and a strategic oscillation for random tabu list sizes.

Abdinnour-Helm (1998) developed a hybrid method based on a combination of genetic algorithm and tabu search in order to solve an uncapacitated hub location problem with single allocation. Genetic Algorithm is used to choose hubs, while Tabu Search is used to assign spokes to hubs.

A two phases of tabu search algorithm (location and routing phases with short

term memory) has been considered by Tuzun and Burke (1999), in order to solve capacitated hub location problem with single allocation.

Simulated annealing (SA) and iterated local search (ILS) were proposed by Zarandi et al. (2015) in order to solve single-allocation hub median problem. Another simulated annealing for single-allocation capacitated hub location problem was proposed by Vincent et al. (2010), given that routes and hubs are constrained by capacity.

Uncapacitated single allocation p-hub median problem was studied by Ilić et al. (2010). They proposed a Variable Neighborhood Search (VNS) where variable neighborhood descent (VND) based local search uses three different neighborhood structures. Another VNS was proposed by Jarboui et al. (2013) but for uncapacitated single allocation hub location problem. VND is used as local search and it consists of five neighborhood structures, including insertion and swap operators. Table 3.1 summarizes some key features in closely related contributions.

Table 3.1: A summary of main elements of the relevant contributions in literature.

Work	Allocation Scheme	No. Hubs	Objective	Capacity	Cycle Length	No. Vehicles	Solution method
Nagy and Salhi (1998)	pickup/delivery routes	endogenous	cost	yes	yes	endogenous	MIP + heuristic
Cetiner et al. (2006)	multiple allocation	endogenous	cost+fleet	no	yes	endogenous	heuristic
de Camargo et al. (2013)	single allocation	endogenous	cost	no	yes	endogenous	MIP + Benders decomposition
Wasner and Zäpfel (2004)	multiple allocation	endogenous	cost	yes	yes	endogenous	MIP + heuristic
Rodríguez-Martín et al. (2014)	single allocation	exogenous	cost	#. spoke per route	no	one per route	MIP + branch-and-cut
Gelareh et al. (2013b)	single allocation	exogenous	cost+fleet	yes	multiple of weeks	variable	MIP+Lagrangian decomposition
Gelareh et al. (2015)	single allocation	$q = 3 \leq .. \leq p$	time (transit+transshipment)	yes	$\geq$ 2 spokes	one per route	MIP+branch-and-cut+Benders
current work	single allocation	exogenous	time (transit+transshipment)	yes	$\geq$ 2 spokes	one per route	MIP+Lagrangian Relaxation + Hyperheuristic

This work contributes to the state-of-the-arts as follows: First, we present a

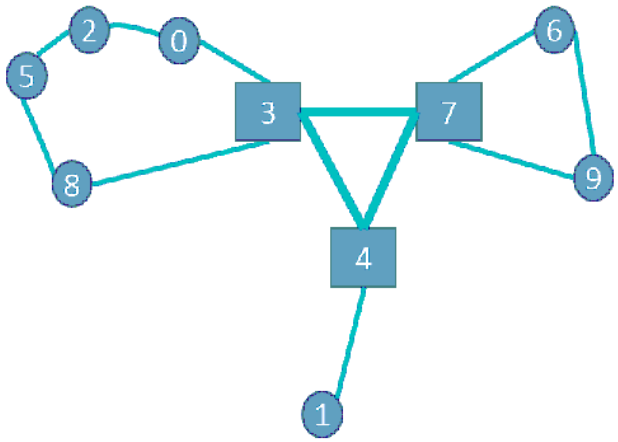

Figure 3.2: A solution of network with 3 hubs and 10 nodes.

mixed integer linear formulation with 2-index design variables that is basically similar to the model in Gelareh et al. (2015) and we proposed a 3-indexed design formulation where the number of hubs is fixed. We then propose a particular Lagrangian relaxation of the problem allowing to exploit the reduced cost of spoke level variables. The outputs of this relaxation scheme are then used in our proposed hyperheuristic framework. To the best of our knowledge, this is the first work tackling a variant of hub location problems using a hypereutectic approach (in particular exploiting Lagrangian relaxation information in hyperheuristic). The proposed hyperheuristic is comprised of a portfolio of low level heuristics (some of which use the Lagrangian relaxation information) and a *heuristic selection* method that learns in the course of process how to choose among the existing heuristics the one leading to a higher likelihood of success and improvement. The later is in fact a reinforcement learning method that has been inspired from the concept of

association rules in the *business data mining* world. To further expand our research, we have also tackled the problem presented by Rodríguez-Martín et al. (2014). For this problem, we propose a hyperheuristic method with another method of reinforcement learning methods called Q-learning, in order to guide in selection heuristic to be applied during the search.

The two aforementioned problems are distinct in the sense that as the first one is a transportation problem, the capacity concerns the volume of flow on the arcs (spoke-level arc) while the second treats the capacity as the maximum number of spokes along a cycle presenting the limited number of ports on switches, routers etc. in a telecommunications network. We denote the first problem as CSApHLRP-1, and the second CSApHLRP-2.

The remainder of this paper is organized as in the following: In section 3.2, we propose a mathematical model is a 3-index design MIP and present the 2-index design formulation for CSApHLRP-1 in Gelareh et al. (2015). Than we present the mathematical formulation of the problem in Rodríguez-Martín et al. (2014), CSApHLRP-2. The section 3.3 is divided into two parts. In the first part we propose five Lagrangian relaxations for the CSApHLRP-1 2-index design model, which is capable of offering approximation of reduced costs of the spoke-level network variables. In the second part a hyperheuristic solution approach and its components exploiting information of Lagrangian dual to guide are presented for the CSApHLRP-1 problem, and another one for the CSApHLRP-2 problem. Computational experiments and discussions are reported in section 3.4. Finally, in section 3.5, we conclude our work and present the possible future work.

3.2 Mathematical Formulation

We present two mathematical models for the first variant of HLRP problem. For the first problem, CSApHLRP-1, we propose a 3-index design variable mathematical formulation CSApHLRP-1-F1, and we present another model proposed by Gelareh et al. (2015) with 2-index design variable, CSApHLRP-1-F2.

Considering CSApHLRP-2, we present the model proposed by Rodríguez-Martín et al. (2014).

The following parameters and variables are used in models, CSApHLRP-1-F1 and CSApHLRP-1-F2. The only difference in variables in these two models, refers to variable r, which will be declared as 2-index or 3.

Table 3.2: CSApHLRP-1-F1 and CSApHLRP-1-F2 Models Parameters.

w_{ij}:	the flow from i to j,
t_{ij}:	the distance/time on a direct link on the edge (arc) $i-j$,
α:	the factor of economies of scale (the factor of travel time efficiency over hub edges),
p:	the upper bound on the number of hubs (depots),
Γ:	the minimum number of spokes allocated to each hub/depot node,
C^v:	the capacity of each vehicle for each feeder network,
φ^k:	the (fixed) average transshipment time at hub k.

Table 3.3: The decision variables for CSApHLRP-1-F1 and CSApHLRP-1-F2.

x_{ijkl}:	the fraction of flow from i to j traversing inter-hub edge $\{k, l\}$,
s_{ijkl}:	the fraction of flow from i to j traversing non-hub edge $\{k, l\}$,
$r_{ij}(r_{ijk})$:	1, if the arc (i, j) belongs to a spoke-level route (i and j are allocated to hub k in CSApHLRP-1-F1), 0 otherwise.
z_{ik}:	1, if the node i is allocated to node k where k is a hub, 0 otherwise.

3.2.1 (CSApHLRP-1-F1)

(CSApHLRP-1-F1)

$$\min \sum_{i,j,k,l} (t_{kl}(s_{ijkl} + \alpha x_{ijkl})) + \sum_{i,j,k,l:(k\neq i \vee l \neq j)} (\varphi^k + \varphi^l) x_{ijkl} \tag{3.1}$$

$s.\,t.$

$$\sum_{k} z_{kk} \leq p \tag{3.2}$$

$$\sum_{l} z_{kl} = 1 \qquad \forall k \in V \tag{3.3}$$

$$r_{ijk} \geq \Gamma z_{kk} \qquad \forall k \tag{3.4}$$

$$\sum_{j} r_{ijk} = z_{ik} \qquad \forall i, k \tag{3.5}$$

$$\sum_{j} r_{jik} = z_{ik} \qquad \forall i, k \tag{3.6}$$

$$\sum_{j:l\neq j} r_{jlk} = \sum_{j:l\neq j} r_{ljk} \qquad \forall l, k \tag{3.7}$$

$$r_{ijk} + r_{jik} \leq z_{ik} \qquad \forall i, j, k : j \neq i \tag{3.8}$$

$$r_{ijk} + r_{jik} \leq z_{jk} \qquad \forall i, j, k : j \neq i \tag{3.9}$$

$$z_{ik} \leq z_{kk} \qquad \forall i, k \in V : k \neq i \tag{3.10}$$

$$\sum_{k\neq i} (x_{ijik} + s_{ijik}) = 1, \qquad \forall i, j \in V : j \neq i \tag{3.11}$$

$$\sum_{l\neq j} (x_{ijlj} + s_{ijlj}) = 1, \qquad \forall i, j \in V : j \neq i \tag{3.12}$$

$$\sum_{l\neq i,k} (x_{ijkl} + s_{ijkl}) = \sum_{l\neq j,k} (x_{ijlk} + s_{ijlk}), \qquad \forall i, j, k \in V,\ k \notin \{i, j, \} \tag{3.13}$$

$$\sum_{l\neq k} x_{ijkl} \leq z_{kk} \qquad \forall i, j, k \in V : j \neq i, k < l \tag{3.14}$$

$$\sum_{l \neq k} x_{ijlk} \leq z_{kk} \qquad \forall i, j, k \in V : j \neq i, k < l \tag{3.15}$$

$$s_{ijkl} \leq \sum_{m} r_{klm} \qquad \forall i, j, k, l \in V : l \neq k \tag{3.16}$$

$$\sum_{ijl:j \neq i} w_{ij} s_{ijkl} \leq C, \qquad \forall k \in V \tag{3.17}$$

$$r \in \mathbb{B}^{|V|^2}, z \in \mathbb{B}^{|V| \times |V|}, \quad x_{ijkl}, s_{ijkl} \in\in \mathbb{R}^{|V|^4}_{[0,1]} \tag{3.18}$$

The objective function (3.1) is comprised of two parts; the first accounts for the total transportation times and the second part is the transshipment times. The transportation part is composed of travel times on the hub-level edges as well as spoke-level arcs. The travel time on the hub-level edges is discounted by α because the transporters offer faster services. The transhipment time is measured twice for every O-D flow; once at the first hub visited along O-D path and once at the last hub along the same path. Constraints (3.2) sets a limit on the number of hub nodes (depots) that can be installed while constraints (3.3) guarantee that every node is allocated to exactly one hub depot. A self-allocation of i (i.e. $z_{ii} = 1$) represents a hub depot i. If a node is designated as a hub node, there must be at least Γ nodes (including itself) allocated to it (making the route) as stated in (3.4). Here, we assume that $\Gamma = 3$. Constraints (3.5) (3.6) ensure that every spoke node that is assigned to a hub node will not be part of more than one route (i.e. will be

part of exactly one route). Constraints (3.7) ensure that, the number of arcs arriving to a spoke node is equal to the number of outgoing ones. Constraints (3.8)-(3.9) ensure that a leg on a given route can be established only if both end-nodes are allocated to the same hub depot. A spoke node can only be allocated to a hub node as in constraints (3.10). Constraints (3.11)-(3.13) stand for the flow conservation for every O-D pair. Constraints (3.11)-(3.15) state that traversing a hub edge is equivalent to traversing an edge where both end-points are hub nodes. Constraints (3.16) ensure that the flows on the route (spoke) edges will traverse in the correct direction and on an existing feeder edge. Constraints (3.17) ensure that the volume of flow on every leg of the routes associated to the hub nodes is constrained to the capacity of vehicles. Given that the flow leaving a spoke node will traverse a unique link encompassed from that spoke node, the term on the left side determines the whole flow leaving the spoke node k no matter originated from k itself or passing through it.

3.2.2 (CSApHLRP-1-F2)

A 2-index (design variables) was proposed in Gelareh et al. (2015) for the Bounded Cardinality Capacitated Hub Routing Problem (BCCHRP). In this model, definition of variable r does not indicate the allocation for end-nodes to any hub. More precisely, r_{ij} does not determine to which hub it belongs. This somehow helps to get rid of some of the constraints in the preceding model, however, in order to make sure that both end-points belong to the same hub node one needs to add some additional constraints.

(CSApHLRP-1-F2)

$$\min \sum_{i,j,k,l} (t_{kl}(s_{ijkl} + \alpha x_{ijkl})) + \sum_{i,j,k,l:(k\neq i \vee l \neq j)} (\varphi^k + \varphi^l) x_{ijkl} \tag{3.19}$$

$s.\,t.$

$$\sum_k z_{kk} \leq p \tag{3.20}$$

$$\sum_l z_{kl} = 1 \qquad \forall k \in V \tag{3.21}$$

$$z_{ik} \leq z_{kk} \qquad \forall i,k \in V : k \neq i \tag{3.22}$$

$$\sum_i z_{ik} \geq \Gamma z_{kk} \qquad \forall k \in V \tag{3.23}$$

$$\sum_{j\neq i} r_{ij} = 1 \qquad \forall i \in V \tag{3.24}$$

$$\sum_{j\neq i} r_{ji} = 1 \qquad \forall i \in V \tag{3.25}$$

$$r_{ij} + r_{ji} \leq 2 - z_{ik} - z_{jl} \qquad \forall i,j,k,l \in V : j \neq i, k \neq l \tag{3.26}$$

$$r_{ij} + r_{ji} \leq 1 \qquad \forall i,j \in V : j \neq i, \tag{3.27}$$

$$\sum_{k\neq i}(x_{ijik} + s_{ijik}) = 1, \qquad \forall i, j \in V : j \neq i, \tag{3.28}$$

$$\sum_{l\neq j}(x_{ijlj} + s_{ijlj}) = 1, \qquad \forall i, j \in V : j \neq i, \tag{3.29}$$

$$\sum_{l\neq i,k}(x_{ijkl} + s_{ijkl}) = \sum_{l\neq j,k}(x_{ijlk} + s_{ijlk}), \qquad \forall i, j, k \in V,\ k \notin \{i, j, \}\ , \tag{3.30}$$

$$\sum_{l\neq k} x_{ijkl} \leq z_{kk} \qquad \forall i, j, k \in V : j \neq i, k < l \tag{3.31}$$

$$\sum_{l\neq k} x_{ijlk} \leq z_{kk} \qquad \forall i, j, k \in V : j \neq i, k < l \tag{3.32}$$

$$s_{ijkl} \leq r_{kl} \qquad \forall i, j, k, l \in V : l \neq k \tag{3.33}$$

$$\sum_{ijl:j\neq i} w_{ij} s_{ijkl} \leq C, \qquad \forall k \in V \tag{3.34}$$

$$r \in \mathbb{B}^{|V|^2}, z \in \mathbb{B}^{|V|\times|V|},\ \ x_{ijkl}, s_{ijkl} \in\in \mathbb{R}^{|V|^4}_{[0,1]} \tag{3.35}$$

The objective function (3.19) and all constraints (3.20)- (3.22) remain the same as in the previous model. Constraints (3.23), are in fact equivalent to the constraints (3.4) in terms of z variables. Constraints (3.24) and (3.25) ensure that one arc

departs and one arc arrives at every node, respectively. Constraints (3.26), ensure that a spoke arc cannot exist if its end-points are allocated to different hubs and constraints (3.27) guarantee that an arc can appear only in either directions. The flow conservation constraints (3.28))-(3.32) are the same as in the previous model. Constraints (3.33) make sure that a spoke flow will traverse an existing spoke arc. The capacity constraints ((3.34)) are the same as in the previous model. It must be noted that in the 2-indexed formulation, the constraints (3.8)-(3.9) are no longer applicable. Instead, we needed to introduce constraints (3.26). Briefly speaking, approximately $2n^3$ constraints and $(n-1)(n^2-n)$ variables are removed in favor of $2n(n-1)+2n^2(n-1)^2$ constraints and $n(n-1)$. Moreover, less number of integer variables in the primal is expected to facilitate resolution and convergence of a branch-and-bound-based method.

3.2.3 (CSApHLRP-2)

Let $E = \{[i,j] : i,j \in \mathrm{V}, i \neq j\}$, the following parameters and variables are used in this model.

Table 3.4: CSApHLRP-2 Model Parameters.

V	set of nodes,
w_{ij}	the traffic demand from i to j,
c_{jl}	cost of routing from node j to l,
o_i	total amount of demand originate at node i,
d_i	total amount of demand with destination at node i,
q	the maximum number of nodes assigned spokes to a hub,
f_e	cost of using the edge $e \in E$ in a cycle,
β	factor of changing the relative weight of the cycle edge costs in the objective function.

Table 3.5: CSApHLRP-2 Decision Variables.

x_e	edge variable in a cycle with at least three edges,
$\delta(S)$	set of edges with exactly one end point in S and $E(S)$
z^1_{jj}	1, if node j $\in V$ is a hub and no other node is assigned to j,
z^1_{ij}	1, if node $j \in V$ is a hub and i is assigned to j with $i \neq j$,
z^2_{ij}	1, if node $j \in V$ is a hub and i is assigned to j,
g^i_{jl}	Amount of traffic that originates at node $i \in V$ and travels from hub $j \in V$ to hub $l \in V - \{j\}$

(CSApHLRP-2)

$$\min \sum_{i \in V} \sum_{j \in V-\{j\}} (c_{ij}o_i + c_{ji}d_i)\left(z^1_{ij} + z^2_{ij}\right) +$$

$$\alpha \sum_{j \in V} \sum_{l \in V-\{j\}} c_{jl} \sum_{i \in V} g^i_{jl} + \beta \left(\sum_{i \in V} \sum_{j \in V-\{j\}} 2f_{ij}z^1_{ij} + \sum_{e \in E} f_e x_e \right) \tag{3.36}$$

$s.t.$

$$\sum_{j \in V-\{i\}} z^1_{ij}z^1_{ii} + \sum_{j \in V-\{i\}} z^1_{ji} + \sum_{j \in V} z^2_{ij} = 1 \qquad \forall i \in V \tag{3.37}$$

$$\sum_{i \in V} z_{ij}^2 \leq q z_{jj}^2 \qquad \forall j \in V \tag{3.38}$$

$$\sum_{j \in V} \left(\sum_{i \in V} z_{ij}^1 + z_{jj}^2 \right) = p \tag{3.39}$$

$$\sum_{l \in V - \{j\}} g_{jl}^i - \sum_{l \in V - \{j\}} g_{lj}^i = \sum_{m \in V - \{i,j\}} \left(z_{ij}^1 + z_{ij}^2 - z_{mj}^1 - z_{mj}^2 \right) + w_{ij} \left(z_{ij}^1 + z_{ij}^2 - \sum_{k \in V} z_{kj}^1 - z_{jj}^2 \right) \qquad \forall i, j \in V, i \neq j \tag{3.40}$$

$$\sum_{l \in V - \{j\}} g_{jl}^j - \sum_{l \in V - \{j\}} g_{lj}^j = \sum_{m \in V - \{j\}} \left(\sum_{k \in V} z_{kj}^1 + z_{jj}^2 - z_{mj}^1 - z_{mj}^2 \right) \qquad \forall j \in V \tag{3.41}$$

$$x(\delta(i)) = 2 \sum_{j \in V} z_{ij}^2 \qquad \forall i \in V \tag{3.42}$$

$$x(\delta(S)) \geq 2 \sum_{j \in V - \{S\}} z_{ij}^2 \qquad \forall S \subset V, i \in S \tag{3.43}$$

$$x_{ii'} + z_{ij}^2 + z_{i'j'}^2 \leq 2 \qquad \forall [i, i'] \in E, j, j' \in V, j \neq j' \tag{3.44}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{3.45}$$

$$z_{ij}^1 \in \{0, 1\} \qquad \forall i, j \in V \tag{3.46}$$

$$z_{ij}^2 \in \{0, 1\} \qquad \forall i, j \in V \tag{3.47}$$

$$g_{jl}^i \geq 0 \qquad \forall i, j \in V, l \in V - \{j\} \tag{3.48}$$

In order to have a feasible solution, we need $pq \geq |V|$. $c_{ij} o_i + c_{ji} d_i$ represents the cost of assigning node i to hub j. Where i and j are hubs then the routing cost is discounted by a factor of α, the same as the previous problem. Two nodes form

a cycle with a cost of $2f_{ij}$. The objective function (3.36) accounts for the total installation cost of the edges and the routing of flows. Constraints (3.37) ensure that a node i can be in one of the four different situations: 1) the unique node allocated to another hub (case $\sum_{j\in V-\{i\}} z^1_{ij} = 1$), 2) a hub without other node allocated to it (case $z^1_{ii} = 1$), 3) it is a hub with one other node specified to it (case $\sum_{j\in V-\{i\}} z^1_{ji} = 1$), 4) a hub is specified to another hub at least two other nodes (case $\sum_{j\in V} z^2_{ij} = 1$). Constraints (3.38) are capacity constrains to guarantee that a cycle does not contain more than q nodes. p is the number of hubs to open in constraint (3.39). The flow balance constraints for the traffic on the hub network is due to constraints (3.40) and (3.41). Constraints (3.42) states that two edges must be incident to a node that is on a ring containing at least two other nodes. Constraints (3.43) maintain the connectivity of the cycles. In case, node $i \in S$ is allocated to a hub in set V/S, then the cycle which holds node i has to cross the cut declared by subset S and $x(\delta(S)) \geq 2$. Constraints (3.44) prohibit nodes allocated to different hubs to be on the same cycle.

