Summer 2015

Meta-RaPS Hybridization with Machine Learning Algorithms

Fatemah Al-Duoli
Old Dominion University

META-RAPS HYBRIDIZATION WITH MACHINE LEARNING ALGORITHMS

by

Fatemah Al-Duoli
B.S. August 2003, Old Dominion University
M.S. May 2005, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT

OLD DOMINION UNIVERSITY
August 2015

Approved by:

Ghaith Rabadi (Director)

Resit Unal (Member)

Pilar Pazos (Member)

Christopher Garcia (Member)

ABSTRACT

META-RAPS HYBRIDIZATION WITH MACHINE LEARNING ALGORITHMS

Fatemah Al-Duoli
Old Dominion University, 2015
Director: Dr. Ghaith Rabadi

This dissertation focuses on advancing the Metaheuristic for Randomized Priority Search algorithm, known as Meta-RaPS, by integrating it with machine learning algorithms. Introducing a new metaheuristic algorithm starts with demonstrating its performance. This is accomplished by using the new algorithm to solve various combinatorial optimization problems in their basic form. The next stage focuses on advancing the new algorithm by strengthening its relatively weaker characteristics. In the third traditional stage, the algorithms are exercised in solving more complex optimization problems. In the case of effective algorithms, the second and third stages can occur in parallel as researchers are eager to employ good algorithms to solve complex problems. The third stage can inadvertently strengthen the original algorithm. The simplicity and effectiveness Meta-RaPS enjoys places it in both second and third research stages concurrently. This dissertation explores strengthening Meta-RaPS by incorporating memory and learning features. The major conceptual frameworks that guided this work are the Adaptive Memory Programming framework (or AMP) and the metaheuristic hybridization taxonomy. The concepts from both frameworks are followed when identifying useful information that Meta-RaPS can collect during execution. Hybridizing Meta-RaPS with machine learning algorithms helped in transforming the collected information into knowledge. The learning concepts selected are supervised and unsupervised learning. The algorithms selected to achieve both types of learning are the Inductive Decision Tree (supervised learning) and Association Rules (unsupervised learning). The objective behind hybridizing Meta-RaPS with an Inductive Decision Tree algorithm is to perform online control for Meta-RaPS' parameters. This Inductive Decision Tree algorithm is used to find favorable parameter values using knowledge gained from previous Meta-RaPS iterations. The values selected are used in future Meta-RaPS iterations. The objective behind hybridizing Meta-RaPS with an Association Rules

algorithm is to identify patterns associated with good solutions. These patterns are considered knowledge and are inherited as starting points for in future Meta-RaPS iteration. The performance of the hybrid Meta-RaPS algorithms is demonstrated by solving the capacitated Vehicle Routing Problem with and without time windows.

Rob, thank you for continuously lifting me up from the many local minima I plunged into during this journey. This dissertation is dedicated to you.

ACKNOWLEDGEMENTS

This work is the result of the unremitting support of Dr. Ghaith Rabadi. Thank you for your time, insights, and guidance. Thank you to the committee members for the encouragement that sustained this effort.

To my parents, brothers, in-laws, friends, ENMA department staff, and my wooliest devotee Cooper, I am grateful for the roles you played in this journey. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

The field of metaheuristics is one of the active combinatorial optimization fields due to the quality of solutions metaheuristics produce for variety of hard combinatorial optimization problems. Research in this field tends to fall into two high-level categories: algorithms or applications. In the first category the focus is on introducing or improving metaheuristics algorithms while the second category focuses on the application of metaheuristics to new complex problems. This dissertation work falls in the first category. The chief objective is to contribute to an existing, relatively new metaheuristic algorithm named Metaheuristics for Randomized Priority Search or Meta-RaPS. This is achieved by hybridization Meta-RaPS with machine learning algorithms.

While many have hybridized existing metaheuristics by combining two or more metaheuristic algorithms, very little research attempted to hybridize metaheuristics with data mining algorithms, a branch of machine learning and artificial intelligence. The dissertation details the process of hybridizing Meta-RaPS with both a classification rule association algorithms. The proposed hybridization concept is classified as a *High-level Relay Hybridization* with *heterogeneous* algorithms that are applied to the *global* search domain with a *general* (same) problem to solve (Talbi, 2009). The effectiveness of the hybridized Meta-RaPS is demonstrated by solving the classical Vehicle Routing Problem, an NP-hard combinatorial optimization problem.

The next chapter will showcase various known metaheuristic algorithms and focus on Meta-RaPS. A literature review of various publications that applied Meta-RaPS to solve combinatorial optimization problems is also included in Chapter 2. Chapter 3 discusses the hybridization of metaheuristic algorithms especially the existing paradigms of hybridization. Subsequently, hybridizing metaheuristic algorithms with data mining techniques are discussed. Chapter 3 ends with a deeper look at both classification and association rules, the two data mining techniques that will be used to hybridize Meta-

RaPS. The combinatorial optimization problem of choice is the Vehicle Routing Problem. The problem, its variants, and the various optimization approaches to solve it will be discussed in Chapter 4. A literature review of well-performing metaheuristics used to solve the classic Vehicle Routing Problem is included in Chapter 4. Chapter 5 describes the application of Meta-RaPS to solve the Vehicle Routing Problem. Chapters 6 and 7 describe the hybridization frameworks of Meta-RaPS with both data mining algorithms. Lastly, Chapter 8 details the methodology followed in creating the computational experiments, the various trails examined under each hybridized experiment, and the associated computation results and analysis.

CHAPTER 2

METAHEURISTICS AND META-RAPS

In this chapter, the focus will be on defining and classifying metaheuristics algorithms used to solve combinatorial optimization problems. Most of the commonly used metaheuristics algorithms will be briefly discussed. The chapter will be concluded by discussing Meta-RaPS in greater detail by covering its history, logic, and applications.

2.1 Combinatorial Optimization

The fields of mathematics, computer science and engineering overlap when tackling optimization problems where the goal is to find an optimal solution. Optimization problems are described by (S, f). S represents the search space where a set of feasible solutions exist for the problem while f is the objective function that ranks the feasible solutions and yields the highest (or lowest) ranking solution. In Figure 1, Talbi (2009) organized the various approaches to solving optimization problems. The approach of interest is the Combinatorial Optimization one, which aims at finding the optimal (or best) solution for problems that consist of discrete variables in a finite search space (Blum & Roli, 2003).

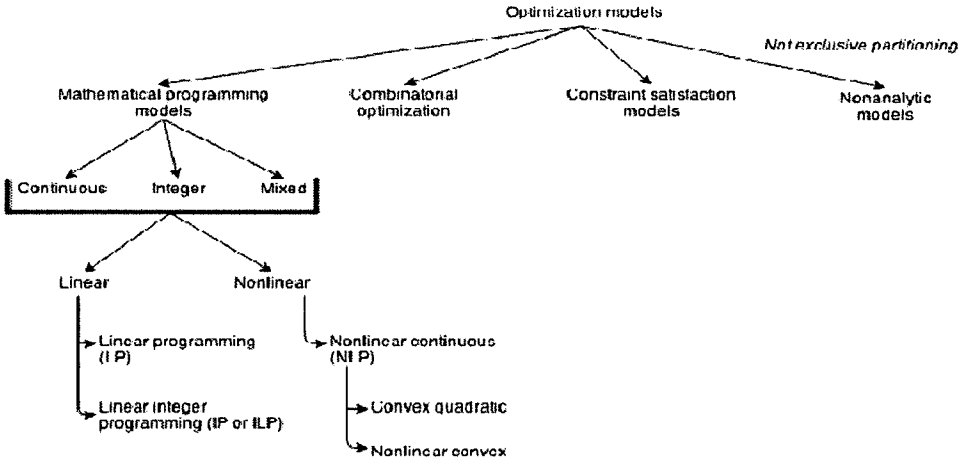


Figure 1 Optimization Models (Talbi 2009)

One of the fundamental combinatorial optimization problems is the Traveling Salesman Problem (TSP). The problem consists of cities and the distance between every city. The objective of function aims at finding the shortest route that consists of visiting each city once. The challenge is in the number of possible solutions that increases as the size of the number of cities increases. A large number of possible solutions would need to be searched and evaluated prior to determining a global optimal solution. Table 1 shows the relationship between the number of cities in a TSP (n) and the number of possible solutions (n!). The following figure also visually illustrates two TSP problems of different sizes.

Number of Cities (n)	Size of the Search Space
5	120
10	3, 628, 800
75	2.5×10^{109}

Table 1 Number of Cities in a TSP and the Corresponding Search Space (Talbi, 2009)

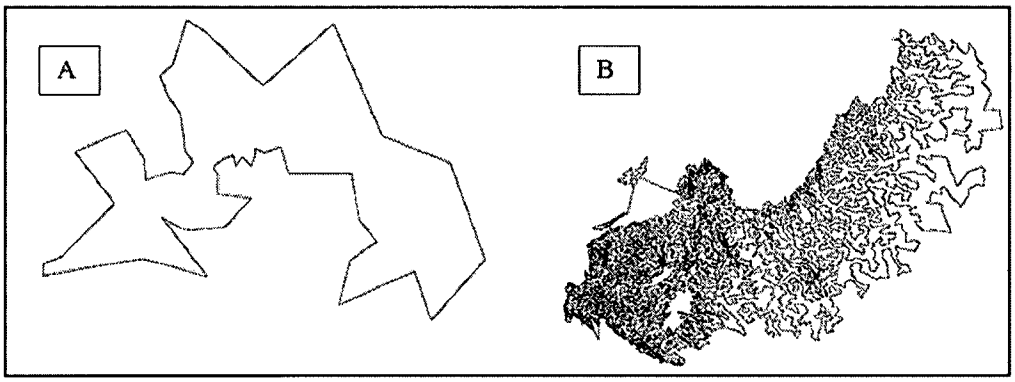


Figure 2 TSP A wih 52 Cities; TSP B with 24,9789 Cities (Talbi, 2009)

The Knapsack problem is another known combinatorial optimization problem that aims at identifying the set of objects that fit the size (or weight) constraint of given knapsacks. Another known combinatorial optimization problem is the Quadratic Assignment Problem (QAP), where the objective is to find an optimal location assignment of facilities/nodes based on the flows between each pair of facilities. The literature contains numerous examples of real-world combinatorial optimization problems. Many of these real-world problems can be mapped to one of the few fundamental combinatorial optimization problems. For example, the problem of finding an optimal schedule for aircraft to land and depart in an airport is mapped to a version of the TSP (Bianco et al., 1999).

The main challenge in solving real-world combinatorial problems lies in their large size and the complexity of their structures (Talbi, 2009). Solving a combinatorial optimization problem usually entails evaluating several candidate solutions and identifying the best one. Small problems are typically solved using exact algorithms. Exact algorithms always find a global optimal solution for a finite size combinatorial

optimization problem in a bounded time. Examples of exact algorithms include constraint programming, dynamic programming, and branch and bound (or branch and cut, or branch and price).

For larger problems, approximate methods are utilized. For many larger, more complex combinatorial optimization problems, researchers typically theoretically prove that no polynomial time exists to solve them and find a global optimal solution. Though approximate methods do not guarantee finding a global optimum, they produce satisfactory solutions in a reasonable amount of time. Approximate methods are classified as approximate algorithms and heuristic algorithms (Talbi, 2009). Approximate algorithms produce provable solutions in a provable run-time while heuristic algorithms provide good solutions in an acceptable amount of time. Heuristic algorithms comprise of classic heuristics and metaheuristics. Classic heuristics tend to be basic and problem-specific. Metaheuristics are more general-purpose algorithms that employ one or more classic heuristics.

2.2 Metaheuristics

Glover defined the first metaheuristic algorithm, Tabu Search (TS), as an algorithm that is "superimposed" on a heuristic algorithm (Glover, 1986). Glover's TS is attributed as the birth of the field of metaheuristic methods. With more metaheuristic activities, several authors attempted to create a definition for this new group of methods. Stutzle (1999) stated that metaheuristic methods are high-level strategies that guide more problem-specific heuristics, to increase their performance.

Metaheuristics generally consist of two phases: construction and improvement. These phases host underlying and more-specific heuristics. The construction phase aims to build a solution for the combinatorial optimization problem using one or more heuristic algorithms. Subsequently, the improvement phase attempts to advance the newly built solution also using one or more heuristic algorithms. As a high-level method, a metaheuristic is in charge of managing the underlying heuristics to balance their

objectives. This is crucial since the heuristic(s) in the construction phase tends to have a diversification outlook when building a solution by exploring the solution space. On the other hand, the heuristic(s) in the improvement phase tends to have an intensification outlook when improving a solution by focusing on a narrow area surrounding the newly built solution when attempting to improve it (Lima et al., 2008).

Some of the well-known metaheuristic methods are: Tabu Search, Genetic Algorithms, Simulated Annealing, Iterated Local Search, Variable Neighborhood Search, Artificial Neural Networks, Ant Colony Optimization, and Greedy Randomized Adaptive Search Procedure (GRASP). The characteristics of the metaheuristics allow for more than one way to categorize them. Blum and Roli (2003) explored several viable categorization approaches:

1. Categorization based on number of solutions used at the same time: population-based methods vs. single solution based methods. Population-based methods work on a set of points in the search space and aim to describe the evolution of this set throughout the search (e.g. Genetic Algorithms) while single solution based methods, often called trajectory methods, work with a single solution by employing local search method (e.g. Tabu Search and GRASP).

2. Categorization based on type of objective function: The objective function for a problem can be static (not modified) or dynamic. While the majority of metaheuristic methods do not modify the objective function, the Guided Local Search method updates the objective function as an approach to avoid local optima.

3. Categorization based on neighborhood structure: Talbi (2009) defined the neighborhood structure as a function that assigns to each solution a set of solutions (or neighbors) that can be defined by performing small moves to the original solution. Metaheuristic methods may use either single or multiple neighborhood structures while exploring the search space. Working with multiple

neighborhood structures, as done in Variable Neighborhood Search method, allows for more diversification.