It must be note that, the constraints (3.36)-(3.48) is a model for the SApHMP (Single Allocation p Hub Median Problem) as $\beta = 0$.

3.3 Solution algorithm

The proposed solution algorithm, for CSApHLRP-1, is a lower/upper bound solution method. We examine several Lagrangian relaxation schemes of this problem

with respect to the quality of their bounds. Each relaxation behaves independently regarding the convergence of dual multipliers as well as development of Lagrangian relaxation objective function variable coefficients. Therefore, the information stored in objective function variable coefficients can be used to guide different heuristic algorithms that are part of a hyperheuristic framework.

3.3.1 Lagrangian relaxation

Lagrangian relaxation for solving (mixed) integer programming problems was first proposed in Geoffrion (1974); Geoffrion and Bride (1978) and later in Fisher (1981, 2004). The idea behind this method is to relax *complicating constraints* by penalizing the objective function upon violation of these constraints. The relaxed problem is expected to be easier to solve than the original problem and provides a dual bound on the optimal value (as well as valuable information about the dual) of the problem (see Guignard (2003) for a comprehensive survey of the method.). Three well-known methods are commonly practiced in the literature for solving the Lagrangian relaxation problems. The oldest and most well-known one is the *subgradient* method as an iterative method for solving convex minimization problems. The subgradient method was originally proposed in the 60's in the former Soviet Union. Almost at the same time, a very similar method has been also proposed in Held and Karp (1971) for solving traveling salesman problem. Later, Lemarechal (1975) proposed the well-known bundle methods as an extension to the simple subgradient. The volume algorithm was later proposed in Barahona

and Anbil (2000) as a methods which simultaneously produces a primal feasible solution and a dual bound for the problem. A further analysis of volume algorithm and its relation with bundle methods has been studies in Bahiense et al. (2002).

Several variants of the subgradient-based methods have been proposed in the literature. Here, based on some observation from our preliminary computational experiments, we opted to use the well-known bundle method of Frangioni (1995) and the corresponding code (publicly available).

Dualizing some constraints of a MIP model, in Lagrangian fashion, may result in a Lagrangian problem that is easier to solve and its optimal value is a lower bound on the original problem (case of minimization problems) Fisher (2004). In brief, Lagrangian relaxation decomposes constraints of the problem into two separate group:1) The first group, usually named as hard constraints, which are dualized (are transferred and penalized in the objective function), and 2) The remaining subset of constraints that remain in the subproblem.

Let u^*_*, u^*_{**} and u^*_{***} are the Lagrangian multipliers with the proper dimensions and sign (unsigned for the equality constraints and nonnegative for the inequalities, i.e. $\leq$). In the following, we present five different lagrangian relaxations for the mathematical model of in CSApHLRP-1.

LRX1: design relaxation

The aim of this relaxation is 1) to accommodate the hub location variables (z_{kk}) in the objective function problem with a cost as a function of Lagrangian multipliers,

2) to obtain a subproblem that can be efficiently solved using a general-purpose MIP solver, and 3) determining the values of flow variables by inspection.

The new costs for design variables are calculated by $-\sum_{ij:j\neq i}(u^4_{ijk}+u^5_{ijk})$ for every z_{kk}. These costs are going to be used as *guiding information* in the heuristic algorithm to bias the search space exploration.

$$
\begin{aligned}
min\ & \sum_{i,j,k,l}(t_{kl}(s_{ijkl}+\alpha x_{ijkl}))+\sum_{i,j,k,l:(k\neq i\vee l\neq j)}(\varphi^k+\varphi^l)x_{ijkl}+\\
& \sum_{i,j\neq i}u^1_{ij}\left(\sum_{k\neq i}(x_{ijik}+s_{ijik})-1\right)+\sum_{i,j\neq i}u^2_{ij}\left(\sum_{l\neq j}(x_{ijlj}+s_{ijlj})-1\right)+\\
& \sum_{i,j,k\neq(i,j)}u^3_{ijk}\left(\sum_{l\neq i,k}(x_{ijkl}+s_{ijkl})-\sum_{l\neq j,k}(x_{ijlk}+s_{ijlk})\right)+\\
& \sum_{ijk:j\neq i,k<l}u^4_{ijk}\left(\sum_{l\neq k}x_{ijkl}-z_{kk}\right)+\sum_{ijk:j\neq i,k<l}u^5_{ijk}\left(\sum_{l\neq k}x_{ijlk}-z_{kk}\right)\\
& \sum_{i,j,k,l\neq k}u^6_{ijkl}(s_{ijkl}-r_{kl})+\sum_k u^7_k(\sum_{ijl:j\neq i}w_{ij}s_{ijkl}-C)
\end{aligned}
\tag{3.49}
$$

$s.\,t.$

(3.20), (3.21), (3.22),

(3.23), (3.24), (3.25), (3.26),

(3.27)

$$
r\in\mathbb{B}^{|V|^2},\ z\in\mathbb{B}^{|V|\times|V|},\quad x_{ijkl},s_{ijkl}\in\in\mathbb{R}^{|V|^4}_{[0,1]}
\tag{3.50}
$$

LRX2: z-relaxed

This relaxation aims at accommodating all the z_{ij} (but no r_{ij}) variables in the objective function in addition to the already existing flow variables. In this way, the dual information will be contained in the coefficients of z_{ij} variables and such modified costs can be exploited in guiding the heuristic algorithm by allowing better identifying location of hub nodes and allocation of spoke nodes to the hub nodes.

$$
\begin{aligned}
min \sum_{i,j,k,l} (t_{kl}(s_{ijkl} + \alpha x_{ijkl})) + \sum_{i,j,k,l:(k\neq i \vee l \neq j)} (\varphi^k + \varphi^l) x_{ijkl} + \\
u^1_{ij} \left(\sum_k z_{kk} - p \right) + \sum_k u^2_k \left(\sum_l z_{kl} - 1 \right) + \sum_{ik:k\neq i} u^3_{ik} \left(z_{ik} - z_{kk} \right) \\
\sum_k u^4_k \left(\Gamma z_{kk} - \sum_i z_{ik} \right) + \sum_{ijk, j\neq i, k<l} u^5_{ijk} \left(\sum_{l\neq k} x_{ijkl} - z_{kk} \right) + \\
\sum_{ijk, j\neq i, k<l} u^6_{ijk} \left(\sum_{l\neq k} x_{ijlk} - z_{kk} \right)
\end{aligned}
\tag{3.51}
$$

$s.\,t.$

(3.24), (3.25), (3.26), (3.27),

(3.28), (3.29), (3.30),

(3.33), (3.34),

$$
r \in \mathbb{B}^{|V|^2}, z \in \mathbb{B}^{|V|\times|V|}, \quad x_{ijkl}, s_{ijkl} \in\in \mathbb{R}^{|V|^4}_{[0,1]} \tag{3.52}
$$

LRX3: r-relaxed

This relaxation aims at accommodating all the r_{ij} (but no z_{ij}) variables in the objective function in addition to the already existing flow variables. By doing so, the dual information will be contained in the coefficients of r_{ij} variables. Such information can partially describe the network and route structures and can be used to either constructing a partial initial solution or, in general, to guide the heuristic by a given frequency.

$$
\begin{aligned}
min \sum_{i,j,k,l} (t_{kl}(s_{ijkl} + \alpha x_{ijkl})) + \sum_{i,j,k,l:(k\neq i \vee l \neq j)} (\varphi^k + \varphi^l) x_{ijkl} + \\
\sum_{i,j\neq i} u^1_{ij} \left(\sum_{j\neq i} r_{ij} - 1 \right) + \sum_{i,j\neq i} u^2_{ij} \left(\sum_{j\neq i} r_{ji} - 1 \right) + \\
\sum_{ijkl:k\neq l, j\neq i} u^3_{ijkl} \left(r_{ij} + r_{ji} - 2 + z_{ik} + z_{jl} \right) + \sum_{i,j\neq i} u^4_{ij} (r_{ij} + r_{ji} - 1) + \\
\sum_{ijkl,l\neq k} u^5_{ijkl} (s_{ijkl} - r_{kl}) +
\end{aligned}
\tag{3.53}
$$

$s.\,t.$

(3.20), (3.21), (3.22), (3.23)

(3.28), (3.29), (3.30)

(3.31), (3.32), (3.34)

$$
r \in \mathbb{B}^{|V|^2}, z \in \mathbb{B}^{|V|\times|V|}, \quad x_{ijkl}, s_{ijkl} \in\in \mathbb{R}^{|V|^4}_{[0,1]} \tag{3.54}
$$

LRX4: z, r**-relaxed**

The idea is almost the same as that of LRX2 except that in addition to the z_{ij} variables, all r_{ij} will also appear in the objective function and their modified costs can be used to guide the heuristic algorithms. Moreover, the resulting subproblem is a computationally inexpensive problem to solve using any of the general-purpose solvers (even often by inspection). Some partial information such as location of hub nodes, allocation of spokes to the routes and some edges along routes can be exploited by using this relaxation.

$$
\begin{aligned}
min \sum_{i,j,k,l} (t_{kl}(s_{ijkl} + \alpha x_{ijkl})) + \sum_{i,j,k,l:(k\neq i \vee l \neq j)} (\varphi^k + \varphi^l) x_{ijkl} + \\
u^1 \left(\sum_k z_{kk} - p \right) + \sum_k u_k^2 \left(\sum_l z_{kl} - 1 \right) + \sum_{i,k\neq i} u_{ik}^3 \left(z_{ik} - z_{kk} \right) \\
\sum_k u_k^4 \left(\Gamma z_{kk} - \sum_i z_{ik} \right) + \sum_i u_i^5 \left(\sum_{j\neq i} r_{ij} - 1 \right) + \sum_i u_i^6 \left(\sum_{j\neq i} r_{ji} - 1 \right) + \\
\sum_{ijkl, l\neq k, j\neq i} u_{ijkl}^7 \left(r_{ij} + r_{ji} - 2 - z_{ik} - z_{jl} \right) + \sum_{i,j\neq i} u_{ij}^8 \left(r_{ij} + r_{ji} - 1 \right)
\end{aligned}
\tag{3.55}
$$

$s.\,t.$

(3.28), (3.29), (3.30)

(3.31), (3.32), (3.33), (3.34)

$$
r \in \mathbb{B}^{|V|^2}, z \in \mathbb{B}^{|V|\times|V|}, \quad x_{ijkl}, s_{ijkl} \in\in \mathbb{R}_{[0,1]}^{|V|^4} \tag{3.56}
$$

LRX5: Lagrangian decomposition

The aim is to decompose the LR into two separate problems, one in the space of design variables z, r and another one in the space of flow variables x, s.

$$
\begin{aligned}
min \sum_{i,j,k,l} (t_{kl}(s_{ijkl} + \alpha x_{ijkl})) + \sum_{i,j,k,l:(k\neq i \vee l \neq j)} (\varphi^k + \varphi^l) x_{ijkl} + \\
\sum_{i,j\neq i,k,l\neq k} u^1_{ijkl} \left(r_{ij} + r_{ji} - 2 + z_{ik} + z_{jl}\right) + \sum_{i,j\neq i,k<l} u^2_{ijk} \left(\sum_{l\neq k} x_{ijkl} - z_{kk}\right) + \\
\sum_{i,j\neq i,k<l} u^3_{ijk} \left(\sum_{l\neq k} x_{ijlk} - z_{kk}\right) + \sum_{ijkl:k\neq l} u^4_{ijkl} \left(s_{ijkl} - r_{kl}\right)
\end{aligned}
\tag{3.57}
$$

$s.\,t.$

(3.20), (3.21), (3.22) (3.58)

(3.23), (3.24), (3.25), (3.27) (3.59)

(3.28), (3.29), (3.30), (3.34) (3.60)

$r \in \mathbb{B}^{|V|^2}, z \in \mathbb{B}^{|V|\times|V|}, \quad x_{ijkl}, s_{ijkl} \in\in \mathbb{R}^{|V|^4}_{[0,1]}$ (3.61)

3.3.2 Hyperheuristic for CSApHLRP-1 and CSApHLRP-2

Our proposed hyperheuristic-selection method is in fact a *reinforcement learning* method and has been inspired from the concept of *association rules* in the *business data mining* world (Aguinis et al., 2012).

In the following, we propose a hyperheuristic framework (based on the defin-

ition of Burke et al. (2010)) that: 1) is a heuristic-selection type (as opposed to heuristic-generation type), 2) applies an online learning, and 3) is a hybrid construction-perturbation heuristic.

The overall flow of algorithm is described in the following: First an initial solution is constructed using a constructive heuristic. Then through a learning process, the hyperheuristic proposes a series of perturbation, constructive and/or improvement heuristics in order to optimize the solution.

In the CSApHLRP-1, The first heuristic, which is a constructive one, optionally exploits the Lagrangian dual (for the first problem) information to construct the initial (probably partial) solution. In the next step, such a solution can be partially destructed and reconstructed, perturbed and improved, directly improved or any other combination. A potential improvement made by a transition from the first constructive heuristic to another heuristic is measured. An acceptance criteria then decides on the transition from the current state to another state.

Here, we use a simulated annealing move acceptance method, where the decision regarding the transition (move) to be accepted depends on the quality of solution obtained from applying heuristics (amount of improvement) and the elapsed CPU time. All improved moves are always accepted, and worsening moves might be accepted in accordance with the Metropolis criterion Kirkpatrick (1984):

$$\exp(-\Delta f/T) > R(0,1) \tag{3.62}$$

where Δf is the difference between the post-transition and pre-transition objective functions, T is a *synthetic temperature* and $R(0,1)$ is a uniform random number between 0 and 1. T represents the global temperature. When T is larger, many non-improving or even deteriorating transitions (moves) are accepted while when it tends to zero $\exp(-\Delta f/T)$ tends to zero, too. Therefore, for sufficiently small values of T, only improvement moves are accepted.

The algorithm starts with solving one or more (in parallel) Lagrangian relaxation(s) of the mathematical model (CSApHLRP-1-F2) using bundle algorithm (Frangioni, 1995). The outputs are in form of one (probably partially) feasible solution as well as *one* set of the coefficients of variables in the objective function, per relaxation.

In the CSApHLRP-2, the general scheme of the hyperheuristic has some similarity with the proposed hyperheuristic for CSApHLRP-1, expect it does not have any kind of information like results of Lagrangian Relaxation. In addition, the applied acceptance criteria is the so called Great Deluge. Great Deluge accepts solutions according to a dynamic threshold. We elaborate on further details in Chapter 1.

Low level heuristics

A rather large set of heuristics categorized in three classes, *constructive*, *improvement*, and *perturbation* are used within our hyperheuristic framework. In the following we describe every heuristic, proposed for CSApHLRP-1, with sufficient details.

Constructive heuristics build a solution from the scratch, based on several predefined rules. In this study, two constructive algorithms are implemented *CS1* and *CS2*.

CS1 that is detailed in Algorithm4, perform as in the following : it chooses p hubs that have the highest flow transshipment. Then it allocates at least Γ nearest spokes to every hub (by partially respecting the capacity constraint as the volume of demand/supply arriving/departing at the hub for the spoke on route must not violate the capacity). Finally, it incrementally constructs each feeder route by sequencing the spokes allocated to a given hub taking into consideration the capacity constraint.

The second constructive heuristic, *CS2*, is distinguished from the first one, i.e. *CS1*, in the way it chooses the set of p hubs candidates. *CS2* uses the information accumulated in the course of solving the third Lagrangian relaxation (LR3) (i.e. the coefficient of z_{ik} in LR3 objective function) as well as some statistics collected about the frequency that a given node appeared as a hub during the iterations of bundle method and the history of hyperheuristic process, if any. It then allocates at least Γ spokes to each hub then allocate remainder spokes. To construct the spoke-level route and sequencing the hub and its allocated spokes along it, we start from the hub and add the arcs incrementally. Let l be the last node already added to the partial spoke-level route and $\mathscr{S}_L$ be the set of remaining spokes to put on the route, unless $\mathscr{S}_L$ is a singleton, the next candidate spoke node, which we call it $R\_Nearest$, to be added to the existing partial route is $i^* =$

$\text{argmin}_{i \in \mathscr{S}_L}\{a_1 t_{s_l\ s_c} + a_2 r_{s_l\ s_c}\},\ 0 \leq a_1 + a_2 \leq 1, a_1, a_2 \geq 0.$ Where $t_{s_l\ s_c}$ represents the travel time between the last spoke s_l in the cycle of the feeder network allocated to h and the candidate spoke s_c. $r(s_l, s_c)$ corresponds to the r coefficient between s_l and s_c resulted from LR3. As long as a_2 increases, the influence of LR3 increases.

Algorithm 4 CS1

```
1:  procedure CONSTRUCTIVE1(𝒩,p,Sigma)
2:      Hubs = ChooseHubAccordingDemand()
3:      getFeeders(Hubs)
4:      for i = 1,...,p   (p = |ℋ|) do
5:          j=0
6:          while j < Sigma do
7:              f= R_Nearest(Hub[i],feeders,tabu)
8:              temp TempTabu ← f
9:              if CapacityConstraint(AssignFeederToHub(Hub[i],f)) then
10:                 AssignFeederToHub(Hub[i],f)
11:                 Tabu ← f
12:                 j = j + 1
13:             end if
14:         end while
15:         TempTabu = Tabu
16:     end for
17:     TempTabu = Tabu
18:     for i = 1,...,p   (p = |ℋ|) do
19:         j=0
20:         while true do
21:             f= R_Nearest(Hub[i],feeders,tabu)
22:             if f==Null then Break;
23:             end if
24:             TempTabu ← f
25:             if CapacityConstraint(AssignFeederToHub(Hub[i],f)) then
26:                 AssignFeederToHub(Hub[i],f)
27:                 Tabu ← f
28:                 j = j + 1
29:             end if
30:         end while
31:         TempTabu = Tabu
32:     end forreturn Hubs
33: end procedure
```

Improvement heuristics start from a complete solution and apply some moves in order to improve the objective function value. In our study, different *improvement* heuristics are proposed:

IMP 1) In *Rotate*, given a route (a hub and a sequence of spokes along the route), it rotates (clock-wise or counter clock-wise) the string. Such rotation of size k will designate the k-th spoke node after the current hub as a hub and renders the current hub as a spoke. This heuristic is useful when the order of nodes on the route is rather correct but the choice of hub is the cause of infeasibility or sub-optimality.

IMP 2) *Re-order* tries to re-sequence the nodes (hub and spokes) in order to find a better order of nodes along the route. While re-ordering, a sequence of $\{H, S_1, S_2, \ldots, S_m\}$ may turn to $\{S_3, S_1, H, S_m, \ldots, S_2\}$ wherein S_3 becomes a hub and H turns to a spoke node. This heuristic takes into the capacity on the route and the total volume of demand and supply of nodes such that by some intelligent re-sequencings, either a feasible solution is obtained or an already feasible solution is improved.

IMP 3) *Re-allocate* tries to remove an allocated port from a feeder network that has the maximum number of feeder nodes, and inserts it to another feeder network. This node can be a hub or spoke in its original route and can be equivalently hub or spoke in the devotional route. If it becomes hub at the destination then the hub at the destination turns to spoke on the route. Normally, such nodes can be the one in an *irreducible subset* of

the nodes on the route that are the cause of infeasibility or the nodes that are far from each other but lie on the same spoke-level arc (imposing high transportation time).

IMP 4) *SW1ffImp* randomly swaps two allocated ports from two different feeder networks, if and only if the solution improves the incumbent. This heuristic can be seen as a kind of search that tries to diversify and sample from different parts of the search space.

IMP 5) *SW2ffImp* randomly swaps tree ports belonging to three different feeder networks, if and only if the solution improves over the incumbent.

Perturbation heuristics start with a complete solution and do some changes in order to inject some diversification. In our study, we implement different perturbation heuristics in a manner of all changes will be in the solution must respect that to connect a pair of spokes i and j, they should have $r(j, j)$ less than a defined threshold $\mathcal{T}_H$:

PRT 1) *SW1ff*, The search tries to randomly swap two spokes allocated to the two different hubs In the end, it either returns the feasible solution with the maximum Hamming distance (with respect to the binary design variables) from the input solution or one of the solutions among the p farthest ones.

PRT 2) *SW2ff*, tries to randomly swap three spokes belonging to three different routes. It follows the same principle as in *SW1ff*, except that a very limited number ($n^2 < |.| \ll n^3$) of samples are examined as a complete 3-opt is very expensive,

PRT 3) *SW1hf*, swaps a random feeder and its corresponding hub. The hub becomes a spoke and the spokes turns to a hub.

PRT 4) *SW2hf* a fast local search that swaps a random feeder and a random hub which should not be its corresponding hub.

PRT 5) *SWhh*, a fast local search swapping two hubs. This can be done by taking into account the supply/demand of the entering hub that can violate the capacity (if we seek generating feasible solutions from perturbation).

PRT 6) *Insert/Delete*, removes an allocated spoke from a route, and inserts it to another one either with respect to the residual capacity available on another route or even randomly .