4. Categorization based on memory usage: Metaheuristic methods can be categorized based on the type of information used when a metaheuristic method is deciding its next action. If a method only uses information from its current state, then it can be categorized as memory-less. In memory-based methods, however, previously stored information (such as previous moves, visited solutions, decisions, etc.) can impact the next action. Memory usage can either be a core feature or a complementary feature found in advanced versions of certain methods. Examples of metaheuristic methods using memory as a core feature include Ant Colony Optimization and Tabu Search, while Simulated Annealing represents an example of a metaheuristic using memory in its advanced versions.

Categorizing this wide variety of metaheuristic methods allows researchers to select the suitable method to solve combinatorial optimization problem(s). Additionally, this helps in potential collaboration between the existing method and introduction of new methods to the field of metaheuristics. The following sections highlight some of the established metaheuristics.

2.2.1 Genetic Algorithms (GA)

Several optimization algorithms have been developed to utilize the principles of natural evolution in solving combinatorial optimization problems. The idea of a population of individuals iteratively reproducing and competing to survive is represented in various algorithms. These algorithms include Genetic Algorithms (GA) (Holland, 1962), evolution strategies (Rechenberg, 1965), evolutionary programming (Fogel, 1962), and genetic programming (Koza, 1992). One of the first and most popular evolutionary algorithms is GA developed by Holland (1962). An initial solution is randomly generated from an initial population. The solution is evaluated and individuals/solutions are selected based on a fitness function. The selected individuals/solutions represent parents for new population. These parents are recombined using the crossover and/or the mutation genetic operators. Lastly, a replacement process is applied to determine the surviving solutions from the new generation. The evolutionary process repeats itself until a stopping criterion is met.

2.2.2 Simulated Annealing (SA)

This method is inspired by the annealing process in metallurgy. To obtain large crystals with little defects, the annealing process involves heating and slowly cooling a metal. The strength of the metal depends on the cooling process. Kirkpatrick (1983) applied the concept of slowly cooling a metal to solving combinatorial optimization problems. The algorithm starts with generating an initial solution and setting the temperature to its maximum level. Then, a neighboring solution is generated randomly. If the new neighboring solution is better than the initial solution, then it is accepted as the new best solution. Else, the new solution is accepted with the probably that depends on the difference in values between the objective function of the best solution and the new solution and the current system temperature. At a given temperature, a large difference in objective function values leads to a higher probability of accepting a solution. The temperature decreases according to a cooling scheduling provided as an input to the algorithm. Decreasing the temperature lowers the probably of accepting new solutions

that do not significantly differ from previous solutions. This process is repeated until a minimum temperature is reached.

2.2.3 Tabu Search (TS)

Glover (1986) introduced a strategy that keeps track of visited solutions in order to assist a local search method in overcoming local optima. By doing so, Glover introduced memory structures to the growing field of metaheuristics. The type of memory simulated by tabu list depends on the information saved. Memory can be classified into:

- Tabu List or Short-term memory: contains specific moves based on the latest solution. Such moves represent the structure of the recent search trajectory, or move that were not taken in the recent solution. Short-term memory aims at preventing cycling or re-visiting recent solutions. The Tabu list is updated after every iteration. Tabu moves that produce better solutions are allowed. This is referred to as the aspiration criterion, which is evaluated when creating new solution.
- Intermediate-term memory: contains best (elite) solutions found throughout the search. Memorizing these elite solutions and their attributes impacts future iterations by giving priority to solutions that resembles them. Intermediate-term memory helps achieve intensification effect.
- Long term memory: contains information about solutions generated throughout the search. The goal behind using such information is to avoid revisiting old solutions' regions and promote exploring unvisited regions of the search space. Long-term memory helps achieve diversification effect.

The pseudo code below shows the general steps followed when executing a TS algorithm.

```
S = S0
Initialize tabu list, medium-term memory, long-term memory
Repeat
        S' = Best neighbor
        S = S'
        Update tabu list, aspiration conditions, medium-term memory, long-term memory
        If intensification criterion holds, then intensify
        If diversification criterion holds, then diversify
Until termination condition met
```

Figure 3 Tabu Search Algorithm (Talbi, 2009)

The process starts with creating an initial (and best by default) solution and initializing the various types of memories employed. The neighbors of the (initial) solution are created and the best neighbor is selected as the new best solution. The Tabu list, along with other employed memories, is updated to include non-admissible moves. In the following iteration, the process repeats until a local optimum is reached. At this point, the search continues by selecting a worse solution to be the next solution.

2.2.4 Iterated Local Search (ILS)

Applying local search methods to a solution is expected to assure that the solution is a local optimum. Therefore, applying a local search to a solution in a poor region, would lead to improvement or local optima. However, this improvement is only relative to the quality of the solution (or input given to the local search). With this motive, ILS strategy aims at repeating the application of local search to solution produced in various neighborhoods by using a Perturbation method. This method modifies the initial solution in order to provide a new starting point to the local search method. Perturbation can be very random to lead to more diversification or less random (by keeping parts of the initial solution).

ILS effectively combines local search and perturbation by executing the following procedures: Generate Initial Solution, Local Search, Perturbation, Local Search and Acceptance Criterion (Gendreau et al., 2010). The local search method can be a heuristic or a single-solution metaheuristic. ILS starts with 'Generate Initial Solution' and 'Local

Search', which represent a common start to many metaheuristic methods where an initial solution is constructed and improved

```
S0 = Generate initial solution
S* = Local search (S0)
Repeat
        S' = Perturbation (S*, history)
        S*' = Locals search (S')
        S* = Acceptance criterion (S*, S*', history)
Until termination condition met
```

Figure 4 ILS Algorithm (Gendreau et al., 2010)

Perturbation modifies the neighborhood of the recently created solution before Local Search is applied again. Perturbation allows ILS "to effectively escape local optima but at the same time avoid the disadvantage of a random start" (den Besten et al., 2001). Following the perturbation, a 'Local Search' is applied to assure local optima. Then, an 'Acceptance Criterion' is evaluated. This procedure employs a policy that enforces either diversification or intensification depending on the problem. The process is repeated until a termination condition is met. The concepts behind ILS have been utilized in various publications; however, Lourenco et al. (2002) formalized ILS as a metaheuristic.

2.2.5 Variable Neighborhood Search (VNS)

Unlike local search-based metaheuristics, VNS has the ability to omit a trajectory search and systematically change neighborhood structures. A neighborhood structure is exited as a result of reaching local optima. VNS uses this mechanism to achieve a global optimum, which is expected to be a local minimum of all neighborhood structures. VNS was first introduced by Mladenovic and Hansen (1997). Two main procedures were described: initialization and main step (figure below). During the initialization procedure, a set of neighborhood structures are selected and an initial solution (S_{best}) is created using the first neighborhood structure.

```
Initialization:
Select a set of neighborhood structures Nk
  Let Nk be: k=1, ... , kmax
  Sbest = Generate an initial solution
Main Step:
Repeat
  Let k = 1
  Repeat
          S* = Generate a random solution from neighborhood k //Shake
          S*' = Local Search (S*)
          If S*' < Sbest
             Sbest = S*'
             k=1
          Else
             k ++
Until k = kmax
Until termination condition met
```

Figure 5 Pseudo Code for VNS (Gomes et al., 2000)

In the main step, a solution is generated at random from the k^{th} neighborhood structure. This is followed by an improvement via a local search method. If the recently created solution is worse than the previous (initial in this case) solution, then the current neighborhood structure is exited (k++). This indicates that a local optimum has been reached in a neighborhood structure. On the other hand, if the new solution is better than the previous/best solution, then searching neighborhood k=1 continues by repeating the steps starting with the shaking process in an effort to find a better neighbor.

2.2.6 Guided Local Search (GLS)

The core idea behind GLS is to vary the objective function based on the output of a local search method (Voudouris, 1997). This is done by defining features associated with a local optimum, associating costs for the features, and incorporating penalties to reflect the importance of the features. For a TSP instance, a feature can be a specific connection between two cities while the cost is the distance associated with this connection. The pseudo code illustrates the general steps following by GLS:

```
S0 = Generate initial solution
S=S0
Pi = 0 //Initialize penalties

Repeat
        S* = Local search (S)
        For each feature i of S*
                u_i = c_i / (1+ p_i) //compute feature's utility
                u_j = max_{i=1,...,m} (u_i) //compute maximum utilities
                p_j = p_j + 1 //update penalties

Until termination condition met
```

Figure 6 Guided Local Search Algorithm (Talbi, 2009)

After defining the features and their associated costs, the process starts by creating an initial solution and initializing the penalties to zero. A local search method is applied to the initial solution in an effort to reach the local optimum. For each presence feature in the solution, the utility is computed, which is proportional to the cost of the feature and inverse to the penalty of the feature. The penalty of the feature with the highest utility is increased. If a feature is not presence in the solution, then its utility is equal to 0 (and ultimately its penalty is not increased). Incorporating the penalty concept into the objective function allows GLS to explore the search space by avoiding highly penalized regions.

2.2.7 Ant Colony Optimization (ACO)

Creatures such as ants, bees, wasps, fish, birds, etc. have mastered intelligent strategies that help them survive. These species are able to indirectly communicate knowledge about the best paths to take in order to achieve a specific goal (e.g. best paths to food). The concepts behind these strategies have been formulated into metaheuristic algorithms and successfully applied to combinatorial optimization methods. These

metaheuristics algorithms are known as swarm intelligence algorithms (Kennedy & Eberhart, 2001).

One of the most successful swarm intelligence algorithms is the Ant Colony Optimization. Inspired by ants in the real world, Dorigo made the connection between ants seeking food and optimization (Blum, 2005). Capturing the indirect communication between ants in quest for food, Dorigo introduced the Ant System and applied it to the TSP. Ant System matured to the become Ant Colony Optimization (ACO) algorithm. The ants-focused process starts with ants depositing chemical pheromones in all the routes they take leading to the food source. Ants that chose the shortest route are the ones that will reach the food and return to the nest first. On their way back to the nest, these ants will continue to deposit chemical pheromones. When a new group of ants leave the nest with the same objective, they will more likely follow the route with the highest pheromone concentration, which is the shortest route to the food. With time, ants will repeat this process and eliminate following long routes to get to their food. This phenomenon is known as stigmegry, a pheromone-based communication method ants use when searching for the shortest way to find food (Dorigo et al., 2000).

The process followed by artificial ants in ACOs is represented in the pseudo code below. Artificial ants mimic the behavior of real ants by using heuristic information as their artificial pheromone trail.

```
Initialize pheromone trail

Repeat
        For each ant
                Construct solution using pheromone trail
                Update pheromone trails

Until termination condition met
```

Figure 7 ACO Pseudo Code (Talbi, 2009)

2.2.8 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP was introduced by Feo and Bard (1995). It is a simple, iterative, single-solution based metaheuristic. GRASP constructs a solution using a greedy heuristic and improves it using a local search heuristic. The solution gets constructed incrementally one element at a time. During the process of constructing a solution, GRASP generates a Restricted Candidate List (RCL) to include candidate elements that could be chosen next. The elements in the RCL are ordered using the greedy, local heuristic. RCL represents the adaptive aspect of GRASP as it gets updated after every iteration, which is done when an element is selected from it. Selecting an element from the RCL can be achieved following cardinality-based criteria or value-based criteria (Talib, 2009). When using the cardinality-based criteria, the best element in RCL is selected next while using the value-based criteria requires using an additional parameter to be used as a cutoff point, where an element better than the parameters is selected randomly. Once the solution is completed, it gets improved using a local search heuristic to guarantee that it is a local optimum.

2.3 Metaheuristics for Randomized Priority Search (Meta-RaPS)

While the aforementioned metaheuristic methods are some of the most studied metaheuristic methods, this study focuses on a relatively new metaheuristic method: Metaheuristic for Randomized Priority Search (Meta-RaPS) introduced by DePuy and her colleagues in 2001 (DePuy et al., 2001). Meta-RaPS is based on the Computer Method of Sequencing Operations for Assembly Lines (COMSOAL) heuristic. COMSOAL was first used to solve the assembly line balancing problem (Arcus, 1965). A series of changes were introduced to transition from COMSOAL to Meta-RaPS (DePuy and Whitehouse 2000, DePuy et al. 2000a, DePuy et al. 2000b).

Similar to other metaheuristic methods, Meta-RaPS is also a high-level strategy with construction and improvement phases. The notable feature in Meta-RaPS is the use of randomness during the construction phase DePuy et al. (2005).

During the construction phase, Meta-RaPS explores the search space and generates a set of feasible components that can be accumulated to the current partial solution (solution being built). The component with the best priority value (according to a greedy or look ahead heuristic) is only added to the current partial solution if a randomly generated number is less than or equal a user-defined priority value (p%). Otherwise, other feasible components with priority values within a user-defined restriction window (r%) are considered candidates to be added to the current partial solution. Selecting one of the candidate solutions is done randomly. This approach allows for worse or less than perfect components to be added to the solution currently being built.

At the end of the construction phase, the improvement phase is initiated by comparing the newly constructed solution with the best solution found so far. A user-defined improvement parameter (i%) is used to create a range around the best solution found so far. If the new solution is within i% of the best solution found thus far, then an improvement phase is initiated by performing an improvement heuristic. The goal is to take advantage of a promising newly constructed solution by applying a local search heuristic method in an attempt to find a solution that exceeds the best solution found so far. Figure below shows a pseudo code of Meta-RaPS.

```
Do Until a termination criteria are met
    Construction Phase:
        Do Until feasible solution generated
            Find priority value for each feasible component
            Find component with best priority value
            P = randomly generated value: RND (1,100)
            If P<= p% Then
                New solution = new solution + component with best priority value
            Else
                Form 'available list' of all feasible component whose priority values are within r%
                of best priority value
                Randomly choose component from available list
                New solution = new solution + randomly selected component
            End If
        End Until
    Improvement Phase:
        If New solution within i% of the best solution found so far
            Apply local search method to the new solution
            If improved new solution is better than best solution found so far
                Best solution found so far = improved new solution
            End If
        End If
End Until
```

Figure 8 Pseudo Code for Meta-RaPS (DePuy et al., 2005)

Using Blum and Roli's (2003) categorization effort listed in the previous section, the original Meta-RaPS can be categorized as a metaheuristic that uses a single point search approach with a static objective function, single neighborhood, and no memory.