It must be emphasized that the *perturbation* heuristics aim at introducing some diversifications. Therefore, we do not expect a perturbed solution be a feasible one. Rather, we hope that it can help exploring unvisited regions in the solution space.

For the CSApHLRP-2, most of the components proposed (such as low-level-heuristics etc.) for CSApHLRP-1, are adapted for this problem. The main modification are:

1. The concept of edges is converted to arc in the feeder networks.
2. The problem has constraint capacity on the vertices rather than on the flow.
3. Add a check on the number of spokes allocated to each hub specially in *CS1*, *Insert/Delete* and *Re-allocate*.

4. LR3 results does not intervene in the process of *CS2*.

Heuristic Selection Methods

As mentioned in chapter 1, several heuristic-selection methods are introduced in the literature. In this study, we work with reinforcement learning mechanism in order to select the corresponding heuristic to be applied during the hyperheuristic process.

Proposed Learning for the CSApHLRP-1

Here, learning mechanism, which is presented in Algorithm 5, is inspired from a data mining technique called Association Rules (AR) guided by a Tabu Search (TS). When looking at the principles of AR, one can observe that AR is actually a kind of reinforcement learning that helps in respecting the dependency order of heuristic applications. The AR was first introduced by Agrawal et al. (1993) and is a technique in data mining that has attracted a lot of attention from the researchers and practitioners. Extracting interesting correlation, frequent patterns, associations or casual structures from group of items in the transaction databases or data repositories are the purpose of association rules technique. The technique of association rules finds interesting relationships among large set of data items. It shows attribute value conditions that occur frequently together in a given dataset. Traditionally, AR has been widely-used in Market Basket Analysis (Aguinis et al., 2012) in order to find how items purchased by customers are related . In association analysis, there are two sets of items (called *itemsets*): 1) the *antecedent* (the `if` part), and 2) the *consequent* (the `then` part). Moreover, an association rule has

two more values that express the degree of uncertainty about the rule. The first value is called the *support* for the rule. The support is defined as the proportion of task-relevant data transactions for which the pattern is true. The second one is known as the *confidence* of the rule. The so called confidence is a measure of *certainty* or *trustworthiness* associated with each discovered pattern. Here, our goal is to find some relationships between the different implemented heuristics, in order to find the best series of heuristics to be applied. The numerical reward is not going to be assigned to a single heuristic but to a series of them. Association Rules heuristic selection method deals with two main variables: *support* and *confidence*. These variables are used to measure the performance of a series of heuristics. If a heuristic does not have the required support at a certain time, or if it does not have enough confidence, the Tabu Search metaheuristic method prevents it from being selected. The relevant variables and parameters are explained in the following:

1. A heuristic series $(H_m, \ldots, H_n)$ weight (reward) is equal to the negative summation (for minimization case) of the objective function value divided by the number of the application of this heuristic.

2. A support of a suggested H_s is equal to the summation of the heuristic series $(H_1, \ldots, H_s)$ reward in the precedent transactions divided by the summation of all heuristic series rewards.

3. The confidence of a suggested H_s in such series $(H_m, \ldots, H_n)$ is equal to the support of H_s divided by the support item set $(H_m, \ldots, H_n)$.

Tabu list T_l: Given our time limit, we initially set $T_l = \emptyset$ and at each iteration, T_l is

updated as follows: if the elapsed CPU time is less than a certain milestone (e.g. the the time limit divided by a given number v_1), heuristics are chosen randomly in order to establish a historical heuristic series performance. Otherwise, if the elapsed CPU time is greater than the milestone $/v1$, T_l will contain all heuristics for each series expect those for which the support value is greater than a threshold th_S. In the case that the elapsed time is greater than the milestone $/v2$, T_l will contain all heuristics for each series expect those for which the confidence value is greater than a threshold th_C. It must be noted that the tabu list is updated with changes of quality of each heuristic series, i.e. calculation of W_{H_S}.

Algorithm 5 Association Rules Heuristic Selection Method

```
procedure CHOOSEHEURISTIC(Elapsed_Time)
    if (Elapsed_Time < V1) then
        Return Random(NbOfHeuristic)
    end if
    if (Elapsed_Time > V1) then
        while TRUE do
            Suggested_Heuristic_ID = Random(Nb_Of_Heuristic)
            S = Support(Suggested_Heuristic_ID)
            if (S > S_Threshold) then
                Return Suggested_Heuristic_ID
            end if
        end while
    end if
    if (Elapsed_Time > V2) then
        while TRUE do
            Suggested_Heuristic_ID = Random(Nb_Of_Heuristic)
            C = Confidence(Suggested_Heuristic_ID)
            if (C > C_Threshold) then
                Return Suggested_Heuristic_ID
            end if
        end while
    end if
end procedure
```

Proposed Learning for the CSApHLRP-2

From among several methods for solving the reinforcement learning problem, *temporal-difference* methods serve our purpose the best as they do not require a complete description of the environment and are fully incremental.

The well-known Q-learning algorithm Watkins (1989) which falls within this category uses the following update formula:

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}}$$

$$\cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

where R_{t+1} is the reward collected after taking action a_t in state s_t, $\alpha_t(s, a)(0 < \alpha \leq 1)$ is the learning factor (may be the same for all pairs) and $\gamma \geq 1$. wherein s is the current state, a is the action taken in the state s, r is the immediate reward received for executing action a in state s, s' is the new state, $\gamma \geq 1$, and $0 < \alpha_q < 1$ is the learning factor.

In a choice scheme as $\epsilon - greedy$ (see Watkins and Dayan (1992) for explanations and the proof of convergence), an agent will select the action resulting in the highest reward Q with probability $1 - \epsilon$ and a random action with probability ϵ.

The *learning rate* determines how the newly information will replace the old information. Two extreme cases are '0' in which the agent will not learn anything

and '1' in which the agent takes into account only the most recent information. Here we use $\alpha_t(s, a) = 0.1 \forall t$.

A discount factor γ close to 0 will make the agent short-sighted by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward.

A higher initial condition will encourage further exploration in any case of action update rule will cause it to have lower values than the other alternative and increases the probability of being chosen.

The process of Q-learning algorithm is shown in Algorithm 6.

Algorithm 6 Q-Learning Controller Algorithm

Inputs(set of state S, set of actions A, γ the discount factor, α the step size)
Inputs(real array Q[S,A], previous state s, previous action a)
repeat
    Select and Carry out an action a
    Calculate reward r
    get state s'
    $Q[s, a] \leftarrow Q[s, a] + \alpha(r + \gamma max_{a'} Q[s', a'] - Q[s, a])$
    s $\leftarrow$ s'
until Termination

Q-learning based hyperheuristic was applied by Falcao et al. (2015), in a problem of scheduling. Results proof that using Q-learning improve siginificantly the solution quality compared by solutions got from no-learning hyperheuristic.

Q-LA is used in secondary control to calculate microgrid regulation error (MRE), which is the regulated power, for real time operation. Economy and environmental benefits are so necessary in real time modification process of the generation sched-

ule of distributed generators (DGs) like batteries with the MRE by the fuzzy theory and particle swarm optimization method Xi et al. (2015). Q-learning is applied in Boyan and Littman (1994) to packet routing, where it is able to discovering efficiently different routing policies in a dynamic change of network with no information needed about the network topology, traffic patterns and etc. A modified version of Q-learning was proposed to plan pathes for mobile robots because it must be able to autonomous complete various intelligent tasks Gao et al. (2008). For interested researchers, Dorigo and Gambardella (2014), Ho et al. (2006), Choi and Yeung (1996) and Gaskett et al. (1999) are among good examples of work considering this reinforcement learning method.

3.4 Computational experiments

We have generated our instances based on the the well-known Australian Post (AP) dataset, for CSApHLRP-1. The transhipment times are generated randomly for some real values within [2, 5] for the given time unit. The existing capacities of the original data (see Ernst and Krishnamoorthy (1999)) are not used because they do not always results in feasible solutions as the problem structures are different. In CSApHLRP-2, the data sets was Civil Aeronautics Board (CAB), which is proposed by O'kelly (1987), and based on airline passenger flow among 25 important cities in the US (in addition to AP data set). CAB and AP nodes distributions shown respectively in Figure 3.3 and Figure 3.4.

All experiments were performed on an Intel 2.54 GHz core i5 CPU and 4 Gb of

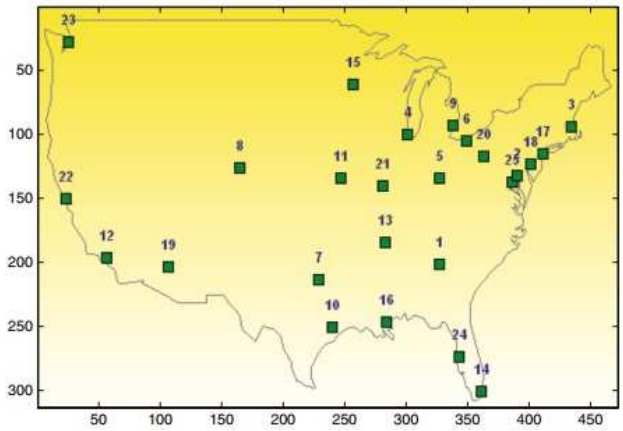

Figure 3.3: Civil Aeronautics Board data with 25 nodes.

memory running on Windows 7.

The termination criteria are chosen in such a way to avoid useless computation as well as premature convergence. The proposed termination criterion, sets a global time limit $T_{limit} = \max\{6000, n^2 \times p \times 250\}$ m.s. for the overall computation. In addition, we set a second termination criterion for terminating the whole algorithm once $10 \times n \times p$ non-improving iterations have been observed.

3.4.1 CSApHLRP-1 Computational experiments

We assume that our fleet of vehicles is homogenous and therefore with a unique capacity size. Let $\mathrm{O}_i = \sum_j W_{ij}$ and $\mathrm{D}_j = \sum_j W_{ij}$ and let p is the number of hubs. The capacity is generated as $C = \frac{\sum_i O_i + \sum_i D_i}{p}$.

All instances are named in a format ni_pj_k where i indicates the number of nodes, j indicates the number of hubs and k indicates the factor of economies of scale.

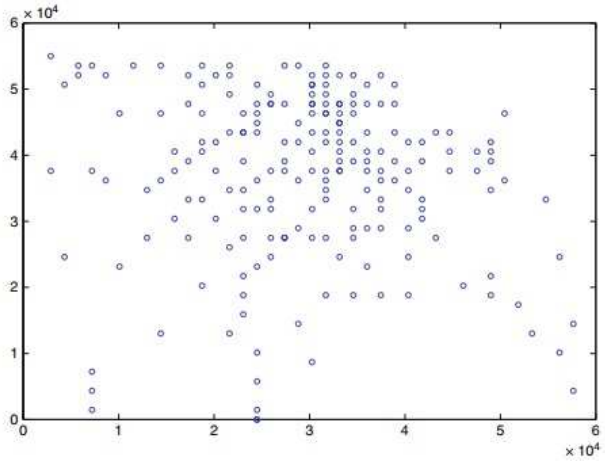

Figure 3.4: Australia post data with 200 nodes.

While a fair comparison between a heuristic algorithm and an exact decomposition is rather non-trivial, in Table 3.6, we report the results of our computational experiments with hyperheuristic (HH), with and without using results of LR, next to the results of Bender decomposition from Gelareh et al. (2015). There are, in total, 24 instances ranging from 10 nodes with 3 hubs to 20 nodes with 6 hubs. The first column reports the instance name, the second(resp. seventh) one indicates the computational times of Bender Decomposition method (resp. HH.).

The best solution found by HH without LR (resp. Bender Decomposition) are reported in the fifth (resp. third). The fourth column reports the Cplex status when apply Bender decomposition method. The gap between HH without LR results (resp. LR HH) and Benders Decompositions are reported in the seventh (resp. ten). The last column indicates the gap between the best objective function value taken by HH with and without using LR results.

One observes that the relative gaps between the solution reported by Benders decomposition and those of HH never exceeded 0.08% if HH use LR results, otherwise it does not exceed 0.32%. In this table, the negative gaps indicate that such HH solutions are better than the feasible solution obtained by applying the Benders decomposition method in Gelareh et al. (2015). When compared with the computational time elapsed to obtain such high quality solutions, it is confirmed that the proposed HH framework is capable of obtaining high quality solutions in very reasonable computational times, specially within LR results.

The gap between HH with and without LR results reaches 3.92%, proves the additional value of using the LR variable coefficient into the process of HH. LR variable coefficient shown in the objective function of LR3, appears very well during the process of HH. In construction phases (resp. in perturbation phase), all routes between any pairs have the coefficient greater than a number ι (resp. ν) are excluded. Furthermore, during the improvement phases pairs with small coefficients are favored to be connected.

Table 3.6: Comparison of the quality between benders decompositions and the proposed hyperheuristic with and without LR

instance	Bender Decompositio			Hypeheuristic without LR			HH with LR			Gap(% between HHs)
	Ex. Time(sec.)	Obj. Val.	CplexStatus	Obj. Val.	HH Ex. Time(sec.)	Gap(%) with BD	Obj. Val.	HH Ex.Time(sec.)	Gap(%) with BD	
n10_q3_0.7	17	3235.99	OptimalTol	3235.99	4.075	0.00	3235.99	3.11	0.00	0.00
n10_q3_0.8	14	3315.81	Optimal	3315.81	6.127	0.00	3315.81	5.01	0.00	0.00
n10_q3_0.9	14	3395.63	Optimal	3395.63	6.064	0.00	3395.63	5.01	0.00	0.00
n15_q3_0.7	929	11120.2	OptimalTol	10990.48296	5.003	-1.18	10990.48	4.10	-1.18	0.00
n15_q3_0.8	1395	11255.74	OptimalTol	11194.96	6.109	-0.54	10929.41	6.10	-2.99	2.37
n15_q3_0.9	398	11244.2	OptimalTol	10421.16	6.058	-7.90	10421.16	5.11	-7.90	0.00
n15_q4_0.7	1073	10683.44	OptimalTol	8532.44	6.06	-25.21	8198.31	6.08	-30.31	3.92
n15_q4_0.8	883	10170.06	OptimalTol	7966.52	5.076	-27.66	7966.52	4.10	-27.66	0.00
n15_q4_0.9	899	9067.25	OptimalTol	8949.29	6.089	-1.32	8949.29	6.11	-1.32	0.00
n15_q5_0.7	626	6899.27	Optimal	6921.75	6.068	0.32	6905.00	5.10	0.08	0.24
n15_q5_0.8	591	7143.31	Optimal	6921.75	5.064	-3.20	6921.75	4.19	-3.20	0.00
n15_q5_0.9	1615	7387.35	Optimal	7387.35	5.028	0.00	7387.35	5.00	0.00	0.00
n20_q3_0.7	15307	24814.09	OptimalTol	23380.95	6.06	-6.13	23380.95	5.09	-6.13	0.00
n20_q3_0.8	-	-	failed	22411.66225	6.044	-	22411.66225	5.10	-	0.00
n20_q3_0.9	28673	24935.81	OptimalTol	23316.54777	6.045	-6.94	23316.54777	5.11	-6.94	0.00
n20_q4_0.7	âĂİ	âĂİ	failed	19382.47479	8.11	-	19382.47479	7.99	-	0.00
n20_q4_0.8	14432	21284.45	OptimalTol	17519.51325	7.077	-21.49	17519.51325	7.00	-21.49	0.00
n20_q4_0.9	15767	21585.4	Optimal	21585.4	8.129	0.00	21585.40	7.01	0.00	0.00
n20_q5_0.7	6470	16933.63	Optimal	16933.63	9.26	0.00	16933.63	8.21	0.00	0.00
n20_q5_0.8	19572	17624.7	OptimalTol	16365.60	10.076	-7.69	16256.34	8.07	-8.42	0.67
n20_q5_0.9	19051	17441.26	Optimal	17441.26	10.146	0.00	17441.26	8.24	0.00	0.00
n20_q6_0.7	27531	15738.56	OptimalTol	14397.56	12.145	-9.31	14397.56	11.13	-9.31	0.00
n20_q6_0.8	46695	16922.59	AbortUser	14495.69	11.233	-16.74	14440.86	9.06	-17.19	0.38
n20_q6_0.9	98991	17274.12	AbortUser	14894.40	12.252	-15.98	14662.78	11.19	-17.81	1.56

Table 3.7 reports the total number of iterations every heuristic has worked on each instance. `Rotate` improvement heuristic has worked the most in all instances. Next, it comes to the constructive heuristics `CS1` and `CS2` which have more or less similar number of iterations for many instances. This may indicate that the information collected in the course of Lagrangian optimization in LR3, the statistics collected during HH process, and the construction by taking into account demand (i.e. `CS1`) when choosing the hubs, have some similarities. Among the perturbation heuristics `SW2hf` contributes in the most number of iterations.

Table 3.7: Number of each heuristic in each instance was applied.

instance	Insert/Delete	Re-Allocate	Re-order	Rotate	SW1ffImp	SW2ffImp	SW2hf	SW1ff	SW2ff	CS1	CS2	SWhh	SW1hf
n10_q3_0.7	3605	3568	3613	14331	3550	3641	3621	3570	3491	3542	3542	3553	3572
n10_q3_0.8	3864	3899	3900	15481	3861	3794	3898	3815	3846	3731	3769	3884	3843
n10_q3_0.9	3770	3895	3853	15165	3856	3786	3897	3796	3706	3821	3751	3868	3838
n15_q3_0.7	2513	2520	2458	9969	2470	2533	2533	2461	2427	2490	2464	2481	2500
n15_q3_0.8	2471	2534	2437	9938	2505	2551	2497	2458	2421	2443	2460	2501	2439
n15_q3_0.9	2466	2515	2482	9876	2394	2561	2483	2467	2436	2428	2437	2427	2461
n15_q4_0.7	2571	2519	2532	10129	2549	2621	2518	2469	2490	2494	2522	2550	2528
n15_q4_0.8	2381	2388	2348	9263	2348	2423	2412	2287	2252	2328	2334	2341	2296
n15_q4_0.9	2451	2509	2427	9824	2474	2529	2502	2391	2426	2483	2422	2454	2472
n15_q5_0.7	2561	2577	2562	10079	2511	2518	2554	2463	2474	2511	2561	2591	2491
n15_q5_0.8	2468	2418	2495	9898	2485	2576	2495	2401	2459	2467	2460	2494	2427
n15_q5_0.9	2483	2479	2469	9917	2514	2579	2478	2407	2449	2405	2442	2505	2443
n20_q3_0.7	1055	1072	1021	4148	1056	1058	1108	1013	1021	1046	1058	1036	1011
n20_q3_0.8	1195	1127	1105	4582	1160	1235	1181	1087	1115	1177	1179	1198	1093
n20_q3_0.9	1164	1124	1112	4602	1181	1194	1161	1126	1133	1118	1152	1158	1122
n20_q4_0.7	1642	1603	1601	6323	1612	1690	1682	1532	1569	1642	1625	1610	1605
n20_q4_0.8	1584	1601	1515	6242	1516	1578	1612	1530	1553	1602	1542	1531	1547
n20_q4_0.9	1672	1614	1581	6444	1500	1624	1693	1552	1578	1608	1587	1637	1554
n20_q5_0.7	1973	1956	1931	7716	1923	2041	1933	1884	1854	1939	1936	1978	1956
n20_q5_0.8	1974	1951	1926	7638	1902	2034	1962	1861	1831	1926	1934	1944	1968
n20_q5_0.9	1928	1973	1995	7687	1889	1951	1914	1928	1882	1892	1942	1942	1881
n20_q6_0.7	2211	2226	2233	8769	2193	2253	2214	2161	2115	2139	2181	2241	2123
n20_q6_0.8	2187	2156	2152	8650	2155	2241	2225	2154	2104	2162	2164	2189	2136
n20_q6_0.9	2156	2159	2172	8707	2148	2196	2172	2137	2147	2184	2123	2157	2067

We have applied the proposed HH framework on a wider ranger of problem instance sizes. Table 3.9 reports the statistics of applying the proposed HH on a tested composed of instances of up to 100 nodes.

In Table Table 3.8, initials objective functions of the HH with and without LR results are reported. In general initial objective function using LR results is better than without LR results. The average relative gap between the initial and the best found objective function value for instances of table without using LR results is 21.45% (resp. 19.36% using LR results) while this becomes 26.66% for the instances between greater than 20 nodes Table 3.9. This on one hand is due to a much larger solution space in larger instances which in turn increases the probability of finding better solutions during the search and on the other hand, may show that the HH is capable of benefiting from such a much larger search space to better improve the initial solutions. Another pessimistic view might question the performance of

Table 3.8: Comparison of the quality between the initial objective function taken by the proposed hyperheuristic with and without LR

instance	HH Initial Obj. Val.	LR Initial Obj. Val	Gap
n10_q3_0.7	4113.93	4113.93	0.00
n10_q3_0.8	3732.68	3732.68	0.00
n10_q3_0.9	4281.89	4281.89	0.00
n15_q3_0.7	11967.42	11967.42	0.00
n15_q3_0.8	14148.48	12075.56	14.66
n15_q3_0.9	14267.24	14267.24	0.00
n15_q4_0.7	12287.22	11194.78	8.89
n15_q4_0.8	12460.91	11373.32	8.73
n15_q4_0.9	11551.86	11551.86	0.00
n15_q5_0.7	7942.10	7942.10	0.00
n15_q5_0.8	9341.05	8160.33	12.64
n15_q5_0.9	7583.26	7583.26	0.00
n20_q3_0.7	30247.17	30093.41	0.51
n20_q3_0.8	28804.76	28804.76	0.00
n20_q3_0.9	29559.33	29559.33	0.00
n20_q4_0.7	28564.87	26512.51	7.18
n20_q4_0.8	29230.58	23481.05	19.67
n20_q4_0.9	24569.19	24569.19	0.00
n20_q5_0.7	22584.80	22584.80	0.00
n20_q5_0.8	24364.62	24364.62	0.00
n20_q5_0.9	21354.40	21354.40	0.00
n20_q6_0.7	15949.97	15949.97	0.00
n20_q6_0.8	17950.91	17950.91	0.00
n20_q6_0.9	18522.41	18522.41	0.00

construction heuristics and quality of initial solution constructed by them.