Meta-RaPS is considered to be a general form of GRASP as Meta-RaPS with p% set to 0%, i% set to 100%, and a greedy heuristic during the construction phase will imitate the behavior of GRASP. Since its introduction, GRASP (Feo & Resende, 1995) has been used to solve a variety of combinatorial optimization problems including routing and scheduling problems (e.g. Villegas et al. (2011) and Goodman et al. (2009)). The relationship between GRASP and Meta-RaPS paves the way for Meta-RaPS to be applied to all problems GRASP has been able to solve. Additionally, Meta-RaPS (not as a general form of GRASP) has been used to solve various combinatorial optimization problems as seen in the next section.

2.3.1 Meta-RaPS Applications

Since the introduction of Meta-RaPS (DePuy & Whitehouse, 2001), several researchers have successfully utilized it in solving combinatorial optimization problems. This section provides a chronological overview of the existing Meta-RaPS applications. As a new metaheuristic, most of the literature published focused on utilizing Meta-RaPS to solve fundamental combinatorial optimization problems.

The effectiveness of Meta-RaPS was first demonstrated by solving the Resource Constrained Project Scheduling Problem (RCPSP). COMSOAL was adapted to solve the RCPSP. Since Meta-RaPS is based on COMSOAL, applying Meta-RaPS to this problem was a natural first choice. To solve RCSP benchmark problems (Patterson Problems), the authors developed various construction heuristics (DePuy & Whitehouse, 2001b). The ACTIM and Summation heuristics outperformed others and thus was selected. Meta-

RaPS (or modified COMSAOL) outperformed the original COMSOAL along with other RCPSP heuristics. This version of Meta-RaPS did not include an improvement phase. The main focus seemed to be on the impact of incorporating both randomness and priority strategies.

The TSP was also solved by Meta-RaPS (DePuy et al., 2005). The Cheapest Insert and Node Insertions were used as the construction phase and improvement phase heuristics respectively. Meta-RaPS outperformed other metaheuristics for problems up to 101 cities, which was very promising. With the incorporation of the improvement phase, Meta-RaPS reached a classic metaheuristic status. Therefore, parameter tuning was critical. The authors followed a basic "parameter search approach" by coming up with ranges for the parameters' values and selecting two benchmark problems to solve. The parameters' values yielding best outcome were utilized.

Lan et al. (2007) tackled the Set Covering Problem (SCP). This problem proved to be more difficult to tackle. For the construction phase, the authors followed the Chvatal's (1979) greedy heuristic. For the improvement phase, a local search heuristic was developed by the authors. However, finding a set of parameter values that resulted in reaching optimal solutions was not possible. This pushed the authors to adjust the framework of Meta-RaPS to produce better quality solutions by revisiting the construction heuristic during the improvement phase, randomizing the selection of priority rules, and penalizing worst solutions. The performance of this version of Meta-RaPS was good when compared to other heuristics. The authors revisited and further analyzed the concept of partial construction (Lan & DePuy, 2006). The authors also introduced the concepts of intra-iteration and inter-iteration randomization, which contributed to better solutions for the SCP.

Moraga et al. (2005) applied Meta-RaPS to solve the 0-1 Multidimensional Knapsack Problem. In similar fashion with previous authors, several construction heuristics were examined. The Dual Greedy Rule outperformed others and thus was utilized in the construction phase. For the improvement phase the insertion and exchanging neighborhood search heuristics were followed. Parameter tuning approach

was the same as the one used by Morage (2002). For small size problems, Meta-RaPS outperformed other metaheuristics, another promising finding. A version of GA outperformed Meta-RaPS when solving large size problems.

Rabadi et al. (2006) utilized Meta-RaPS to solve the unrelated parallel machines scheduling problem (PMSP) with machine-dependent and sequence-dependent setup times to minimize makespan. The construction phase employed the Shorted Adjusted Processing Time with Smallest Load while the improvement phase utilized three neighborhood search heuristics (inter-machine job insertion, inter-machine pair-wise job exchange, and intra-machine random job exchange). Moraga's (2005) parameter tuning approach was followed. For small size problems, Meta-RaPS reached optimal solutions, which promising. For larger size problems, Meta-RaPS outperformed the Partitioning heuristic used to solve the problems.

In 2009, Hepdogan and colleagues (2009) applied Meta-RaPS to solve the Early/Tardy single machine scheduling problem with common due dates and sequence-dependent setup time. The construction heuristic utilized was the shortest adjusted processing time (SAPT). The improvement heuristic was a generalized pairwise exchange. The process of parameter tuning was not discussed in details. Meta-RaPS solved both small and large size problems. The performance was acceptable when compared to Simulated Annealing.

Hancerliogullari et al. (2013) used Meta-RaPS to solve the Aircraft Scheduling Problem (ASP). The problem was modeled as a Parallel Machine Scheduling Problem with Unequal Ready-times, Target times and Deadlines. A real-life ASP also requires sequence-dependent separation times on each runway to prevent the dangers associated with wake-vortex effects. The authors developed and evaluated several heuristics to use the construction phase of Meta-RaPS. In addition to Meta-RaPS, the authors solved the problem with Simulated Annealing (SA) metaheuristic using the same greedy heuristics. For both Meta-RaPS and SA, the parameter tuning was achieved by following a Design of Experiment approach. Both Meta-RaPS and SA outperformed the greedy heuristics.

Though optimal or near optimal solutions were found for many instances, SA outperformed Meta-RaPS.

CHAPTER 3

HYBRIDIZING METAHEURISTICS WITH DATA MINING

The development of the field of metaheuristic algorithm started with focusing of formulating and introducing new metaheuristic methods. This phase lasted for approximately two decades, assuming the start of metaheuristics was in the 1980s. Following the *pure* metaheuristics development phase, the characteristics of many of the well-established, pure metaheuristics became well-defined. This has led to combining metaheuristics with each other and with other existing optimization algorithms. The resulting metaheuristics are known as *Hybrid* Metaheuristics. Successful hybrid metaheuristics found the appropriate balance when combined with other algorithms, such as:

- Combining metaheuristics with other metaheuristics,
- Combining metaheuristics with operational research methods, such as exact mathematical programming methods,
- Combining metaheuristics with artificial intelligence methods, such as constraint programming methods, and
- Combining metaheuristics with machine learning and data mining techniques (Talbi, 2009)

The formulization of the field of hybrid metaheuristics started in the early 2000s (Talbi, 2002). The field is very active especially as more complex problems are being solved using approximate methods. A group of researchers are focused on continuing to define the field and surveying the literature has published several references on the subject (Blum et al., 2010; 2011). Additionally, various members meet at the International Workshop on Hybrid Metaheuristics and the International Conference on Hybrid Metaheuristics to further advance the field.

All the ongoing efforts, has led to findings few patterns:

- Positive:
    - For various real-life combinatorial optimization problems, hybrid metaheuristics were able to find better results (Talbi, 2002)
- Neutral:
    - A formal definition for hybrid metaheuristics has yet to be introduced. However, this is viewed as a normal part of an evolving field (Blum et al., 2011). An early exact definition might lead to limitation in future research
- Negative:
    - Working with hybrid metaheuristics is often a challenge as they require developing expertise in more than one field, and
    - Producing general hybrid metaheuristics is more challenging than producing general metaheuristics as more parameters, rules, and algorithms are involved in hybrid metaheuristics (Blum et al., 2011).

In the following sections, taxonomy for hybrid metaheuristics proposed by Talbi (2009) will be discussed to give a better understanding of how hybridization might occur. Though various hybridization options exist, the goal of this effort is to hybridize Meta-RaPS with a data mining technique. This will be discussed in detail in subsequent sections. The chapter ends with reviewing data mining techniques that will be combined with Meta-RaPS.

3.1 Hybrid Metaheuristics Categorization

Few frameworks of hybrid metaheuristics have been published (Blum et al., 2008; Raidl, 2006). The classification discussed here is by Talbi (2009), which is illustrated in the fire below. This classification helps clarify issues faced when designing hybrid metaheuristics.

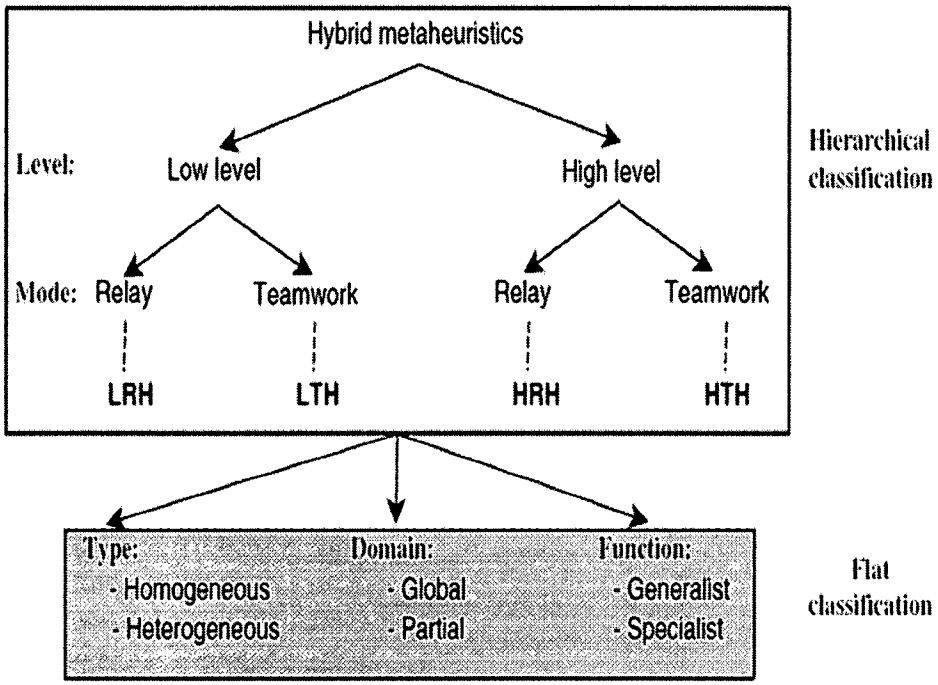


Figure 9 Metaheuristics Design Classification (Talbi, 2009)

The level-wise, hybridization can either be low or high level. Low-level hybridization represents a class where a metaheuristic is embedded in another metaheuristic. High-level hybridization represents a class where different algorithms coordinate with each other without impacting the internal structure of each algorithm. Within each hybridization level, two modes are possible: Relay and Teamwork. In Relay mode, algorithms are executed in a pipeline-like manner, where the output of one is the input for the next algorithm. In Teamwork mode, algorithms coordinate with each other and work in parallel. The levels and modes described yield four classes of hybrid metaheuristics: Low-level Relay Hybrid (LRH), Low-level Teamwork Hybrid (LTH), High-level Relay Hybrid (HRH), and High-level Teamwork Hybrid (HTH).

- In LRH class, algorithms are embedded in single-solution metaheuristics, such as embedding a local search algorithm in the Simulated Annealing metaheuristic.
- In LTH class, single-solution metaheuristic is typically embedded in population-based metaheuristic. This hybridization allows for both diversification (strength of

population-based metaheuristics) and intensification (strength of single-solution metaheuristics). This results in many combinations that involve Simulated Annealing or Tabu Search with Evolutionary Algorithms or Swarm Algorithms.

- In HRH class, metaheuristics are sequentially executed. The characteristics of the selected metaheuristics complement each other. For example: a population-based metaheuristic is typically followed by single-solution metaheuristic. This hybridization allows for diversification and then fine-tuning and intensification.
- In HTH class, several self-contained metaheuristics tackle the search space in parallel. These metaheuristics coordinate with each other to find optima over various regions of the search space.

In addition to four hierarchical hybridization classes, Talbi (2009) provides a flat classification for hybrid metaheuristics. Aspects such as Type, Domain, and Function represent another layer where hybrid metaheuristics vary. Type-wise, a hybrid metaheuristic may consist of a homogenous or a heterogeneous set of algorithms. Though homogeneous metaheuristics are combined, they are expected to have different values for their parameters and/or search components (e.g. heuristics, neighborhood structures, etc.). Another aspect used to classify hybrid metaheuristics is the Domain where the hybrid metaheuristics are applied. In global hybridization, all metaheuristics tackle the entire search space. On the other hand, partial domain hybridization aims at partitioning the search space in advanced and assigning specific metaheuristics to solve specific regions. The last level of hybridization aspect categorizes the functionality performed by the algorithms that make up a hybrid metaheuristic into general and specialist hybrids. In general hybrids, all metaheuristics have and solve the same problem. Specialist hybrids divide the problem into sub-problems and assign each sub-problem to one of the metaheuristics in the hybridized combo.

Both hybridization approaches described here are classified as *HRH* with *heterogeneous* algorithms that are applied to the *global* search domain with a *general* (same) problem to solve.

3.2 Hybridizing Metaheuristics with Data Mining

In the field of Data Mining (or Knowledge Discovery in Databases) computer science and statistics techniques are employed to process large amounts of data and transform the data into knowledge (Chen et al., 1996). The fundamental rules and patterns obtained by mining datasets are: association rules, sequential patterns, classification rules, and data clusters (L. F. Santos, 2005). This information can be viewed as learning. The process of learning can be simplified as the process of utilizing the knowledge produced by these techniques in making future decisions. Data mining techniques are categorized based on the way they learn into: supervised, semi-supervised, and unsupervised learning techniques. The supervised and unsupervised learning methods are most common. Supervised learning algorithms learn from given correct input. On the other hand, unsupervised learning algorithms discover relationships without input. Some of the most known tasks performed by these two techniques are data classification and data association rules correspondingly.