In order to ensure the quality of the solution, we reserve only information about the allocation resulted from LR HH and ignore all spokes routes information. The problem has some similarities with the picked-up delivery vehicle routing problem with multiple depots. Each hub is considered as depot, and the objective is to construct a feeder network for each hub from its allocated spokes. We inject all allocation information into solver, which find feeders network. All constraints related to the number of hubs and allocations was ignored. This process, which use some data resulted from a heuristic approach and complete the solution using a solver is referred to as *Matheuristic* chapter 1.

Results of matheuristic are reported in Table 3.10, within the gap with the LR HH objective function, and the execution time. All instances for which we

Table 3.9: HH results for large instances of pHub location problem.

instance	HH Initial Obj. Val.	HH Obj. Val.	HH Execution Time (sec.)
n30_p3_0.7	105144.53	75757.20	21.08
n30_p3_0.8	114888.19	76278.81	21.05
n30_p3_0.9	104911.22	72447.12	21.02
n30_p5_0.7	69112.47	53366.71	35.09
n30_p5_0.8	78602.46	54171.46	35.03
n30_p5_0.9	73163.24	54901.29	35.03
n40_p3_0.7	202842.00	164427.72	31.63
n40_p3_0.8	212202.17	170884.40	31.54
n40_p3_0.9	206603.16	178590.38	31.53
n40_p5_0.8	184960.11	132972.45	52.58
n40_p5_0.9	189692.00	117903.04	52.54
n40_p6_0.7	140120.51	106682.03	63.13
n50_p3_0.7	370022.47	283335.66	36.09
n50_p3_0.8	357808.84	290623.44	36.12
n50_p3_0.9	374514.09	225264.35	36.19
n50_p5_0.7	291375.19	208124.35	60.10
n50_p5_0.8	287717.05	208912.96	60.50
n50_p5_0.9	288860.51	232705.22	60.09
n70_p3_0.7	808406.46	638821.53	51.28
n70_p3_0.8	805167.64	617315.20	51.07
n70_p3_0.9	880752.26	617315.20	51.14
n70_p5_0.7	789712.18	484787.83	85.14
n70_p5_0.8	693946.36	542097.71	85.14
n70_p5_0.9	717687.40	551465.33	85.16
n80_p3_0.7	1129687.50	905846.24	61.64
n80_p3_0.8	1128080.39	919430.31	61.74
n80_p3_0.9	1172538.60	853701.87	61.73
n80_p5_0.7	902766.77	676355.62	102.69
n80_p5_0.8	908105.30	713127.24	102.84
n80_p5_0.9	949987.94	723792.60	102.77
n90_p3_0.7	1430318.33	1265909.60	66.10
n90_p3_0.8	1594559.93	1186128.92	66.29
n90_p3_0.9	1407958.75	1166788.83	66.38
n90_p5_0.7	1085625.40	959702.73	110.30
n90_p5_0.8	1288459.47	971146.25	110.34
n90_p5_0.9	1126831.55	988784.52	110.15
n100_p3_0.7	1679715.79	1484907.58	76.69
n100_p3_0.8	1692753.82	1491646.31	76.96
n100_p3_0.9	2061323.94	1512299.50	76.88
n100_p5_0.7	1839058.26	1251859.24	127.96
n100_p5_0.8	1499370.20	1291282.92	127.77
n100_p5_0.9	1450410.23	1275010.77	127.75
n100_p10_0.7	1095903.40	980333.22	255.55
n100_p10_0.8	1185961.32	967479.92	255.37
n100_p10_0.9	1143466.25	952694.14	255.30

do not have BD optimal solution or that have negative gap between HH and BD objective function were tested, in addition to some medium size instances. The gap reaches 2.66% in instance n20_p5_0.8, in addition it finds optimal solution for given allocations to some instances such as n15_p3_0.7, n15_q3_0.8, n15_q3_0.9, etc. The added value of *matheuristic*, which appears in instances less than 20 nodes, does not appear much pertinent in the medium instances within time limit of 1 hour. Table 3.10 report these results and showing the quality gained, while the

negative gaps proves the deteriorated quality by *matheuristic*.

Table 3.10: A comparison between results of Matheuristic with that of LR HH.

instance	Matheuristic Status	Time	OF	LR HH Obj. val.	GAP(%) LR HH and Math.
n10_q3_0.7	Optimal	0.30	3235.991	3235.99	0.00
n15_q3_0.7	Optimal	13.82	10990.48	10990.48	0.00
n15_q3_0.8	Optimal	40.30	10902.53	10929.41	0.25
n15_q3_0.9	Optimal	16.97	10276.65	10421.16	1.39
n15_q4_0.7	Optimal	1.76	8198.31	8198.31	0.00
n15_q4_0.8	Optimal	11.56	7966.52	7966.52	0.00
n15_q4_0.9	Optimal	2.00	8945.09	8949.29	0.05
n20_q3_0.7	Feasible	3605.57	23213.17	23380.95	0.72
n20_q3_0.8	Feasible	3605.70	22411.66	22411.66	0.00
n20_q3_0.9	Feasible	3605.73	23316.55	23316.55	0.00
n20_q4_0.7	Optimal	3675.79	19382.47	19382.47	0.00
n20_q4_0.8	Optimal	484.40	17519.51	17519.51	0.00
n20_q5_0.8	Optimal	7304.80	15824.26	16256.34	2.66
n20_q6_0.7	Optimal	12.45	14397.56	14397.56	0.00
n20_q6_0.8	Optimal	26.89	14321.04	14440.86	0.83
n20_q6_0.9	Optimal	33.88	14662.78	14662.78	0.00
n30_p3_0.7	Feasible	3672.64	80871.04	75757.20	-6.75
n30_p3_0.8	Feasible	4118.88	81845.67	76278.82	-7.30
n30_p3_0.9	Feasible	7756.06	78998.28	73284.34	-7.80
n40_p3_0.7	Feasible	3770.47	292132.10	164427.72	-77.67
n40_p3_0.8	Feasible	3787.60	352933.80	170884.40	-106.53
n40_p3_0.9	Feasible	3796.32	316753.50	178590.38	-77.36

In Table 3.11, we report results of applying iterative method for solving the five variants of Lagrangian relaxation methods proposed in subsection 3.3.1. The different relaxations were solved by using bundle. The bundle method is known for its higher precision and faster convergence compared to the simpler versions of subgradient method. We have initialized the Lagrangean multipliers with the corresponding dual values of the LP relaxation, wherever possible.

In Table 3.11, for every relaxation we have reported the CPU time as well as the best bound obtained for the given amount of CPU time. For some relaxations (for example LR2) the computational time for the very small instances was several order of magnitude higher than other relaxations and the bound quality was not really dominating. Therefore, we have not gone further with the larger instances as the run times were increasing exponentially. From among all five relaxations, only

LR3 was able to converge for the problem until size of 20 nodes in reasonable times. Furthermore, LR3 offers bests bounds among other relaxations. LR1 was able to solve problem up to 15 nodes and 4 hubs but for 15 nodes and 5 hubs, it could just terminated in reasonable time for $\alpha = 0.9$ from among other values. As mentioned earlier, LR2 is the most expensive relaxation in terms of computational time as it was able to solve instances of 10 nodes with 3 hubs in reasonable time. LR4 and LR5 appear effective in terms of computational time only until instances with 15 nodes and 5 hubs. When doing iterations of bundle algorithm, the subproblem was solved using CPLEX 12.6.2, the time limit was set to 3600 seconds and the number of iterations was set to 100.

Table 3.11: Lagrangian relaxation results.

instance	LR1		LR2		LR3		LR4		LR5		nbIterations
	Time (sec.)	BestLB	Time (sec.)	BestLB	Time (sec.)	BestLB	Time (sec.)	BestLB	Time (sec.)	BestLB	
n10 p3 0.9	231.69	2068.61	1987.52	2325.97	228.177	2384.16	23.317	2123.11	25.419	2049.61	100
n10 p3 0.8	195.81	2020.86	1806.25	2228.71	231.55	2319.98	22.153	2088.45	24.662	2066.57	100
n10 p3 0.7	182.26	1956.84	1537.42	2133.36	219.116	2239.29	21.091	2031.92	23.965	2032.45	100
n15 p3 0.9	993.1	5253.67	—	—	1665.9	5785.28	118.179	5352.14	113.771	5312.45	100
n15 p3 0.8	947.26	5089.8	—	—	1704.89	5732.76	107.336	5275.19	114.311	5380.57	100
n15 p3 0.7	1009.9	4969.5	—	—	1744.7	5580.06	105.078	5152.99	113.224	5293.16	100
n15 p4 0.9	2467.3	4975.18	—	—	1901.39	5593.87	103.942	4983.24	114.761	5038.77	100
n15 p4 0.8	2027.5	4842.1	—	—	2020.9	5418.13	112.094	5026.48	114.53	5071.04	100
n15 p4 0.7	2431.2	4688.62	—	—	1937.78	5253.88	108.545	4916.29	116.671	4957.92	100
n15 p5 0.9	51424	4770.47	—	—	2164.69	5332.59	104.984	4881.33	115.739	4787.32	100
n15 p5 0.8	—	—	—	—	2246.72	5134.72	104.874	4854.58	112.559	4738.18	100
n15 p5 0.7	—	—	—	—	2185.9	4992.37	105.495	4687.78	113.274	4648.12	100
n20 p3 0.9	—	—	—	—	10818	10239.6	—	—	—	—	100
n20 p3 0.8	—	—	—	—	10693	10066.4	—	—	—	—	100
n20 p3 0.7	—	—	—	—	10240	10014.9	—	—	—	—	100
n20 p4 0.9	—	—	—	—	12879	9933.42	—	—	—	—	100
n20 p4 0.8	—	—	—	—	13407	9636.57	—	—	—	—	100
n20 p4 0.7	—	—	—	—	12408	9426.4	—	—	—	—	100
n20 p5 0.9	—	—	—	—	22194	9586.66	—	—	—	—	100
n20 p5 0.8	—	—	—	—	19660	9389.53	—	—	—	—	100
n20 p5 0.7	—	—	—	—	15495	9188.13	—	—	—	—	100
n20 p6 0.9	—	—	—	—	33290	9366.45	—	—	—	—	100
n20 p6 0.8	—	—	—	—	23062	8991.41	—	—	—	—	100
n20 p6 0.7	—	—	—	—	17112	8652.79	—	—	—	—	100

3.4.2 CSApHLRP-2 Computational experiments

Instances was named in a format $Cabi\_j\_\alpha_\beta$ where i indicates the number of nodes, j indicates the number of hubs, α indicates the factor of economies of scale and β the factor of changing the relative weight of the cycle.

In Table 3.12, Table 3.13, Table 3.14 and Table 3.15, we report the results of branch and cut in (Rodríguez-Martín et al., 2014), next the HH results. The gap between CPLEX and HH never exceeds 1%. Furthermore, the average of the gap between initial objective and the best found is 21.01%, which inherits some similarity to CSApHLRP-1. Execution time for AP instances with 40 and 50 nodes, was show in Figure 3.8. It is clear that HH process for for these instances doesn't exceed 4.5 sec.

Figure 3.5 - Figure 3.7 sketch the optimal solution for a CAB instance, as proven in Rodríguez-Martín et al. (2014) and offered by the proposed HH, having 25 nodes, $q = 13$, α=0.8, β=0.01 or 0.05 or 0.2. Due to the influence of the objective function, while β increase while cycles try be more large.

3.5 Conclusion and future work

We proposed a hyperheuristic approach for solving two variants of Hub Location Routing Problems, CSApHLRP-1 and CSApHLRP-2. Given a set of nodes and a set of O-D demands, the problem seeks designating p hubs, allocating each non-hub (spoke) node to one hub while each hub lies on a Hamiltonian path comprised of

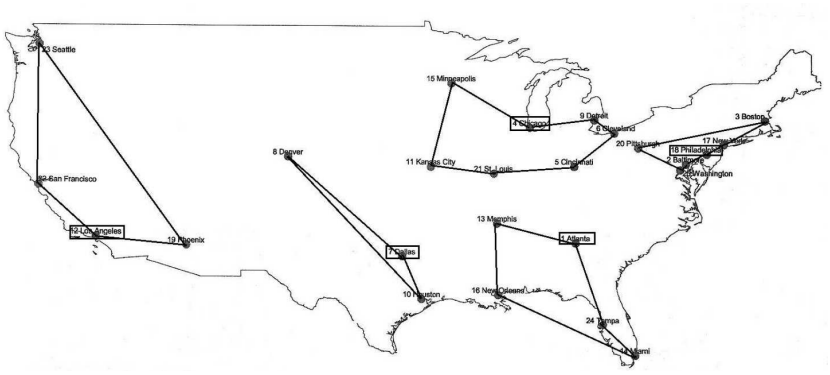

Figure 3.5: Optimal solution for CAB25 with p=5, q=13, α=0.8, β=0.01.

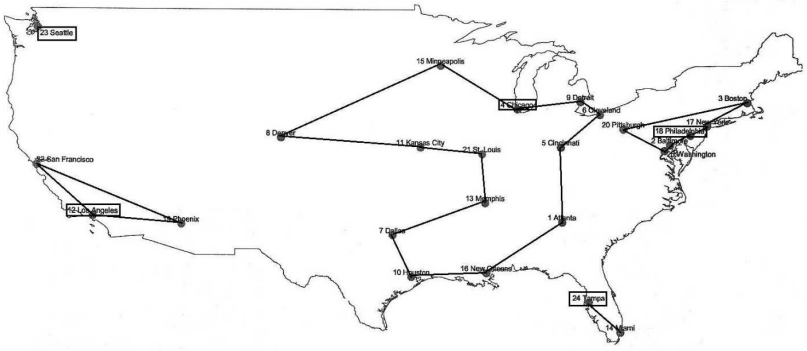

Figure 3.6: Optimal solution for CAB25 with p=5, q=13, α=0.8, β=0.05.

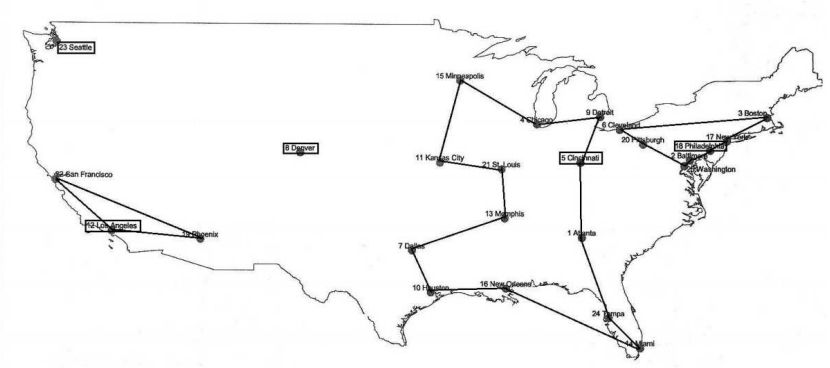

Figure 3.7: Optimal solution for CAB25 with p=5, q=13, α=0.8, β=0.2.

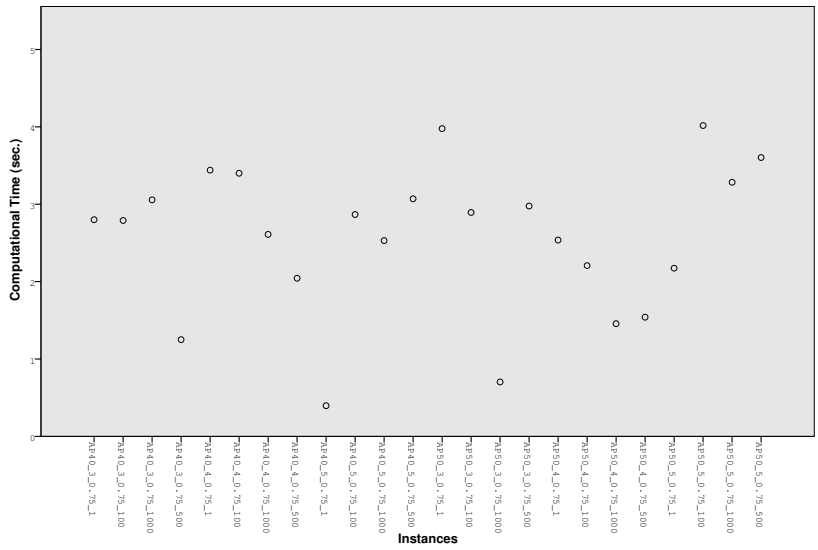

Figure 3.8: HH execution time in CSApHLRP-2.

the hub and its spokes. The allocation of nodes and the construction of Hamiltonian paths is done in such a way that the transported volume on the route respects a given capacity in CSApHLRP-1, and not exceed a certain number of spokes in CSApHLRP-2 and the hub-level network is a connected one. While exact methods have shown a very limited success, for CSApHLRP-1, when dealing with instances of problem and the instances of moderate size remain still very intractable, one has to resort to non-exact (heuristic-based) methods for obtaining solutions to the moderate-to-large size instances. We applied five Lagrangian Relaxation on the model of CSApHLRP-1, and we use the results of the best one in the process of HH. LR HH has proven in the numerical results as better than HH without LR results, even its initial fitness. We proposed hyperheuristic method for the two problem, in order to solve large instances. The proposed hyperheuristic consists

of several low level heuristics guided by means of a learning method, which is inspired from a topic in business data mining world known as *Association Rules* for CSApHLRP-1, and *Q-Learning* method for CSApHLRP-2.

Our numerical results, for the two problems, show that in the case of instances for which an optimal solution or a solution with known quality is known, the proposed method is capable of providing high quality solutions. In all cases, such solutions are obtained in a very reasonable CPU times. Therefore, it is expected that for larger size instances for which finding an optimal solution needs an effort beyond the capacity of current hardware resources, the proposed framework that is equipped with advances learning mechanisms performs sufficiently well.

In our future work we will add further aspects of real-life application to the model and consider solving problem taking into account heterogeneous fleet of vessels with different route capacities, inter-route transshipment etc.