Though metaheuristics have been widely employed to improve the performance of data mining techniques, the opposite is not true. Hybridizing metaheuristics by incorporating data mining techniques have rarely been studied (Talbi, 2009). The potential hybridization between metaheuristics and data mining techniques can be classified based on the following categories:

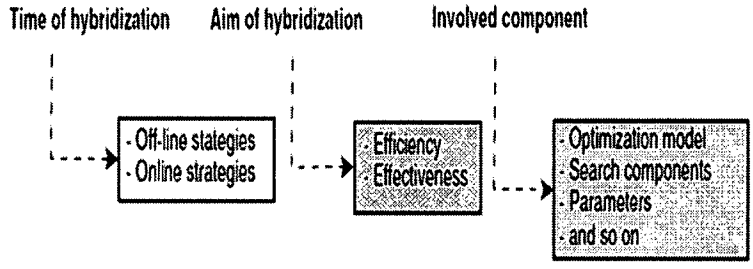


Figure 10 Data Mining Integration Approaches (Talbi, 2009)

- Time of hybridization: data mining techniques can be used extract knowledge and formulate strategies off-line (before the search starts) or on-line (during the search).
- Aim of hybridization: data mining techniques can assist in managing the search space (efficiency) or guiding a metaheuristic algorithm in search space (effectiveness).
- Involved component: data mining techniques can on improving a specific aspect of the metaheuristic, such as solution encoding, initialization, parameter tuning, etc.

The use of data mining can vary from one hybridization class to another. A survey mapping the aforementioned data mining hybridization categories to existing publications can be found in (Jourdan et al., 2006). In this survey, the majority of the metaheuristics hybridized with data mining are population-based metaheuristics. This conclusion is logical due to the expected guidance and intensification the knowledge extracted using data mining techniques can provide to diverse population-based metaheuristics.

As an example, Lessmann et al. (2011) utilized data mining methods to improve the parameter tuning aspect in Particle Swarm Optimization (PSO) metaheuristic. Regression models were used to mine the data generated by particles during typical PSO iterations. The information available about a particle is presented in a particle's signature, which consists of position vector, flying vector velocity, and best position vector. The particle's next/future movement is calculated following an equation that makes use of three parameters: c_1, c_2, and V_{max}. These parameters are expected to impact how a particle searches the search space. These parameters are the target parameters to be tuned. The hypothesis put forward by the authors states that a particle's signature contains relevant information, which will impact the effectiveness of the particle's next move. To confirm this hypothesis, the authors described a model that identifies the appropriate information to extract and a group of regression methods (multiple linear regression model, stepwise multiple linear regression model, least square support vector machine - linear, least

square support vector machine radical, and regression forest) to utilize the extracted information and forecast appropriate values for the PSO parameters. The proposed process was applied to the water supply network planning problem. The results behind the empirical study focused on identifying the best regression method by examining the forecasting accuracy of each method. Both non-linear regression methods (least square support vector machine radical, and regression forest) resulted in better forecasting accuracy than the original (no-regression) and the linear methods. Additionally, the performance of the forecasting methods was analyzed against increasing the size of the problem.

Very few non-population-based metaheuristics were listed. Three GRASP hybridized with data mining were surveyed (Plastino et al., 2011; Ribeiro et al., 2004; L. F. Santos, 2005). GRASP was hybridized with a portion of the unsupervised learning data mining similar to the Apriori Algorithm (Argawal et al., 1993) named Frequent Itemset Mining or FIM. The target combinatorial optimization problems are the Set Packing Problem, Maximum Diversity Problem, and the p-median problem. Learning is done by forgoing the multi-start feature of GRASP and guiding the start of the search based patterns frequently found in elite solutions found after running GRASP for a defined number of iterations.

In addition to being hybridized with GRASP, a customized version of the Apriori Algorithm (Argawal et al., 1993) was also integrated with GA to solve the Oil Collecting VRP (Santos et al., 2006). The high-level objective of the hybridization is to accelerate the occurrences of good solution in the GA population by discovering and incorporating patterns found frequently in elite solutions.

As the hybridization category of interest, the HRH class can benefit from the knowledge provided by data mining in:

- Search components: instead of relying on random multi-start approach, data mining techniques can help guide the search to better regions.

- Parameter setting: data mining methods can help find the optimal parameter settings using knowledge gained during the search
- Optimization model: data mining can assist in decomposing the optimization problem via classification and clustering techniques. Decomposition allows the metaheuristic to be more effective handling more manageable sets of problems.

3.3 Classification via Decision Trees

Classification is learning a function that maps or classifies input data into one of several predefined classes (Fayyad et al., 1996). While classification handles discrete input, regression learning (another supervised learning approach) handles continuous input variables. Popular classification methods are Inductive Decision Trees (IDTs), neural networks, support vector machines.

The IDT is a data mining technique that aims at classifying data by generating classification rules without prior knowledge (Chen et al., 1999). IDT achieves this goal after completing a training phase, during which Training Examples (TE) are used as input. Data mining algorithms can be classified based on the content of TE into: supervised, semi-supervised, and unsupervised. For a supervised learning method such as IDT, the set of TE consists of labeled data, which can be viewed as a paired data: a vector of system attributes and the desired label (see table below). A desired label can be described as the desired outcome of learning or a class (Park et al., 1997).

Instance	Vector of Attributes				Class
	Attribute 1	Attribute 2	...	Attribute n	
1	Xxx	X		xx	Success
2	Xxx	X		xx	Success
3	Xxx	X		xx	Failure
...					...

Table 2 Theoretical View of Labeled TE (Kotsiantis, 2007)

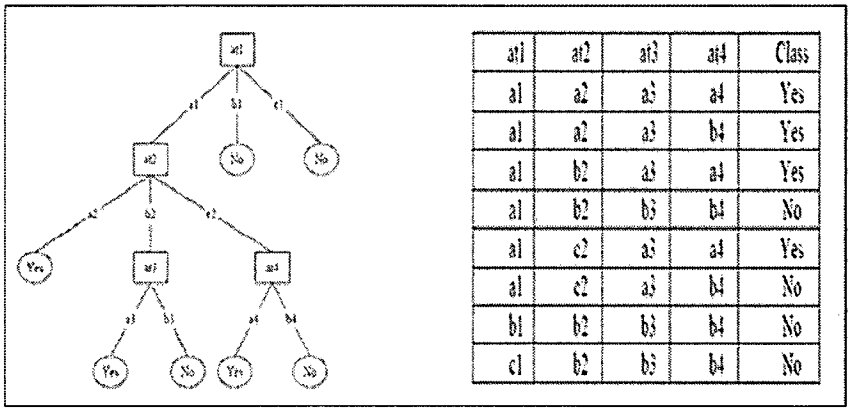
Following the training phase, IDT is expected to have *learned* the concepts of the classes in TE. This is portrayed by generating classification rules. Examining the quality of the classification rules is achieved when IDT is faced with never seen before data. The TE for semi-supervised algorithms contains a mix of labeled and unlabeled data. While unsupervised algorithms utilize unlabeled data during their training phase.

When compared to other supervised machine learning algorithms (Table 3), IDT fared well especially when it comes to its speed, handling of discrete, binary, and continuous attributes, and comprehensibility.