Table 3.12: Comparison of the quality between results of HH and B&C Obj. Value Rodríguez-Martín et al. (2014), cab instance with 25 nodes

Instance	q	B&C Obj. Value Rodríguez-Martín et al. (2014)	HH. Obj. Value	Gap (%)
Cab25_3_0.2_0.01	|V|	858.76	858.76	0.00
Cab25_3_0.2_0.05	|V|	1193.41	1193.41	0.00
Cab25_3_0.2_0.2	|V|	2448.35	2448.35	0.00
Cab25_3_0.4_0.01	|V|	998.04	998.04	0.00
Cab25_3_0.4_0.05	|V|	1332.69	1332.69	0.00
Cab25_3_0.4_0.2	|V|	2587.63	2587.63	0.00
Cab25_3_0.8_0.01	|V|	1254.02	1254.02	0.00
Cab25_3_0.8_0.05	|V|	1605.91	1605.91	0.00
Cab25_3_0.8_0.2	|V|	2827.03	2827.03	0.00
Cab25_4_0.2_0.01	|V|	720.84	720.84	0.00
Cab25_4_0.2_0.05	|V|	1041.09	1041.09	0.00
Cab25_4_0.2_0.2	|V|	2227.04	2227.04	0.00
Cab25_4_0.4_0.01	|V|	876.30	876.30	0.00
Cab25_4_0.4_0.05	|V|	1206.25	1206.25	0.00
Cab25_4_0.4_0.2	|V|	2392.19	2392.19	0.00
Cab25_4_0.8_0.01	|V|	1176.44	1176.44	0.00
Cab25_4_0.8_0.05	|V|	1528.42	1528.42	0.00
Cab25_4_0.8_0.2	|V|	2615.26	2615.26	0.00
Cab25_5_0.2_0.01	|V|	626.71	626.71	0.00
Cab25_5_0.2_0.05	|V|	947.54	947.54	0.00
Cab25_5_0.2_0.2	|V|	2027.18	2027.18	0.00
Cab25_5_0.4_0.01	|V|	795.61	795.61	0.00
Cab25_5_0.4_0.05	|V|	1120.99	1130.89	0.88
Cab25_5_0.4_0.2	|V|	2179.65	2179.65	0.00
Cab25_5_0.8_0.01	|V|	1126.18	1126.18	0.00
Cab25_5_0.8_0.05	|V|	1446.56	1446.56	0.00
Cab25_5_0.8_0.2	|V|	2457.77	2457.77	0.00
Cab25_3_0.2_0.01	|V|/2	865.42	865.42	0.00
Cab25_3_0.2_0.05	|V|/2	1213.10	1223.16	0.82
Cab25_3_0.2_0.2	|V|/2	2495.76	2495.76	0.00
Cab25_3_0.4_0.01	|V|/2	999.62	999.62	0.00
Cab25_3_0.4_0.05	|V|/2	1359.94	1359.94	0.00
Cab25_3_0.4_0.2	|V|/2	2648.59	2648.59	0.00
Cab25_3_0.8_0.01	|V|/2	1254.02	1254.02	0.00
Cab25_3_0.8_0.05	|V|/2	1623.26	1623.26	0.00
Cab25_3_0.8_0.2	|V|/2	2917.72	2917.72	0.00
Cab25_4_0.2_0.01	|V|/2	720.84	727.50	0.92
Cab25_4_0.2_0.05	|V|/2	1041.09	1041.09	0.00
Cab25_4_0.2_0.2	|V|/2	2227.04	2227.04	0.00
Cab25_4_0.4_0.01	|V|/2	876.30	876.30	0.00
Cab25_4_0.4_0.05	|V|/2	1206.25	1206.25	0.00
Cab25_4_0.4_0.2	|V|/2	2392.19	2392.19	0.00
Cab25_4_0.8_0.01	|V|/2	1176.44	1176.44	0.00
Cab25_4_0.8_0.05	|V|/2	1528.42	1528.42	0.00
Cab25_4_0.8_0.2	|V|/2	2716.52	2716.52	0.00
Cab25_5_0.2_0.01	|V|/2	626.71	626.71	0.00
Cab25_5_0.2_0.05	|V|/2	947.54	947.54	0.00
Cab25_5_0.2_0.2	|V|/2	2040.02	2040.02	0.00
Cab25_5_0.4_0.01	|V|/2	795.61	795.61	0.00
Cab25_5_0.4_0.05	|V|/2	1120.99	1120.99	0.00
Cab25_5_0.4_0.2	|V|/2	2208.83	2208.83	0.00
Cab25_5_0.8_0.01	|V|/2	1126.18	1126.18	0.00
Cab25_5_0.8_0.05	|V|/2	1446.56	1446.56	0.00
Cab25_5_0.8_0.2	|V|/2	2514.93	2514.93	0.00
Cab25_3_0.2_0.01	|V|/p	943.25	943.25	0.00
Cab25_3_0.2_0.05	|V|/p	1348.93	1348.93	0.00
Cab25_3_0.2_0.2	|V|/p	2789.59	2789.59	0.00
Cab25_3_0.4_0.01	|V|/p	1089.05	1089.05	0.00
Cab25_3_0.4_0.05	|V|/p	1494.95	1496.75	0.12
Cab25_3_0.4_0.2	|V|/p	2926.27	2926.27	0.00
Cab25_3_0.8_0.01	|V|/p	1302.98	1302.98	0.00
Cab25_3_0.8_0.05	|V|/p	1708.64	1708.64	0.00
Cab25_3_0.8_0.2	|V|/p	3099.42	3099.42	0.00
Cab25_4_0.2_0.01	|V|/p	721.98	721.98	0.00
Cab25_4_0.2_0.05	|V|/p	1063.03	1063.03	0.00
Cab25_4_0.2_0.2	|V|/p	2341.94	2341.94	0.00
Cab25_4_0.4_0.01	|V|/p	881.26	881.26	0.00
Cab25_4_0.4_0.05	|V|/p	1222.30	1222.30	0.00
Cab25_4_0.4_0.2	|V|/p	2501.22	2501.22	0.00
Cab25_4_0.8_0.01	|V|/p	1178.69	1178.69	0.00
Cab25_4_0.8_0.05	|V|/p	1531.41	1531.41	0.00
Cab25_4_0.8_0.2	|V|/p	2810.33	2822.19	0.42
Cab25_5_0.2_0.01	|V|/p	686.85	686.85	0.00
Cab25_5_0.2_0.05	|V|/p	1050.38	1050.38	0.00
Cab25_5_0.2_0.2	|V|/p	2393.88	2393.88	0.00
Cab25_5_0.4_0.01	|V|/p	857.24	857.24	0.00
Cab25_5_0.4_0.05	|V|/p	1221.48	1221.48	0.00
Cab25_5_0.4_0.2	|V|/p	2564.98	2564.98	0.00
Cab25_5_0.8_0.01	|V|/p	1165.79	1165.79	0.00
Cab25_5_0.8_0.05	|V|/p	1532.11	1532.11	0.00
Cab25_5_0.8_0.2	|V|/p	2875.61	2885.14	0.33

Table 3.13: Comparison of the quality between results of HH and B&C Obj. Value Rodríguez-Martín et al. (2014), AP instance for 25 nodes

Instance	q	B&C Obj. Value Rodríguez-Martín et al. (2014)	HH. Obj. Value	Gap (%)
AP25_3_0.75_1	|V|	155482.14	155539.68	0.04
AP25_3_0.75_100	|V|	177838.26	177838.26	0.00
AP25_3_0.75_500	|V|	262544.57	262544.57	0.00
AP25_3_0.75_1000	|V|	366638.05	366638.05	0.00
AP25_4_0.75_1	|V|	139430.10	139773.20	0.25
AP25_4_0.75_100	|V|	161485.26	161485.26	0.00
AP25_4_0.75_500	|V|	243004.56	243004.56	0.00
AP25_4_0.75_1000	|V|	344903.68	344903.68	0.00
AP25_5_0.75_1	|V|	123802.90	123802.90	0.00
AP25_5_0.75_100	|V|	145099.06	145099.06	0.00
AP25_5_0.75_500	|V|	227204.68	227204.68	0.00
AP25_5_0.75_1000	|V|	327043.26	327043.26	0.00
AP25_3_0.75_1	|V|/2	155482.14	155482.14	0.00
AP25_3_0.75_100	|V|/2	177838.26	177838.26	0.00
AP25_3_0.75_500	|V|/2	262544.57	262591.10	0.02
AP25_3_0.75_1000	|V|/2	366638.05	366638.05	0.00
AP25_4_0.75_1	|V|/2	139430.10	139430.10	0.00
AP25_4_0.75_100	|V|/2	161485.26	161485.26	0.00
AP25_4_0.75_500	|V|/2	243004.56	243004.56	0.00
AP25_4_0.75_1000	|V|/2	344903.68	344903.68	0.00
AP25_5_0.75_1	|V|/2	123802.90	123802.90	0.00
AP25_5_0.75_100	|V|/2	145099.06	145099.06	0.00
AP25_5_0.75_500	|V|/2	227204.68	227204.68	0.00
AP25_5_0.75_1000	|V|/2	327043.26	327567.38	0.16
AP25_3_0.75_1	|V|/p	156287.34	156287.34	0.00
AP25_3_0.75_100	|V|/p	178328.06	178328.06	0.00
AP25_3_0.75_500	|V|/p	267381.48	267381.48	0.00
AP25_3_0.75_1000	|V|/p	376932.18	376998.36	0.02
AP25_4_0.75_1	|V|/p	139876.23	139876.23	0.00
AP25_4_0.75_100	|V|/p	161720.99	161720.99	0.00
AP25_4_0.75_500	|V|/p	249982.62	249995.77	0.01
AP25_4_0.75_1000	|V|/p	359669.90	359899.70	0.06
AP25_5_0.75_1	|V|/p	130727.14	130986.57	0.20
AP25_5_0.75_100	|V|/p	154151.28	154222.44	0.05
AP25_5_0.75_500	|V|/p	245105.99	247225.09	0.86
AP25_5_0.75_1000	|V|/p	357731.82	359951.33	0.62

Table 3.14: Comparison of the quality between results of HH and B&C Obj. Value Rodríguez-Martín et al. (2014), AP instance for 40 nodes

Instance	q	B&C Obj. Value Rodríguez-Martín et al. (2014)	HH. Obj. Value	Gap (%)
AP40_3_0.75_1	|V|	159131.34	159307.66	0.11
AP40_3_0.75_100	|V|	188910.27	190118.62	0.64
AP40_3_0.75_500	|V|	306243.01	309118.03	0.93
AP40_3_0.75_1000	|V|	445218.18	449552.36	0.96
AP40_4_0.75_1	|V|	144269.55	145531.86	0.87
AP40_4_0.75_100	|V|	174036.22	174210.98	0.10
AP40_4_0.75_500	|V|	291653.08	294126.26	0.84
AP40_4_0.75_1000	|V|	430540.90	432062.32	0.35
AP40_5_0.75_1	|V|	134569.34	135248.65	0.50
AP40_5_0.75_100	|V|	164038.24	165552.92	0.91
AP40_5_0.75_500	|V|	277247.49	277838.75	0.21
AP40_5_0.75_1000	|V|	411710.46	413003.15	0.31
AP40_3_0.75_1	|V|/2	159131.34	160361.65	0.77
AP40_3_0.75_100	|V|/2	188910.27	190482.33	0.83
AP40_3_0.75_500	|V|/2	306243.01	307611.19	0.44
AP40_3_0.75_1000	|V|/2	445218.18	448957.91	0.83
AP40_4_0.75_1	|V|/2	144269.55	144898.52	0.43
AP40_4_0.75_100	|V|/2	174036.22	175315.75	0.73
AP40_4_0.75_500	|V|/2	291653.08	294172.56	0.86
AP40_4_0.75_1000	|V|/2	430540.90	434211.48	0.85
AP40_5_0.75_1	|V|/2	134569.34	134937.21	0.27
AP40_5_0.75_100	|V|/2	164038.24	165195.15	0.70
AP40_5_0.75_500	|V|/2	277247.49	278422.13	0.42
AP40_5_0.75_1000	|V|/2	411710.46	413651.82	0.47
AP40_3_0.75_1	|V|/p	161989.74	163463.58	0.90
AP40_3_0.75_100	|V|/p	191404.41	191537.87	0.07
AP40_3_0.75_500	|V|/p	309484.80	312184.78	0.86
AP40_3_0.75_1000	|V|/p	454614.67	455895.18	0.28
AP40_4_0.75_1	|V|/p	145732.10	145926.45	0.13
AP40_4_0.75_100	|V|/p	176241.71	177536.93	0.73
AP40_4_0.75_500	|V|/p	295787.67	296627.18	0.28
AP40_4_0.75_1000	|V|/p	443393.74	444343.50	0.21
AP40_5_0.75_1	|V|/p	139032.42	140286.01	0.89
AP40_5_0.75_100	|V|/p	168736.29	169422.29	0.40
AP40_5_0.75_500	|V|/p	286728.55	289342.59	0.90
AP40_5_0.75_1000	|V|/p	453254.27	455785.30	0.56

Table 3.15: Comparison of the quality between results of HH and B&C Obj. Value Rodríguez-Martín et al. (2014), AP instance for 50 nodes

Instance	q	B&C Obj. Value Rodríguez-Martín et al. (2014)	HH. Obj. Value	Gap (%)
AP50_3_0.75_1	|V|	158880.67	159311.06	0.27
AP50_3_0.75_100	|V|	189643.25	190608.90	0.51
AP50_3_0.75_500	|V|	313509.45	314390.99	0.28
AP50_3_0.75_1000	|V|	461294.25	464755.89	0.74
AP50_4_0.75_1	|V|	143692.01	143819.90	0.09
AP50_4_0.75_100	|V|	174356.01	174676.08	0.18
AP50_4_0.75_500	|V|	297364.38	297604.55	0.08
AP50_4_0.75_1000	|V|	444031.52	445899.83	0.42
AP50_5_0.75_1	|V|	132689.72	133359.01	0.50
AP50_5_0.75_100	|V|	163460.64	163743.78	0.17
AP50_5_0.75_500	|V|	281644.93	283820.89	0.77
AP50_5_0.75_1000	|V|	426543.08	430379.06	0.89
AP50_3_0.75_1	|V|/2	158880.67	160187.65	0.82
AP50_3_0.75_100	|V|/2	189643.25	190272.34	0.33
AP50_3_0.75_500	|V|/2	313509.45	315496.93	0.63
AP50_3_0.75_1000	|V|/2	468204.82	471234.36	0.64
AP50_4_0.75_1	|V|/2	143692.01	145047.48	0.93
AP50_4_0.75_100	|V|/2	174356.01	175116.27	0.43
AP50_4_0.75_500	|V|/2	297364.38	300110.39	0.92
AP50_4_0.75_1000	|V|/2	444031.52	446973.31	0.66
AP50_5_0.75_1	|V|/2	132688.63	133157.79	0.35
AP50_5_0.75_100	|V|/2	163460.64	164354.90	0.54
AP50_5_0.75_500	|V|/2	281644.93	282177.99	0.19
AP50_5_0.75_1000	|V|/2	426543.08	430587.17	0.94
AP50_3_0.75_1	|V|/p	162358.48	163567.25	0.74
AP50_3_0.75_100	|V|/p	193611.51	194612.66	0.51
AP50_3_0.75_500	|V|/p	318506.03	321222.49	0.85
AP50_3_0.75_1000	|V|/p	487540.97	490968.03	0.70
AP50_4_0.75_1	|V|/p	144210.66	145604.69	0.96
AP50_4_0.75_100	|V|/p	175349.65	177053.61	0.96
AP50_4_0.75_500	|V|/p	300103.26	302645.79	0.84
AP50_4_0.75_1000	|V|/p	462471.69	463936.61	0.32
AP50_5_0.75_1	|V|/p	140093.97	140132.69	0.03
AP50_5_0.75_100	|V|/p	173088.85	173680.76	0.34
AP50_5_0.75_500	|V|/p	311508.49	311727.23	0.07
AP50_5_0.75_1000	|V|/p	524502.41	528222.94	0.70

Chapter 4

Selection Methods

4.1 Introduction

In the context of combinatorial problem, hyperheuristic is defined as *heuristics to select heuristics* Cowling et al. (2001). While the objective of a heuristic is to offer a feasible (near-optimal) solution, selective heuristic aims at identifying a series of heuristics to be applied. The selection process can be totally random or based on some historical performances. Statistical performances should be related to the solution quality offered by each heuristic and the computational time taken by each low-level-heuristic. When these accumulated statistical performance intervene in the selection process, we talk about learning mechanism. Learning is defined as the act of acquiring new, or modifying and reinforcing existing knowledge and information. According to the Oxford dictionaries, it is the acquisition of knowledge or skills through study, experience, or being taught. Hence, The science of enabling

computer to learn is known as machine learning (ML), that is considered as the capacity of a computer to learn from historical experiences.

In the context of hyperheuristic, machine learning contributes in selecting an incumbent heuristic to be applied based on historical performances. ML appears in different aspects such as *reinforcement learning*, *choice function*, *metaheuristic* etc, which will be elaborated in the second section. The problem of selection heuristics during hyperheuristic process is closely related to other problems such as *Algorithm Selection Problem* Rice (1976), *MetaLearning* Smith-Miles (2008) and etc. Algorithm Selection Problem deals with the selecting the best algorithm among a pool to solve a given problem. According to Rice (1976), The broad methodology which binds problems and solution methods with performance and problem characteristics is the algorithm selection framework. Three important dimensions should be taken into consideration when addressing algorithm selection problems: (i) the problem space; (ii) the algorithm space, and (iii) the performance measure. It is an abstract model as shown in Figure 4.1, which limits its wide application and it did not provide a specific implementation methods. However, metalearning has been successfully applied in different problem domains Smith-Miles (2008). Metalearning is a subfield of Machine learning Hilario et al. (2009), which exploits metadata (metafeatures) resulted from previous experiments by constructing models that can be used for prediction. In that situation in which no additional knowledge of available algorithms or problem structure exist, the automation of the whole process will then be called the *hyperheuristic*.

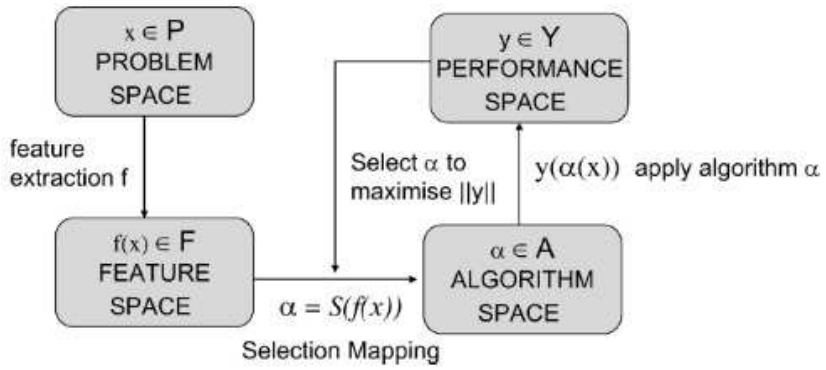

Figure 4.1: Algorithm selection model taken by Rice (1976).

4.2 The Proposed Selection Methods

Several selection methods have been proposed in the literature. Chakhlevitch and Cowling (2008a) categorizes such methods as in the sequel: i) random selection, ii) greedy selection, iii) peckish selection iv) metaheuristic-based hyperheuristic, and v) hyperheuristic with learning.

In this chapter, we will present and propose several heuristic selection methods, which are categorized in two main class: i) random selection (i.e. pure random selection, random descent, and etc.) (see in (Özcan and Kheiri, 2012), (Kendall and Mohamad, 2004)), and ii) intelligent method, which it is based on some accumulated historical performances (i.e. choice function (see in (Kendall et al., 2002a), (Drake et al., 2015)), metaheuristic (see in (Kendall and Mohamad, 2004), (Burke and Soubeiga, 2003)), etc.). In the other way, selection methods can be categorized as sequential selection(i.e. Random Permutation Descent (Cowling et al., 2001), Bayes (our proposition), Association Rules (our proposition), etc.) and parallel selection (i.e. peckish (see in (Cowling and Chakhlevitch, 2003), Cowling

and Chakhlevitch (2007)), greedy (see in (Cowling et al., 2001), (Cowling et al., 2002b)), (Özcan and Kheiri, 2012)) , etc.). Some of them perform by applying a selected heuristic until a worsening move is hit.

4.2.1 Random Selection Based Hyperheuristic

The term *Random* can be defined as "made, or occurring without definite aim, reason, or pattern". Selecting a random item from a predefined set, expects that all items have the same probability to be selected.

Randomness is the loss of pattern or predictability in activities. Therefore, *random heuristic selection method* is based on selecting one of existing heuristic with no data based input. For that, expected running time depends on the random choices only, not on any input distribution. Random selection algorithms are often simpler and faster than other algorithms. Another important feature of this method, is the inherit diversification in the process.

The disadvantages can be summarized as: i) lack of intensification strategy in the algorithm, ii) The solution quality can vary among more than one execution, and iii) heuristic selection probability is totaly independent of its performance.

In the following, we present three approaches of random selection regularly used in the literature: i) Pure random, ii) Random descent, and iii) Random permutation descent.

Pure Random

Pure random hyperheuristic is the most straightforward way to implement a of hyperheuristic. It is based on randomly selecting one of the existing low-level-heuristic from a set of predefined low-level-heuristics at each decision point Chakhlevitch and Cowling (2008a).

Pure random hyperheuristics have been applied in many problem domains such as scheduling problems (see Cowling et al. (2001), Cowling et al. (2002b), Cowling et al. (2002a)), space allocation problems (see Bai and Kendall (2005), Burke and Kendall (2005)), channel assignment in mobile communication problems(see Kendall and Mohamad (2004)) and time tabling problems(see Burke and Kendall (2005)).

Random Descent (Gradient)

Random descent or random gradient declare the heuristic selection strategy as choosing a low-level-heuristic randomly and employ it as long as the candidate solution in hand is improved Özcan et al. (2012). Hence, the improving low-level-heuristic is applied repeatedly until a worsening move is hit.

Random Descent method was applied in many domains (see Cowling et al. (2001), Soubeiga (2003) for examples of such a selection strategy).

Random Permutation Descent (Gradient)

Random Permutation Descent selection chooses randomly a set of low-level-heuristics, which form a cyclic list. It then applies the first heuristic and employ it until a worsening move is observed. It then applies the next ones in the order dictated by the cycle. Examples of such selection strategy can be found in Özcan and Kheiri (2012), Cowling et al. (2001) and Burke and Soubeiga (2003).

4.2.2 Greedy Based Hyperheuristic

Greedy selection applies all low-level-heuristics independently at each iteration, to the same solution candidate and chooses the one that generates the best solution quality. No diversification type is considered, while only improvement moves are accepted (see Cowling et al. (2002b), Cowling et al. (2001) and Özcan and Kheiri (2012) for examples of such strategies).

The limitation of this approach is mainly the limited capacity to explore the search space in order to avoid local optima (Chakhlevitch and Cowling, 2008a).

4.2.3 Peckish Based Hyperheuristic

The term *Peckish* follows the same concepts as a greedy selection, but a subset of low-level-heuristics is considered instead of all low-level-heuristics. (Cowling and Chakhlevitch, 2003) and (Cowling and Chakhlevitch, 2007) are among the examples of work considering this selection method.

In this chapter, we will present more than one method of *Peckish* selection based

on the way of choosing subset of low-level-heuristic to be applied in a parallel manner. There are as the following:

1. Peckish selection based on Tabu Search (TS), where TS, selects a predefined number of heuristics to be applied in parallel at each iteration.

2. Peckish selection based on Simulated Annealing (SA), where SA, selects predefined number of heuristics.

3. Peckish selection based on Association Rules (AR), where AR, selecst the subset of heuristics to be applied in parallel at each iteration.

4. Peckish selection based on Bayes Theorem (BT)

5. Peckish selection based on Q-Learning (QL)

4.2.4 Choice Function Based Hyperheuristic

Choice function heuristic selection methods, defined by Cowling et al. (2001), addresses learning as a model selection problem. The choice function selects a low-level-heuristic to be applied according to a weight resulted from a combination of three different measures. Therefore, selection of such low-level-heuristic at each iteration depends on its corresponding choice function. For that reason, score computations for the low-level-heuristics are repeated at each iteration.

The low-level heuristics are ranked according to several measures. The heuristic ranking technique is based on: I) the heuristic's individual performance (Equation (4.1)), II) improvement observed as a results of invoking a heuristic subsequent

to another one (Equation (4.2)), and III) the time elapsed since it was last called (Equation (4.3)). $I_n(y)$ and $T_n(y)(I_n(x,y)$ and $T_n(x,y))$ indicate the change of evaluation function and the amount of execution time, when the n^{th} last time the heuristic y was chosen and applied immediately after heuristic x.

Intensification strategies were provided by the first two components, while the diversification was assured by the third on. α, β, and δ, represents the weights of the three components and their relative influence on the choice function. Both α and $\beta \in [0,1]$ reflect the greater influence attached to recent performance.
The approach mentioned above is prevented from reaching its full potential by several obstacles. The first obstacle being that it requires a warm-up period during which the values of the choice functions will be initialized by randomly selected heuristics. The second one, is that achieving the best results requires the weights α, β, and δ of individual components in the choice function to be manually tuned.

$$f_1(h_j) = \sum_n \alpha^{n-1}(\frac{I_n(h_j)}{T_n(h_j)}) \tag{4.1}$$

$$f_2(h_k, h_j) = \sum_n \beta^{n-1}(\frac{I_n(h_k, h_j)}{T_n(h_k, h_j)}) \tag{4.2}$$

$$f_3(h_j) = elapsedTime(h_j) \tag{4.3}$$

$$f(h_j) = \alpha f_1(h_j) + \beta f_2(h_k, h_j) + \delta f_3(h_j) \tag{4.4}$$

Cowling et al. (2001), Cowling et al. (2002b), Kendall et al. (2002b) and Soubeiga (2003) are examples of work considering this heuristic selection method.

4.2.5 Reinforcement Learning

In reinforcement learning we are dealing with an *agent* that learns its behavior through some interaction with a so called *dynamic environment*. Such interaction is usually realized in trial-and-error efforts fashion. At each point in time, t, an agent is in a given state $s_t \in S$ and can choose an action a_t, eventually from among several, to make a transition to another state $s_{t+1} \in S$. In this work, we deal with the sets S and A of bounded cardinality, i.e. $card(|A|)+card(|S|) < card(\mathbb{R}) = \aleph$. More precisely, where we work with a finite number of transition that can be made.

When an action is taken, a numerical reward, r_{t+1} is attributed. Such reward is a feedback from the environment that depends on the quality of action taken by the agent. An agent that wants to maximize the total reward attributed to it, has to be able to exploit the experiences it has obtained so far and also be able to explore other parts of the environment and identify better actions Sutton and Barto (1998). In general, there are two types of reinforcement learning:

1. *Online Learning*: heuristic rewards are updated online (i.e. dynamically) throughout the search according to the performance of every heuristic.

2. *Off-line Learning*: heuristics rewards are retrieved from a database, which stores the historical performance of every heuristic in the past.

Naive Bayes Classifier based Hyperheuristic

In the context of probability science, inverse probability is an obsolete term for the probability distribution of an unobserved variable. Hence, Thomas Bayes (1701 1761) proposed a solution to a problem of inverse probability. The proposed solution was presented in "An Essay towards solving a Problem in the Doctrine of Chances" which was read to the Royal Society in 1763 after Bayes' death.

Today, the problem of determining an unobserved variable (by whatever method) is called inferential statistics, the method of inverse probability (assigning a probability distribution to an unobserved variable) is called Bayesian probability, the "distribution" of an unobserved variable given data is rather the likelihood function (which is not a probability distribution), and the distribution of an unobserved variable, given both data and a prior distribution, is the posterior distribution. The development of the field and terminology from "inverse probability" to "Bayesian probability" is described by Fienberg (2006).

Bayesian probability is one interpretation of the concept of probability. In contrast to interpreting probability as frequency or propensity of some phenomenon, Bayesian probability is a quantity that we assign to represent a state of knowledge, Jaynes (1986) or a state of belief.De Finetti (1974) In the Bayesian view, a probability is assigned to a hypothesis, whereas under frequentist inference, a hypothesis is typically tested without being assigned a probability.

Naive Bayes classifier assumes that all the features are conditionally independent of each other. This therefore permits us to use the Bayesian rule for probability. Usually this independence assumption works well for most cases, if even in actuality they are not really independent. Bayesian rules is stated mathematically as the following equation De Finetti (1974):

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}, \tag{4.5}$$

A and B are events, products, actions or etc. $P(A)$ and $P(B)$ are the probabilities of A and B without regard to each other. $P(A|B)$, a conditional probability, is the probability of observing event A given that B is true. $P(B|A)$, is the probability of observing event B given that A is true.

In our proposed method, after each heuristic being applied, the frequency of worsening moves and the frequency of improvement move was accumulated. Therefore, in our hand we have at each iteration, the number of times each heuristic has improved/worsened the solution. Hence, we can calculate the probability of a heuristic h_i given that the solution is improved.

The method is based on dividing the time horizon into three portion ($V1$, $V2$ and $V3$). In $V1$, which represents the time when the process starts till the time limit (T_l) divided by 5, heuristics are selected totaly random. In $V2$, which is bounded between $V1$ and $3T_l$ divided by 5, a heuristic h_i will be selected in the case of having the probability of being applied given that the solution was improved ($P(h_i|class = Improvement)$) is grater than 0.4. In addition, a

chance of 0.2 is acceptable by the search space in the case where the heuristic is selected randomly. In the last period, $V3$, the selected heuristic h_i must have $P(h_i|class = Improvement)$ greater than 0.6, or it may be selected in a probability of 0.1.

Bayesian approach is usually used in the domain of data mining, such as in Heckerman (1997), Kirkos et al. (2007), and Kantarcıoglu et al. (2003). Also, many researches applied bayesian approach in the context of optimization such as Inza et al. (2000), Mockus (2012), and Pelikan (2005).

4.2.6 Metaheuristics Based Hyperheuristic

Metaheuristic, as mentioned in the chapter 1, can be defined as an iterative generation of a process, which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space Osman and Laporte (1996). Here, the role of metaheuristic is not to search in a solution space, but to search in the space of heuristics in order to guide HH to choose heuristics during HH process. We implement different metaheuristic methods: Simulated Annealing, Ant Colony, Variable Neighborhood Search and Genetic Algorithm.

Simulated Annealing Based Hyperheuristic

Simulated Annealing (SA) as an idea is inspired from a paper published by Metropolis et al. (1953), is motivated by an analogy to annealing in solids. It describes an algorithm that simulates the process called 'annealing' which is the cooling

of materials in a heat bath. Cooling a solid after heating it past its melting point makes its structural properties dependent on the cooling rate. It is also possible to notice the formation of crystals in the case where the cooling process was slow enough. If cooling the materials was achieved rapidly then there is a high chance that the crystals will contain imperfections. The materials used in Metropolis's algorithm were simulated as a system of particles. The simulation lowered the system's temperature gradually until it converged to a steady, frozen state. In 1982, optimization problems were subjected to SA for the first time in Kirkpatrick (1984). Simulated annealing was used to search for feasible solutions that could converge into an optimal solution.

The law of thermodynamics express that at temperature, t, the likelihood of an expansion in vitality of greatness, δE, is given by

$$P(\delta E) = \exp(-\delta E/kT) \tag{4.6}$$

Where k is a consistent known as Boltzmann's steady. The reenactment in the Metropolis calculation computes the new vitality of the framework. On the off chance that the vitality has diminished then the framework moves to this state. In the event that the vitality has expanded then the new state is acknowledged utilizing the likelihood returned by the above recipe. A specific number of emphases are done at every temperature and afterward the temperature is diminished. This is rehashed until the framework solidifies into an unfaltering state. This mathematical

statement is specifically utilized as a part of reenacted toughening, in spite of the fact that it is common to drop the Boltzmann consistent as this was just acquainted into the comparison with adapt to various materials. Along these lines, the likelihood of tolerating a more terrible state is given by the mathematical statement

$$P = \exp(-c/t) \geq r \tag{4.7}$$

Where:

c = the adjustment in the assessment capacity.

t = the present temperature.

r = an irregular number somewhere around 0 and 1.

The likelihood of tolerating a more terrible move is a component of both the temperature of the framework and of the adjustment in the cost capacity. It can be valued that as the temperature of the framework diminishes the likelihood of tolerating a more terrible move is diminished. This is the same as slowly moving to a solidified state in physical annealing. Additionally take note of, that if the temperature is zero then just better moves will be acknowledged which successfully makes recreated strengthening act like slope climbing.

SA was widely applied in many different combinatorial problem, such as scheduling problem (see Ma et al. (2016), Bouleimen and Lecocq (2003) and Schlünz and Van Vuuren (2013)), hub location problem (see Sedehzadeh et al. (2014), Parvaresh et al. (2013) and Rabbani and Kazemi (2015)), routing problem (see Grosse et al.

(2014),BañOs et al. (2013) and Vincent and Lin (2014)), workover rig problem (see Ribeiro et al. (2011) and Paiva et al. (2000)) and etc.

SA was used in hyperheuristic approaches in a two manners: I) in selection heuristics and II) in accepting a candidate solution. SA as an accepting criteria was explained in Chapter 1.

SA heuristic selective hyperheuristic was applied as a tool to select a heuristic of a reward r ,$0 < r < 1$, having $\exp(\frac{-r}{t}) > random(0, 1)$. Annealing schedule is considered as linear, where:

$$T(t) = T_0 - \eta t \tag{4.8}$$

Dowsland et al. (2007), Bai et al. (2012), Anagnostopoulos and Koulinas (2010) are examples of SA selective hyperheuristic.

Ant Colony Based Hyperheuristic

Ant colony optimization (ACO) is among the well-known population-based metaheuristics that has been extensively proposed to solve difficult optimization problems. In the natural world, ants move randomly, and then they lay down pheromone trails by the way of finding food and returning to their colony. Other ants stop wandering randomly upon discovering such a path by following the trail. With time, evaporation of pheromone trail will take place and its attractive strength will be minimized. The pheromone evaporates more as the time taken by the ants to stroll the path back and forth increases. For clarification, the pheromone density

becomes higher on shorter paths, which is the more frequent marched path, than longer ones. Pheromone evaporation is the key process of searching for optimal solution. However, if evaporation process does not exist, all ants will follow the first path chosen by first ants. Therefore, the exploration of the solution space would be constrained. In such cases, when one ant finds the shortest path from colony to a food source, all ants are aiming to follow this single path with their positive leading feedback. The goal of the ant colony algorithm is to avoid such a behaviour with a kind of 'simulated ants' walking around the graph representing the problem to solve. Successful applications of ACO has been reported in hub location routing problem (Randall (2008), Ting and Chen (2013)), vehicle routing problem (Chávez et al. (2016), Gajpal and Abad (2009)), timetable problem (Dowsland and Thompson (2005), Ayob and Jaradat (2009)) etc.

This technique has been used as a heuristic selection method in hyperheuristic approaches. The ant based hyperheuristic, proposed by Kiraz et al. (2013), is proven to be better than the choice function hyperheuristic on some problem domains. Another ant-based hyperheuristic proposed by Ferreira et al. (2015). A hybrid flow shop scheduling problem is solved by an ant colony optimization-based hyperheuristic combined with a genetic algorithm, which forms a hyperheuristic that generate and select heuristics.

Our proposed method is the following:

1. An arc is formed between every two heuristic, which allows the transition between any heuristic to another.

2. A pheromone τ_{ij} and a visibility function η_{ij} were defined for each vertex <i,j>, and they are updated after each heuristic application using (4.10) and (4.9). The visibility function in our proposition is inspired from the choice function hyperheuristic.

$$\eta_{ij} = \beta \frac{I(h_i, h_j)}{T(h_i, h_j} \tag{4.9}$$

$$\tau_{ij} = \begin{cases} \rho \times \eta_{ij} + \frac{C^{best}}{C^k} & \text{if ant } k \text{ chooses heuristic } j \text{ after heuristic } i \\ \rho \times \eta_{ij} & \text{Otherwise} \end{cases} \tag{4.10}$$

It should be to note that, the ρ coefficient indicates the evaporation rate.

3. Since the hyperheuristic starts with no knowledge about the existing low-level-heuristics, it must be initially unbiased and impartial while continuously adaptive.

4. The probability of selecting a heuristic j after a applying heuristic i is defined in (4.11).

$$P_{ij} = \frac{\eta_{ij} \times \tau_{ij}}{\sum_i \sum_j \eta_{ij} \times \tau_{ij}} \tag{4.11}$$

Variable Neighborhood Search based Hyperheuristic

Variable neighborhood search (VNS), proposed by Mladenović and Hansen (1997b), is another kind of metaheuristics for COPs. It explores neighborhoods, which ap-

plied on solution space, and moves from there randomly or intelligently. VNS includes two main phases: I) shaking phase and II) local search phase. The first one aims to diversify the search space in order to escape from local optima, while the objective of the second phase is to intensify the search around the current solution in order to improve it. Figure 4.2 shows how each combination of these two phases contributes in avoiding a local optimum and tries to explore and exploit the search space in order to achieve a near optimal solution. Each phase contains more than one heuristic, which necessitates a definition of the heuristic selection method.

VNS appears an affective tool when it is combined with hyperheuristic methods. Hsiao et al. (2012) propose a VSN based hyperheuristic method for solving more than one problem such as job shop scheduling, bin packing, and etc Hansen and Mladenović (2014). Remde et al. (2007) proposed a VNS hyperheuristic in order to solve workforce scheduling problem. In this work, hyperheuristic appears a powerful tools compared with the solution quality resulted by a genetic algorithm. The proposed VNS hyperheuristic, shown in Figure 4.3, aims to constructs a feasible solution at the beginning, then construct the shaking and finally the local search phases. In the shaking phase, the tabu search method, which is proposed in chapter 2, selects a low level heuristic from the set of perturbation heuristic. In the second phase, a random cycle of the existing predefined improvement heuristics is formed. It applies improvement heuristic while it is still improving the solution and switches to the next one in the cycle in case it stops improving the solution. It keeps on repeating this process until reaching n consecutive non-improving solution.

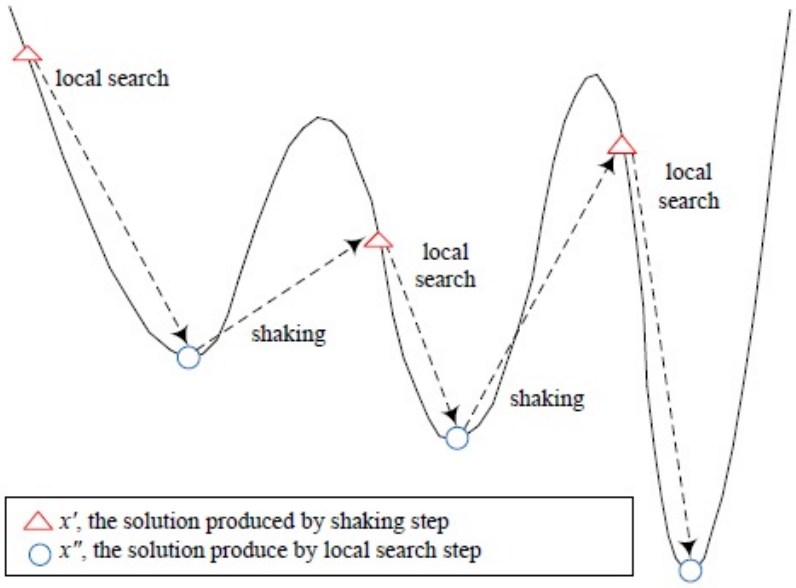

Figure 4.2: A general schema of the VNS algorithm.

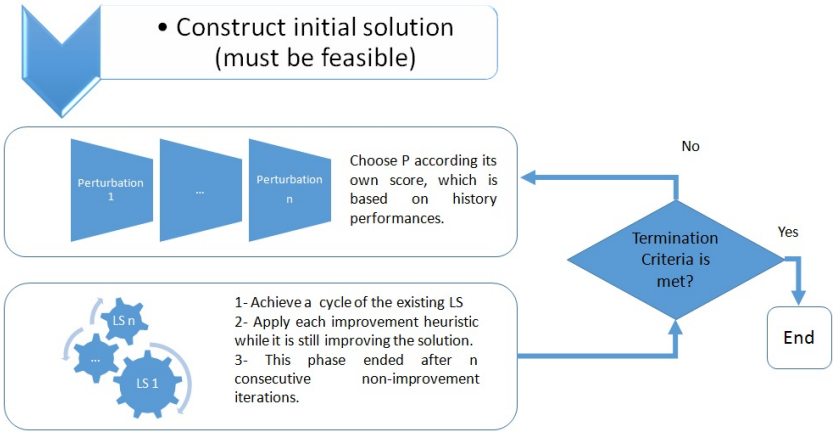

Figure 4.3: The process of the proposed VNS Hyperheuristic.

4.3 Numerical Results

In order to examine performance of different selection methods, every method is in turn examined on the same problem in chapter 3, CSApHLRP-1, and the same testbed.

Results of HH within different heuristic selection methods are reported in this section (including Tabu Search method proposed in chapter 2, Association Rules and Q-learning proposed in chapter 3) in 4.4 (and further elaborated in Figure 4.5, Figure 4.6 and Figure 4.7) while the acceptance criteria employe here is SA and all methods have the same input. Low gaps between the different methods, even the pure random selection method, prove the effectiveness of the hyperheuristic. Overall, pure random and greedy selection methods were the worst one, while the Association Rules selection method was the better. Note that, for small instances, the solution quality differences between most selection methods is minimal. Random Descent and Random Permutation Descent are better than pure random method. Peckish based on Bayes theorem was proven to be exceptionally effective compared with the most other methods. Reinforcement learning and metaheuristic method have a good solutions compared with the other, specially Association Rules, Q-learning and Tabu Search.

In general, SA acceptance criteria with the different selection methods is the better, while minimal differences exists when using the other acceptance criteria. It should be note that:

1. Association Rules with SA is indeed better than with Great Deluge by an

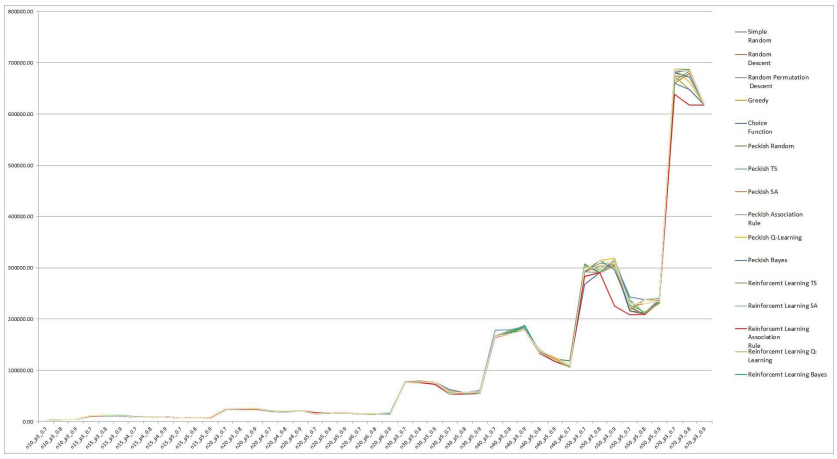

Figure 4.4: HH results within different heuristic selection methods

average of 0.7%.

2. In all selection methods, almost any other acceptance criteria is better than random acceptance in 0.62% except the choice function in 0.38%.

3. Choice function with SA is better than OI in 0.84%.

4. Choice function with SA is better than IE in 0.24%.

5. Q-learning with Great Deluge is better than with SA in 0.102%.

Furthermore, if we expand the execution time no additional gain was observed.

4.4 Conclusion

In this chapter, we presented and proposed different heuristic selection methods in order to integrate them in HH framework. We propose different learning and

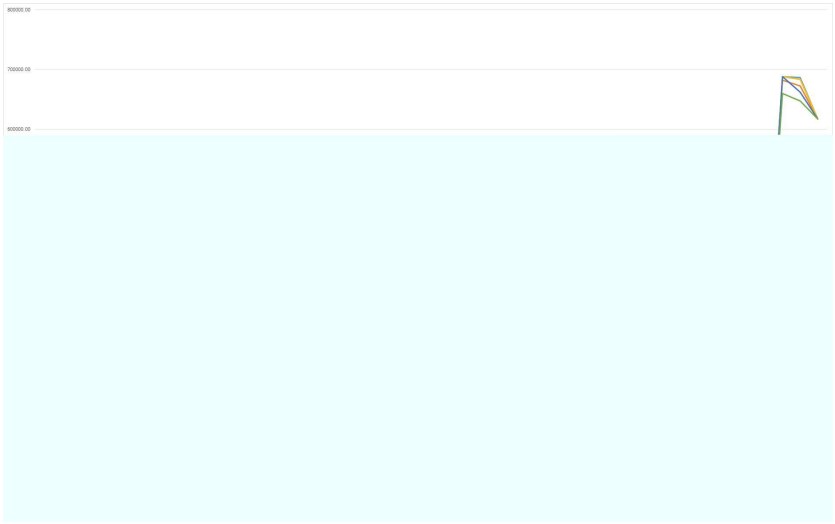

Figure 4.5: HH results within different peckish selection methods

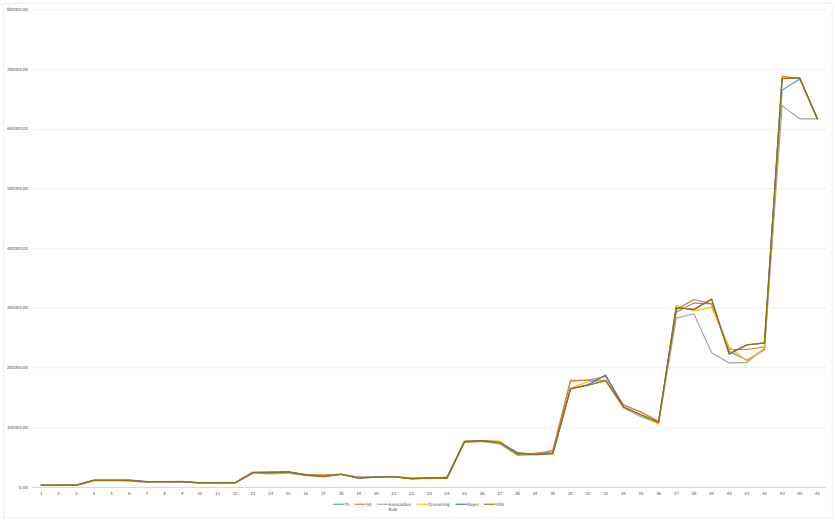

Figure 4.6: HH results within different RL selection methods

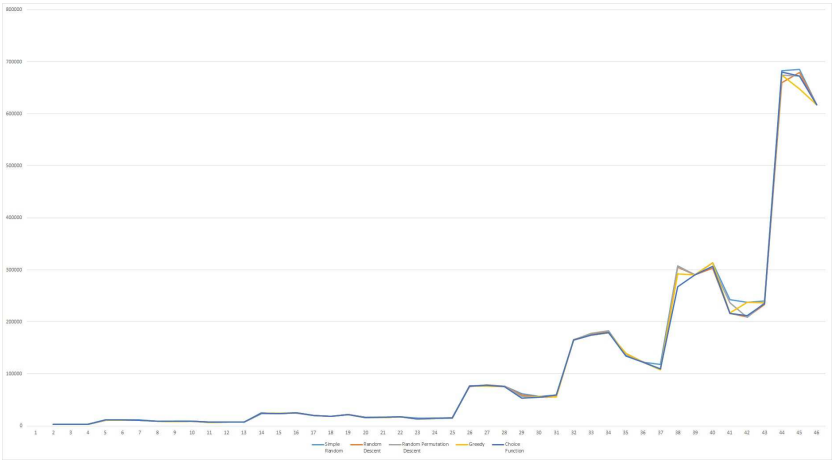

Figure 4.7: HH results within simple random, random descent, random permutation descent, greedy and choice function selection methods

metaheuristic methods as heuristic selection methods and we compared their performances when applyed on HLRP proposed in chapter 3. Here, SA acceptance criteria appears in average the best acceptance method among others such as all moves method, naive method, great deluge, etc. The performance of the overall performance of methods; even the random and greedy selection which is the last effective one; appears very promising, which proves the effectiveness of the other strategies of HH as well as robustness of LLHs. AR methods, which is well stablished in business data mining application, has proven to be the most effective learning method to select subsequent heuristics during the process of HH.

Chapter 5

General Conclusion

In this thesis, we have studies the Hyperheuristic (Burke et al., 2008b, 2013), which is a search method or learning mechanism to select or generate heuristic in order to solve a specific combinatorial problem. Hyperheuristic is considered as a high-level approach that, given a problem and a number of low-level heuristics, can select/generate and apply a suitable low-level heuristic at each iteration. Topcuoglu et al. (2014) defines it as methods for automation of the process of selecting and generating multiple low-level heuristics. Clearly, the difference between metaheuristics and hyperheuristics lies in the fact that metaheuristics search directly on the solution space while hype-heuristics work on the heuristic space. This work proposes a selective hyperheuristic framework, which is comprised of a set of low-level heuristics and a heuristic selection mechanisms.

In our study, we are dealing with problems in logistics. For general business concept, logistics is the administration of the flow of any matters between the point

of origin and that of consumption orderly to convene the customers or corporations needs. The managed resources in logistics may involve some physical matters like food, materials, animals, equipments and liquids, and some abstract matters like time and information. The logistics of physical matters can include the gathering of information flow, material handling, production, packaging, inventing, transportation, warehousing, and mainly security. Logistics management, which represented as an important part of supply chain management, focuses on planing, implementing, and controlling the efficient, effective forward, and reverse flow and storage of goods, services, and related information between the origin point to reach the destination.

In the first thesis phase, we are invited to be a member with *Brazilian* collaboration group, that is interested in a real logistic problem: Workover Rig Problem. The Workover Rig Routing Problem (WRRP) arises on onshore oil fields that can have a large number of wells spread over a wide region and relatively few number of the mobiles maintenance machines called workover rigs. To serve wells that demand every type of maintenance, WRRP aims to determine over the next finite horizon of time H, a feasible routes of workover rigs that minimize the sum of oil production lossRibeiro et al. (2013). WRRP states on finding the best routes of rigs in order to serve all wells, in an accepted time and the oil production loss is minimized Ribeiro et al. (2012c). For this problem, we propose a mathematical formulation model, several valid inequalities, and finally a hyperheuristic method based on a proposed tabu search metaheuristic method as a learning method. Our proposed learning has been proven its effectiveness when compared with an exact method

(Monemi et al., 2015), as long as the optimal solution (or the optimal interval) is known, the solutions reported by the hyper-heuristic produce very near optimal solutions. Our learning method outperforms a generic intelligent hyper-heuristic (GIHH) proposed earlier in the literature both in terms of time and quality. In addition, the proposed hyper-heuristic output has served as the initial columns for the proposed branch-and-price in Monemi et al. (2015).

On the other hand, another logistic problem has been studied in this thesis: Hub Location Routing Problem. Hub Location Routing Problem dealing with selecting from several nodes (e.g. ports), which will be considered as hub and which will not be (spoke), then design the full network according a given rules. Many transportation operators and logistics service providers operate on hub-and-spoke structures. In this way, the long haul (in national or international scale) transport of their cargo is done by using larger transporters (with higher volume concentration) circulating between major hubs. In a short-distance level, transportation service is realized by means of smaller transporters. In this thesis, we was interested on pHub location routing problem, which fix the number of hubs as p. A hyperheuristic approach has been proposed as an effective solution for a recently introduced two variant of Hub Location Routing Problems proposed by Gelareh et al. (2015) and Rodríguez-Martín et al. (2014), with two different learning methods: Association Rules and Q-learning. Furthermore, two solution acceptance criteria, for the two problems, was in the HH high level strategies: Simulated Annealing and Great Deluge. Five Lagrangian Relaxation have been proposed for the first problem, and we use the results of the best one in the process of HH. HH with Lagrangian Relax-

ation results has been proven in the numerical results as better than HH without LR results, even in the initial fitness. A part of HH results has been injected to the solver, in order to find an optimal solution for some given allocation information, as a *Matheuristic* method. As well as the HH has been proven its effectiveness by the too minimal gap with the proposed Matheuristic.

Finally, in order to examine how different heuristic selection mechanism can impact the performance of a hyperheuristic framework and quality of solution for a given problem, several exiting methods were studied and some new ones are proposed. Through an extensive computational experiment, we could identify some of the methods that outperform the others for the problem at hand.

Özcan et al. (2010) categorizes such methods as the following: i) Purely Random selection, ii) Random Descent selection, iii) Random Permutation Descent selection, iv) Greedy selection, v) Peckish selection, vi) Metaheuristic-based and vii) Reinforcement Learning. Here, all these categories were implemented. Hence, we propose different reinforcement learning and metaheuristic methods such as Naive Bayes, ACO, SA, VNS, etc. that guide in selection of heuristics. these methods was examined with several solution acceptance criteria such as SA, Great Deluge, IO, etc. Association Rules method, an intelligent tools used in business data mining application, was the most effective method in selection of heuristics during HH process when combined with SA solution acceptance criteria.

Our future directions consider solving these two problems with taking into consideration more real-world aspects such as task time window, constraints related to the environment such that related to green logistics, etc. In addition, we are very

interested to faced problems of Emergency Logistics (or Humanitarian Logistics), which deals with denote specific time-critical modes of transport used to move service or goods or objects rapidly in the event of an emergency. This motivation has been introduced in a our contribution Danach et al. (2015a). Furthermore, we will be more interested to develop more intelligent methods, that are proposed in artificial intelligence science such as Neural Network (NN), Decision Tree (DT), etc. in order to perform heuristic selection phase in the HH framework. Finally, developing a hybrid method that combine metaheuristic and hyperheuristic, in order to relate the heuristic to be applied to the solution state in addition to its performance, can be intervenes in our future work.

Bibliography

Inmaculada Rodríguez-Martín, Juan-José Salazar-González, and Hande Yaman. A branch-and-cut algorithm for the hub location and routing problem. *Computers & Operations Research*, 50:161–174, 2014.

Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyperheuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010.

John R. Rice. The algorithm selection problem. *Advances in Computers*, 15: 65–118, 1976.

Alexander Schrijver. Combinatorial optimization: Polyhedra and efficiency. *Discrete Applied Mathematics*, 146:120–122, 2005.

Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The*

traveling salesman problem: a computational study. Princeton university press, 2011.

Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. Siam, 2014.

Zvi Drezner and Horst W Hamacher. *Facility location: applications and theory*. Springer Science & Business Media, 2001.

Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

Ibrahim H Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations research*, 63(5):511–623, 1996.

El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

Emile Aarts and Jan Korst. Simulated annealing and boltzmann machines. 1988.

Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

Lawrence Davis. Handbook of genetic algorithms. 1991.

KG Dyall, IP Grant, CT Johnson, FA Parpia, and EP Plummer. Grasp: A general-purpose relativistic atomic structure program. *computer physics communications*, 55(3):425–456, 1989.

Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997a.

Ravindra K Ahuja, James B Orlin, and Dushyant Sharma. Very large-scale neighborhood search. *International Transactions in Operational Research*, 7(4-5): 301–317, 2000.

Fred Glover. A template for scatter search and path relinking. *Lecture notes in computer science*, 1363:13–54, 1998.

Russ C Eberhart, James Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

Enrico Bartolini and Aristide Mingozzi. Algorithms for the non-bifurcated network design problem. *Journal of Heuristics*, 15(3):259–281, 2009.

Jörg Denzinger, Marc Fuchs, and Matthias Fuchs. *High performance ATP systems by combining several AI methods*. Citeseer, 1996.

Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Practice and Theory of Automated Timetabling III*, pages 176–190. Springer, 2001.

Wallace B Crowston, Fred Glover, Jack D Trawick, et al. Probabilistic and parametric learning combinations of local job shop scheduling rules. Technical report, DTIC Document, 1963.

Hsiao-Lan Fang, Peter Ross, and David Corne. *A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems*. University of Edinburgh, Department of Artificial Intelligence, 1993.

E. Özcan, M. Misir, G. Ochoa, and E. K. Burke. A reinforcement learning-great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing (IJAMC)*, 1(1):39–59, 2010.

Haluk Rahmi Topcuoglu, Abdulvahid Ucar, and Lokman Altin. A hyper-heuristic based framework for dynamic optimization problems. *Applied Soft Computing*, 19:236–251, 2014.

Gabriela Ochoa, Rong Qu, and Edmund K Burke. Analyzing the landscape of a graph based hyper-heuristic for timetabling problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 341–348. ACM, 2009.

Peter Ross. *Search Methodolgies*, chapter Hyper-heuristics, pages 529–556. 2005.

Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and Rong Qu. A survey of hyper-heuristics. *Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham*, 2009a.

Mohamed Bader-El-Den and Riccardo Poli. Generating sat local-search heuristics using a gp hyper-heuristic framework. In *Artificial evolution*, pages 37–49. Springer, 2008.

Eric Soubeiga. *Development and application of hyperheuristics to personnel scheduling*. PhD thesis, University of Nottingham, 2003.

Konstantin Chakhlevitch and Peter Cowling. Hyperheuristics: recent developments. In *Adaptive and multilevel metaheuristics*, pages 3–29. Springer, 2008a.

Peter Cowling, Graham Kendall, and Eric Soubeiga. Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In *Applications of Evolutionary Computing*, pages 1–10. Springer, 2002a.

Alexander Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer decision-making*, pages 523–544. Springer, 2004.

Masri Ayob and Graham Kendall. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the international conference on intelligent technologies, InTech*, volume 3, pages 132–141. Citeseer, 2003a.

Ruibin Bai and Graham Kendall. An investigation of automated planograms using a simulated annealing based hyper-heuristic. In *Metaheuristics: Progress as Real Problem Solvers*, pages 87–108. Springer, 2005.

Gunter Dueck. New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92, 1993.

Edmund K Burke and Yuri Bykov. A late acceptance strategy in hill-climbing for exam timetabling problems. In *PATAT 2008 Conference, Montreal, Canada*, 2008.

Masri Ayob and Graham Kendall. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the international conference on intelligent technologies, InTech*, volume 3, pages 132–141, 2003b.

Aris Anagnostopoulos, Laurent Michel, Pascal Van Hentenryck, and Yannis Vergados. A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, 9(2):177–193, 2006.

Ruibin Bai, Jacek Blazewicz, Edmund K Burke, Graham Kendall, and Barry McCollum. A simulated annealing hyper-heuristic methodology for flexible decision support. *School of CSiT, University of Nottingham, UK, Tech. Rep*, 2007.

Burak Bilgin, Ender Özcan, and Emin Erkan Korkmaz. An experimental study on hyper-heuristics and exam timetabling. In *Practice and Theory of Automated Timetabling VI*, pages 394–412. Springer, 2007.

Graham Kendall and Mazlan Mohamad. Channel assignment in cellular communication using a great deluge hyperheuristic. In *Networks, 2004.(ICON 2004).*

Proceedings. 12th IEEE International Conference on, volume 2, pages 769–773. IEEE, 2004.

Paul Mcmullan. An extended implementation of the great deluge algorithm for course timetabling. In *Computational Science–ICCS 2007*, pages 538–545. Springer, 2007.

Ei Shwe Sin and Nang Saing Moon Kham. Hyper heuristic based on great deluge and its variants for exam timetabling problem. *arXiv preprint arXiv:1202.1891*, 2012.

Ender Özcan, Yuri Bykov, Murat Birben, and Edmund K Burke. Examination timetabling using late acceptance hyper-heuristics. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 997–1004. IEEE, 2009.

Edmund K Burke, Graham Kendall, Mustafa Mısır, and Ender Özcan. A study of simulated annealing hyper-heuristics. In *Proceedings of the international conference on the practice and theory of automated timetabling (PATAT 2008)*, 2008a.

R Raghavjee and N Pillay. A genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem. *ORiON*, 31(1):39–60, 2015.

Alena Shmygelska and Holger H Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC bioinformatics*, 6(1):1, 2005.

SM GiriRajkumar, K Ramkumar, and Sanjay Sarma O V. Real time application of ants colony optimization. *International Journal of Computer Applications*, 3(8), 2010.

Leena N Ahmed, Ender Özcan, and Ahmed Kheiri. Solving high school time-tabling problems worldwide using selection hyperheuristics. *Expert Systems with Applications*, 42(13):5463–5471, 2015.

Yu-Jun Zheng, Min-Xia Zhang, Hai-Feng Ling, and Sheng-Yong Chen. Emergency railway transportation planning using a hyperheuristic approach. *Intelligent Transportation Systems, IEEE Transactions on*, 16(1):321–329, 2015.

Rajni Aron, Inderveer Chana, and Ajith Abraham. A hyperheuristic approach for resource provisioning-based scheduling in grid environment. *The Journal of Supercomputing*, 71(4):1427–1450, 2015.

Kassem Danach, Jomana Al-Haj Hassan, Wissam Khalil, and Shahin Gelareh. Routing heterogeneous mobile hospital with different patients priorities: Hyperheuristic approach. In *Digital Information and Communication Technology and its Applications (DICTAP), 2015 Fifth International Conference on*, pages 155–158. IEEE, 2015a.

Kassem Danach, Wissam Khalil, and Shahin Gelareh. Multiple strings planing problem in maritime service network: Hyperheuristic approach. In *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 Third International Conference on*, pages 85–88. IEEE, 2015b.

Rahimeh Neamatian Monemi, Kassem Danach, Wissam Khalil, Shahin Gelareh, Francisco C Lima, and Dario José Aloise. Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. *Expert Systems with Applications*, 42(9):4493–4505, 2015.

Richard J Marshall, Mark Johnston, and Mengjie Zhang. Developing a hyperheuristic using grammatical evolution and the capacitated vehicle routing problem. In *Simulated Evolution and Learning*, pages 668–679. Springer, 2014.

Pablo Garrido and María Cristina Riff. Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyperheuristic. *Journal of Heuristics*, 16(6):795–834, 2010.

Pablo Garrido and Carlos Castro. A flexible and adaptive hyperheuristic approach for (dynamic) capacitated vehicle routing problems. *Fundamenta Informaticae*, 119(1):29–60, 2012.

Kevin Sim, Emma Hart, and Ben Paechter. A lifelong learning hyperheuristic method for bin packing. *Evolutionary computation*, 23(1):37–67, 2015.

Muhammed Beyaz, Tansel Dokeroglu, and Ahmet Cosar. Robust hyperheuristic algorithms for the offline oriented/non-oriented 2d bin packing problems. *Applied Soft Computing*, 36:236–245, 2015.

Peter Ross, Sonia Schulenburg, Javier G Marín-Blázquez, and Emma Hart. Hyperheuristics: learning to combine simple heuristics in bin-packing problems. In *GECCO*, pages 942–948, 2002.

Ali Keles, Yayimli, et al. Ant based hyper heuristic for physical impairment aware routing and wavelength assignment. In *Sarnoff Symposium, 2010 IEEE*, pages 1–5. IEEE, 2010.

Carlos Segura, Gara Miranda, and Coromoto León. Parallel hyperheuristics for the frequency assignment problem. *Memetic Computing*, 3(1):33–49, 2011.

Hugo Terashima-Marín, José Carlos Ortiz-Bayliss, Peter Ross, and Manuel Valenzuela-Rendón. Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 571–578. ACM, 2008.

José Carlos Ortiz-Bayliss, Ender Özcan, Andrew J Parkes, and Hugo Terashima-Marín. Mapping the performance of heuristics for constraint satisfaction. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

Konstantin Chakhlevitch and Peter Cowling. Hyperheuristics: recent developments. In *Adaptive and multilevel metaheuristics*, pages 3–29. Springer, 2008b.

Graham Kendall, Peter Cowling, and Eric Soubeiga. Choice function and random hyperheuristics. In *Proceedings of the fourth Asia-Pacific conference on simulated evolution and learning, SEAL*, pages 667–671, 2002a.

Edmund K Burke, Graham Kendall, and Eric Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.

Edmund K Burke, Bart L MacCarthy, Sanja Petrovic, and Rong Qu. Knowledge discovery in a hyper-heuristic for course timetabling using case-based reasoning. In *Practice and Theory of Automated Timetabling IV*, pages 276–287. Springer, 2002.

Edmund K Burke, Sanja Petrovic, and Rong Qu. Case-based heuristic selection for timetabling problems. *Journal of Scheduling*, 9(2):115–132, 2006.

Sanja Petrovic and Rong Qu. Case-based reasoning as a heuristic selector in a hyper-heuristic for course timetabling problems. 2002.

Hugo Terashima-Marín, EJ Flores-Alvarez, and Peter Ross. Hyper-heuristics and classifier systems for solving 2d-regular cutting stock problems. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 637–643. ACM, 2005a.

Hugo Terashima-Marín, Armando Morán-Saavedra, and Peter Ross. Forming hyper-heuristics with gas when solving 2d-regular cutting stock problems. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1104–1110. IEEE, 2005b.

Stewart W Wilson, SW Wilson, Generalization Xcs, et al. Generalization in the xcs classifier system. 1998.

Sonia Schulenburg, Peter Ross, Javier G Marín-Blázquez, and Emma Hart. A hyper-heuristic approach to single and multiple step environments in bin-packing prob-

lems. In *5th International Workshop on Learning Classifier Systems (IWLCS)*, pages 7–8, 2002.

Peter Cowling, Graham Kendall, and Eric Soubeiga. Hyperheuristics: A robust optimisation method applied to nurse scheduling. In *Parallel Problem Solving from Nature-PPSN VII*, pages 851–860. Springer, 2002b.

Peter Cowling, Graham Kendall, and Limin Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1185–1190. IEEE, 2002c.

EK Burke and JP Newall. A new adaptive heuristic framework for examination timetabling problems. *University of Nottingham, Working Group on Automated Timetabling, TR-2002-1 http://www. cs. nott. ac. uk/TR-cgi/TR. cgi*, pages 2002–1, 2002.

Jerry Swan, John Drake, Ender Özcan, James Goulding, and John Woodward. A comparison of acceptance criteria for the daily car-pooling problem. In *Computer and Information Sciences III*, pages 477–483. Springer, 2013.

Patricia Ryser-Welch and Julian F Miller. A review of hyper-heuristic frameworks. In *Proceedings of the 50th anniversary convention of the AISB, 1–4 April 2014, London*, 2014.

Eugene Nudelman, Kevin Leyton-Brown, Alex Devkar, Yoav Shoham, and Hol-

ger Hoos. Satzilla: An algorithm portfolio for sat. *Solver description, SAT competition*, 2004, 2004.

Horst Samulowitz, Chandra Reddy, Ashish Sabharwal, and Meinolf Sellmann. Snappy: A simple algorithm portfolio. In *Theory and Applications of Satisfiability Testing–SAT 2013*, pages 422–428. Springer, 2013.

Edmund K Burke, Tim Curtois, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Sanja Petrovic, and José Antonio Vázquez-Rodríguez. Hyflex: A flexible framework for the design and analysis of hyper-heuristics. In *Multidisciplinary International Scheduling Conference (MISTA 2009), Dublin, Ireland*, pages 790–797, 2009b.

Willem Van Onsem and Bart Demoen. Parhyflex: A framework for parallel hyper-heuristics. In *BNAIC 2013: Proceedings of the 25th Benelux Conference on Artificial Intelligence, Delft, The Netherlands, November 7-8, 2013*. Delft University of Technology (TU Delft); under the auspices of the Benelux Association for Artificial Intelligence (BNVKI) and the Dutch Research School for Information and Knowledge Systems (SIKS), 2013.

Mustafa Misir. Intelligent hyper-heuristics: a tool for solving generic optimisation problems. *status: published*, 2012.

Thompson G.L. Muth J.F. Industrial scheduling. *Prentice Hall*, pages 225–251, 1963.

Jacques Carlier and Eric Pinson. An algorithm for solving the job-shop problem. *Management science*, 35(2):164–176, 1989.

Wayne E Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

J. W. Barnes, J. J. Brennan, and R. M. Knap. Scheduling a backlog of oil well workovers. *Journal of Petroleum Technology*, 29(12):1651âĂŞ1653, 1977.

TF Noronha, FCJ Lima, and DJ Aloise. Um algoritmo heurístico guloso aplicado ao problema do gerenciamento das intervenções em poços petrolíferos por sondas de produção terrestre. In *Proceedings of the XXXIII Brazilian Symposium on Operations Research, Campos do Jordão*, page 135, 2001.

Dario J. Aloise, Daniel Aloise, Caroline T.M. Rocha, Celso C. Ribeiro, JosÃl' C. Ribeiro Filho, and Luiz S.S. Moura. Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, 154(5):695 – 702, 2006.

Glaydston Mattos Ribeiro, Geraldo Regis Mauri, and Luiz Antonio Nogueira Lorena. A simple and robust simulated annealing algorithm for scheduling workover rigs on onshore oil fields. *Computers & Industrial Engineering*, 60(4):519–526, 2011.

Christophe Duhamel, Andréa Cynthia Santos, and Lucas Moreira Guedes. Models and hybrid methods for the onshore wells maintenance problem. *Computers & Operations Research*, 39(12):2944–2953, 2012.

Glaydston Mattos Ribeiro, Gilbert Laporte, and Geraldo Regis Mauri. A comparison of three metaheuristics for the workover rig routing problem. *European Journal of Operational Research*, 220(1):28–36, 2012a.

Glaydston Mattos Ribeiro, Guy Desaulniers, and Jacques Desrosiers. A branch-price-and-cut algorithm for the workover rig routing problem. *Computers & Operations Research*, 39(12):3305–3315, 2012.

Glaydston Mattos Ribeiro, Guy Desaulniers, and Jacques Desrosiers. A branch-price-and-cut algorithm for the workover rig routing problem. *Computers & Operations Research*, 39(12):3305 – 3315, 2012b. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2012.04.010.

Glaydston Mattos Ribeiro, Guy Desaulniers, Jacques Desrosiers, Thibaut Vidal, and Bruno Salezze Vieira. Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon. *Journal of Heuristics*, 20(6):677–708, dec 2014. doi: 10.1007/s10732-014-9262-1.

Artur Pessoa, Eduardo Uchoa, Marcus Poggi de Aragão, and Rosiane Rodrigues. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2(3-4):259–290, 2010.

Edmund K Burke, Mustafa Misir, Gabriela Ochoa, and Ender Ozcan. Learning heuristic selection in hyper-heuristics for examination timetabling. In *Pro-*

ceedings of 7th International Conference of Practice and Theory of Automated Timetabling (PATAT08), Montreal, Canada, 2008b.

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

Sibel Alumur and Bahar Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

Shahin Gelareh, Rahimeh Neamatian Monemic, and Frédéric Semet. Capacitated bounded cardinality hub routing problem: Model and solution algorithm. Technical report, 2015.

J. Ebery, M. Krishnamoorthy, A. Ernst, and N. Boland. The capacitated multiple allocation hub location problem: Formulations and algorithms. *European Journal of Operational Research*, 120:614–631, 2000.

Isabel Correia, Stefan Nickel, and Francisco Saldanha-da Gama. *Single-allocation hub location problems with capacity choices*. Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, Fraunhofer (ITWM), 2009.

A. Campbell, T. Lowe, and L. Zhang. The p-hub center allocation problem. *European Journal of Operational Research*, 176 (2):819–835, 2007.

Juanjo Peiró, Ángel Corberán, and Rafael Martí. Grasp for the uncapacitated r-allocation p-hub median problem. *Computers & Operations Research*, 43: 50–60, 2014.

Dongdong Ge, Yinyu Ye, and Jiawei Zhang. The fixed-hub single allocation problem: a geometric rounding approach. *Preprint available at http://www. stanford. edu/˜ yyye/revisedHub. pdf*, 2007.

Shahin Gelareh, Nelson Maculan, Philippe Mahey, and Rahimeh Neamatian Monemi. Hub-and-spoke network design and fleet deployment for string planning of liner shipping. *Applied Mathematical Modelling*, 37(5):3307–3321, 2013a.

Jozef Kratica, Zorica Stanimirović, Dušan Tošić, and Vladimir Filipović. Genetic algorithm for solving uncapacitated multiple allocation hub location problem. *Computing and Informatics*, 24(4):415–426, 2012.

Masoud Rabbani and S Kazemi. Solving uncapacitated multiple allocation p-hub center problem by dijkstraŠs algorithm-based genetic algorithm and simulated annealing. *International Journal of Industrial Engineering Computations*, 6(3): 405–418, 2015.

Zorica Stanimirovic. An efficient genetic algorithm for the uncapacitated multiple allocation p-hub median problem. *Control and Cybernetics*, 37:669–692, 2008.

Marcus Randall. Solution approaches for the capacitated single allocation hub location problem using ant colony optimisation. *Computational Optimization and Applications*, 39(2):239–261, 2008.

Ching-Jung Ting and Chia-Ho Chen. A multiple ant colony optimization algorithm

for the capacitated location routing problem. *International Journal of Production Economics*, 141(1):34–44, 2013.

J.G. Klincewicz. Avoiding local optima in the p-hub location problem using tabu search and grasp. *Annals of Operations Research*, 40:283–302, 1992.

Erik Rolland, David A Schilling, and John R Current. An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research*, 96(2):329–342, 1997.

S. Abdinnour-Helm. A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 106(23):489–499, 1998.

Dilek Tuzun and Laura I Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.

MH Fazel Zarandi, S Davari, and SA Haddad Sisakht. An empirical comparison of simulated annealing and iterated local search for the hierarchical single allocation hub median location problem. *Scientia Iranica. Transaction E, Industrial Engineering*, 22(3):1203, 2015.

F Yu Vincent, Shih-Wei Lin, Wenyih Lee, and Ching-Jung Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, 58(2):288–299, 2010.

Aleksandar Ilić, Dragan Urošević, Jack Brimberg, and Nenad Mladenović. A general variable neighborhood search for solving the uncapacitated single allocation

p-hub median problem. *European Journal of Operational Research*, 206(2): 289–300, 2010.

Bassem Jarboui, Houda Derbel, Saïd Hanafi, and Nenad Mladenović. Variable neighborhood search for location routing. *Computers & Operations Research*, 40(1):47–57, 2013.

Gábor Nagy and Said Salhi. The many-to-many location-routing problem. *Top*, 6 (2):261–275, 1998.

S. Cetiner, C. Sepil, and H. Sural. Hubbing and routing in postal delivery systems. Technical report, Industrial Engineering Department, Middle East Technical University, 06532 Ankara, Turkey., 2006.

Ricardo Saraiva de Camargo, Gilberto de Miranda, and Arne Løkketangen. A new formulation and an exact approach for the many-to-many hub location-routing problem. *Applied Mathematical Modelling*, 37(12):7465–7480, 2013.

Michael Wasner and Günther Zäpfel. An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. *International Journal of Production Economics*, 90(3):403–419, 2004.

Shahin Gelareh, Nelson Maculan, Philippe Mahey, and Rahimeh Neamatian Monemi. Hub-and-spoke network design and fleet deployment for string planning of liner shipping. *Applied Mathematical Modelling*, 37(5):3307–3321, 2013b.

A. M. Geoffrion. Lagrangian relaxation and its uses in integer programming. *Math. Programming Stud*, 2:82–114, 1974.

A Geoffrion and R Me Bride. Lagrangean relaxation applied to capacitated facility location problems. *AIIE transactions*, 10(1):40–47, 1978.

Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981. ISSN 00251909.

Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 50(12_supplement):1861–1871, 2004.

M. Guignard. Lagrangian relaxation. *TOP*, 11(2):151–228, 2003.

M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1(1):6–25, 1971. ISSN 0025-5610.

C. Lemarechal. An extension of Davidon methods to non differentiable problems. *Nondifferentiable Optimization*, pages 95–109, 1975.

F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000. ISSN 0025-5610.

Laura Bahiense, Nelson Maculan, and Claudia A. Sagastizábal. The volume algorithm revisited: relation with bundle methods. *Math. Program.*, 94(1): 41–69, 2002.

Antonio Frangioni. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. Technical report, 1995.

Herman Aguinis, Lura E Forcum, and Harry Joo. Using market basket analysis in management research. *Journal of Management*, page 0149206312466147, 2012.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

Diamantino Falcao, Ana Madureira, and Ivo Pereira. Q-learning based hyper-heuristic for scheduling system self-parameterization. In *Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on*, pages 1–7. IEEE, 2015.

Yuanyuan Xi, Liuchen Chang, Meiqin Mao, Peng Jin, Nikos Hatziargyriou, and Haibo Xu. Q-learning algorithm based multi-agent coordinated control method for microgrids. In *Power Electronics and ECCE Asia (ICPE-ECCE Asia), 2015 9th International Conference on*, pages 1497–1504. IEEE, 2015.

Justin A Boyan and Michael L Littman. Packet routing in dynamically changing

networks: A reinforcement learning approach. *Advances in neural information processing systems*, pages 671–671, 1994.

Meijuan Gao, Jin Xu, and Jingwen Tian. Mobile robot global path planning based on improved augment ant colony algorithm. In *Genetic and Evolutionary Computing, 2008. WGEC'08. Second International Conference on*, pages 273–276. IEEE, 2008.

Marco Dorigo and LM Gambardella. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning*, pages 252–260, 2014.

Junius Ho, Daniel W Engels, and Sanjay E Sarma. Hiq: a hierarchical q-learning algorithm to solve the reader collision problem. In *Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on*, pages 4–pp. IEEE, 2006.

S Choi and Dit-Yan Yeung. Predictive q-routing: A memory-based reinforcement learning approach to adaptive tra c control. *Advances in Neural Information Processing Systems*, 8:945–951, 1996.

Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, pages 417–428. Springer, 1999.

A.T. Ernst and M. Krishnamoorthy. Solution algorithms for the capacitated single

allocation hub location problem. *Annals of Operations Research*, 86:141–159, 1999.

Morton E O'kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.

Kate A Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6, 2008.

Melanie Hilario, Alexandros Kalousis, Phong Nguyen, and Adam Woznica. A data mining ontology for algorithm selection and meta-mining. In *Proceedings of the ECML/PKDD09 Workshop on 3rd generation Data Mining (SoKD-09)*, pages 76–87, 2009.

Ender Özcan and Ahmed Kheiri. A hyperheuristic based on random gradient, greedy and dominance. In *Computer and Information Sciences II*, pages 557–563. Springer, 2012.

John H Drake, Ender Ozcan, and Edmund K Burke. A modified choice function hyperheuristic controlling unary and binary operators. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2015)*, 2015.

E Burke and E Soubeiga. Scheduling nurses using a tabu-search hyperheuristic. In *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003), Nottingham, UK*, pages 180–197, 2003.

Peter Cowling and Konstantin Chakhlevitch. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Evolutionary*

Computation, 2003. CEC'03. The 2003 Congress on, volume 2, pages 1214–1221. IEEE, 2003.

Peter I Cowling and Konstantin Chakhlevitch. Using a large set of low level heuristics in a hyperheuristic approach to personnel scheduling. In *Evolutionary Scheduling*, pages 543–576. Springer, 2007.

Edmund K Burke and Graham Kendall. *Search methodologies*. Springer, 2005.

Ender Özcan, Mustafa Mısır, Gabriela Ochoa, and Edmund K Burke. A reinforcement learning: Great-deluge hyperheuristic. *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends*, page 34, 2012.

Graham Kendall, Peter Cowling, and Eric Soubeiga. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, pages 667–671. Citeseer, 2002b.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Edwin T Jaynes. Bayesian methods: General background. 1986.

Bruno De Finetti. *Theory of Probability: A critical introductory treatment. Vol. 2*. Wiley, 1974.

David Heckerman. Bayesian networks for data mining. *Data mining and knowledge discovery*, 1(1):79–119, 1997.

Efstathios Kirkos, Charalambos Spathis, and Yannis Manolopoulos. Data mining techniques for the detection of fraudulent financial statements. *Expert Systems with Applications*, 32(4):995–1003, 2007.

Murat Kantarcıoglu, Jaideep Vaidya, and C Clifton. Privacy preserving naive bayes classifier for horizontally partitioned data. In *IEEE ICDM workshop on privacy preserving data mining*, pages 3–9, 2003.

Iñaki Inza, Pedro Larrañaga, Ramón Etxeberria, and Basilio Sierra. Feature subset selection by bayesian network-based optimization. *Artificial intelligence*, 123 (1):157–184, 2000.

Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

Martin Pelikan. Bayesian optimization algorithm. In *Hierarchical Bayesian optimization algorithm*, pages 31–48. Springer, 2005.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Shu-Mei Ma, Yun Sun, and Ai-Ping Li. A simulated annealing algorithm for multi-objective hybrid flow shop scheduling work partially supported by national major science and technology of high-grade cnc machine tools and basic manufacturing equipment special project of china under grant no. 2013zx04012-071. 2016.

KLEIN Bouleimen and HOUSNI Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 2003.

EB Schlünz and JH Van Vuuren. An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. *International Journal of Electrical Power & Energy Systems*, 53: 166–174, 2013.

Samaneh Sedehzadeh, Reza Tavakkoli-Moghaddam, Mehrdad Mohammadi, and Fariborz Jolai. Solving a new priority m/m/c queue model for a multi-mode hub covering location problem by multi-objective parallel simulated annealing. *Economic Computation and Economic Cybernetics studies and Research*, 48(4): 299–318, 2014.

F Parvaresh, SA Hashemi Golpayegany, SM Moattar Husseini, and B Karimi. Solving the p-hub median problem under intentional disruptions using simulated annealing. *Networks and Spatial Economics*, 13(4):445–470, 2013.

EH Grosse, CH Glock, and R Ballester-Ripoll. A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. Technical report, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), 2014.

RaúL BañOs, Julio Ortega, Consolación Gil, Antonio FernáNdez, and Francisco De Toro. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40(5):1696–1707, 2013.

F Yu Vincent and Shih-Wei Lin. Multi-start simulated annealing heuristic for the location routing problem with simultaneous pickup and delivery. *Applied Soft Computing*, 24:284–290, 2014.

Glaydston Mattos Ribeiro, Geraldo Regis Mauri, and Luiz Antonio Nogueira Lorena. A simple and robust simulated annealing algorithm for scheduling workover rigs on onshore oil fields. *Computers & Industrial Engineering*, 60(4): 519–526, 2011.

RO Paiva et al. Optimizing the itinerary of workover rigs. In *16th World petroleum congress*. World Petroleum Congress, 2000.

Kathryn A Dowsland, Eric Soubeiga, and Edmund Burke. A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *European Journal of Operational Research*, 179(3):759–774, 2007.

Ruibin Bai, Jacek Blazewicz, Edmund K Burke, Graham Kendall, and Barry McCollum. A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR*, 10(1):43–66, 2012.

Konstantinos P Anagnostopoulos and Georgios K Koulinas. A simulated annealing

hyperheuristic for construction resource levelling. *Construction Management and Economics*, 28(2):163–175, 2010.

J Chávez, J Escobar, and M Echeverri. A multi-objective pareto ant colony algorithm for the multi-depot vehicle routing problem with backhauls. *International Journal of Industrial Engineering Computations*, 7(1):35–48, 2016.

Yuvraj Gajpal and Prakash Abad. An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36(12):3215–3223, 2009.

KA Dowsland and JM Thompson. Ant colony optimization for the examination scheduling problem. *Journal of the Operational Research Society*, 56(4):426–438, 2005.

Masri Ayob and Ghaith Jaradat. Hybrid ant colony systems for course timetabling problems. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 120–126. IEEE, 2009.

Berna Kiraz, A Şima Etaner-Uyar, and Ender Özcan. *An ant-based selection hyper-heuristic for dynamic environments*. Springer, 2013.

Alexandre Silvestre Ferreira, POZO Aurora, and Richard Aderbal Gonçalves. An ant colony based hyper-heuristic approach for the set covering problem. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 4(1): 1–21, 2015.

Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997b.

Ping-Che Hsiao, Tsung-Che Chiang, and Li-Chen Fu. A vns-based hyper-heuristic with adaptive computational budget of local search. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

Pierre Hansen and Nenad Mladenović. *Variable neighborhood search*. Springer, 2014.

Stephen Remde, Peter Cowling, Keshav Dahal, and Nic Colledge. Exact/heuristic hybrids using rvns and hyperheuristics for workforce scheduling. In *Evolutionary Computation in Combinatorial Optimization*, pages 188–197. Springer, 2007.

GM Ribeiro, G Desaulniers, J Desrosiers, T Vidal, and BS Vieira. Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon. 2013.

Glaydston Mattos Ribeiro, Gilbert Laporte, and Geraldo Regis Mauri. A comparison of three metaheuristics for the workover rig routing problem. *European Journal of Operational Research*, 220(1):28–36, 2012c.

Hyperheuristics in Logistics

Success in using exact methods for large scale combinatorial optimization is still limited to certain problems or to specific classes of instances of problems. The alternative way is either using metaheuristics or matheuristics.In the context of combinatorial optimization, we are interested in heuristics to choose heuristics invoked to solve the addressed problem. In this thesis, we focus on hyperheuristic optimization in logistic problems. We focus on proposing a hyperheuristic framework that carries out a search in the space of heuristic algorithms and learns how to change the incumbent heuristic in a systematic way along the process.We propose HHs for two optimization problems in logistics: the workover rig scheduling problem and the hub location routing problem. Then, we compare the performances of several HHs described in the literature for the latter problem, which embed different heuristic selection methods such as a random selection, a choice function, a Q-Learning approach, and an ant colony based algorithm. The computational results prove the efficiency of HHs for the two problems in hand, and the relevance of including Lagrangian relaxation information for the second problem.
Keywords: Metaheuristic, Heuristic, Hyperheuristic, Matheuristic, Reinforcement learning, Hub location problem, Workover rig scheduling problem, Association rules.

Hyperheuristiques pour des problèmes d'optimisation en logistique

Le succeès dans l'utilisation de méthodes exactes d'optimisation combinatoire pour des problèmes de grande taille est encore limité à certains problèmes ou à des classes spécifiques d'instances de problèmes. Une approche alternative consiste soit à utiliser des métaheuristiques ou des matheuristiques.Dans le contexte de l'optimisation combinatoire, nous nous intéressons des heuristiques permettant de choisir les heuristiques appliquées au problème traité. Dans cette thèse, nous nous concentrons sur l'optimisation à l'aide d'hyperheuristiques pour des problèmes logistiques. Nous proposons un cadre hyperheuristique qui effectue une recherche dans l'espace des algorithmes heuristiques et apprend comment changer l'heuristique courante de manière systématique tout au long du processus. Nous étudions plus particulièrement deux problèmes d'optimisation en logistique pour lesquels nous proposons des HHs: un problème de planification d'interventions sur des puits de forage et un problème conjoint de localisation de hubs et de routage. Ensuite, nous comparons les performances de plusieurs HH décrites dans la littérature pour le second problème abordé reposant sur différentes méthodes de sélection heuristique. Les résultats numériques prouvent l'efficacité de HHs pour les deux problèmes traités, et la pertinence d'inclure l'information venant d'une relaxation de Lagrangienne pour le deuxième problème.
Mots-clefs: Metaheuristique, Heuristique, Hyperheuristique, Matheuristique, Apprentissage par renforcement, Problème de localisation des concentrateurs, Problème d'ordonnancement de workover, Règles d'association.