Criteria	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM	Rule Learners
Accuracy in general	**	***	*	**	****	**
Speed of learning with respect to number of attributes and the number of instances	***	*	****	****	*	**
Speed of classification	****	****	****	*	****	****
Tolerance to missing values	***	*	****	*	**	**
Tolerance to irrelevant attributes	***	*	**	**	****	**
Tolerance to redundant attributes	**	**	*	**	***	**
Tolerance to highly interdependent attributes (e.g. parity problems)	**	***	*	*	***	**
Dealing with discrete /binary/continuous attributes	****	*** (not disc.)	*** (not cont.)	*** (not directly disc.)	** (not disc.)	*** (not directly ccont.)
Tolerance to noise	**	**	***	*	**	*
Dealing with danger of overfitting	**	*	***	***	**	**
Attempts for incremental learning	**	***	****	****	**	*
Explanation ability/transparency of knowledge/classifications	****	*	****	**	*	****
Model parameter handling	***	*	****	***	*	***

Table 3 Supervised Learning Algorithms (** Best, * Worst) (Kotsiantis, 2007)**

The clarity IDT enjoys is largely because a decision tree can easily be represented as If-Then decision rules or as graphical trees.



at1	at2	at3	at4	Class
a1	a2	a3	a4	Yes
a1	a2	a3	b4	Yes
a1	b2	a3	a4	Yes
a1	b2	b3	b4	No
a1	c2	a3	a4	Yes
a1	c2	a3	b4	No
b1	b2	b3	b4	No
c1	b2	b3	b4	No

Figure 11 Graphical Representation of IDT with Associate TE (Kotsiantis, 2007)

Employing the concept of an IDT involves making use of decision tree construction algorithms. These algorithms include ID3 and its extension C4.5, which grew in popularity to become the standard decision tree construction algorithms (Quinlan, 1993). Both ID3 and C4.5 start by evaluating the attributes in a TE. An attribute that best divides the data is selected to be the root node of a tree. The branches of each node represent all possible values of an attribute. As a tree branches out, a leaf (end) node is reached. Leaf nodes represent the outcomes of the classification (i.e. the classes to which instances belong).

IDT can have poor predictive performance due to its tendency to overfit the TE. Overfitting ties the learned classification rules to the TE used during the construction and training phases. Overfitting can be prevented by

1- Terminating training prior to fitting all pairs in a TE, or

2- Pruning/reducing the size of tree to be constructed.

Two pruning approaches exist: pre and post pruning. Pre-pruning is a method to stop building a tree at a predetermined point. On the other hand, post-pruning is an approach that evaluates the performance of an IDT as it is being pruned (Kotsiantis, 2007).

3.4 Rules Learning via Association

In addition to learning via classification (supervised learning), learning could also achieved by discovering rules associating different seemingly independent items, identifying frequent patterns, and/or discovering causal relationship between items without any advanced knowledge. This category of learning is known as association rules, which is one the unsupervised learning techniques. An association rule is "an implication of the form $X \rightarrow Y$, where X is the antecedent and Y is the consequent of the rule" (Alpaydin, 2010). Association rules are applied in various fields including mining biological data in bioinformatics databases to product recommendation in websites based on customers' recent purchases. The classical application of association rules is the Basket Analysis, where retailers have an interest in discovering the dependency between two products purchased by customers.

To quantify the relationship between two (or more) items or products, three measures are often calculated: Support, Confident, and Lift.

- *Support* of the association rule $X \rightarrow Y$ or Support (X,Y):

$$\textit{Support}\ (X, Y) \equiv P\ (X,Y) = \frac{\#\text{items containing both X and Y}}{\#\text{items}} \quad (1)$$

- *Confidence* of the association rule $X \rightarrow Y$:

$$\textit{Confidence}\ (X \rightarrow Y) \equiv P\ (Y|X) = \frac{P(X,Y)}{P(X)} = \frac{\#\text{items containing both X and Y}}{\#\text{items containing X}} \quad (2)$$

- *Lift* or Interest of the association rule $X \rightarrow Y$:

$$Lift\,(X \rightarrow Y) \equiv P\,(X|Y) = \frac{P(X,Y)}{P(X)\,P(Y)} = \frac{P\,(X\,|\,Y)}{P(Y)} \tag{3}$$

The *Support* of a rule indicates its statistical significance of a rule. Higher support for a rule is of interest since a rule or a relationship that rarely occurs is not likely to impact the learning process (e.g. only 0.01% of customers bought two products together will not influence decision makers to reconfigure a market to place these items closer to each other). Along with a potential rule with strong *Support*, the *Confidence* is calculated to solidify that a conditional probably exists tying two seemingly independent items. In the case of Basket Analysis, a *Confidence* value close to 1 and higher than P(Y) indicates that the probably of customers purchasing product X and Y is much higher than the probability of customers only purchasing product Y. Threshold values for both *Support* and *Confidence* are usually set by the user in order to discard any rules of less statistical significance and strength. The *Lift* is used to demonstrate the relationship between two items. If *Lift* is less than 1, then the relationship between items X and Y is less likely (purchasing product X makes purchasing product Y less likely). If *Lift* is more than 1, then the relationship between items X and Y is more likely (purchasing product X makes purchasing product Y more likely). If *Lift* is close to 1, then X and Y are independent.

CHAPTER 4

VEHICLE ROUTING PROBLEM

The Vehicle Routing Problem is one of the fundamental combinatorial optimization problems. Though this dissertation focuses on tackling the classic Vehicle Routing Problem using the Meta-RaPS metaheuristic method, this chapter aims at providing a brief background about the problem. The chapter starts by describing the field of combinatorial optimization. Subsequent sections describe the Vehicle Routing Problem, its variations and the various approaches to solving this problem.

4.1 Problem Overview

Dantzig and Ramser (1954) introduced the Truck Dispatching Problem, which is now known as the Vehicle Routing Problem. The problem consists of a central depot, where products are stored, and a fleet of trucks to deliver the products to geographically dispersed customers. The objective is to identify the optimal set of routes each truck is to follow in order to serve all customers. The VRP is a combinatorial problem as it is considered a special version of the TSP if the traveling salesman is constrained by returning to the central depot after visiting each city. Similar to the TSP, the VRP is an NP-hard problem (Toth & Vigo 2002).

The classical VRP, also known as the Capacitated VRP (CVRP), is defined by Laporte (1992) as graph G= (V, A), where

N = the number of customer to be served

V = $\{0,\ldots, n\}$is a set of vertices each representing a customer. Vertex 0 is the central depot

$A = \{(i,j) : i,j \in V, i \neq j\}$ is the set of arcs connecting the customers, where an arc connecting customers i and j is A_{ij}

m = trucks are located at the central depot

n_i = the number of customer to be served by truck j

Q = capacity of each truck

q_i = demand of customer *i*

c_{ij} = cost or distance of traveling arc A_{ij}. The cost is typically symmetrical, where $c_{ij} = c_{ij}$

The objective function identifying the set of routes (taken by each truck) that minimizes the total distance traveled. The constraints applied in the CVRP consist of visiting all customers, not exceeding the capacity of the trucks.

$$\sum_{k=1}^{k=m} \sum_{j=0}^{j=n_j} c_{j,j+1} \tag{4}$$

The figure below shows a CVRP that consists of 14 customers (each with demand shown next to the vertex). Four tours are used to deliver goods. The capacity of each truck is 10.

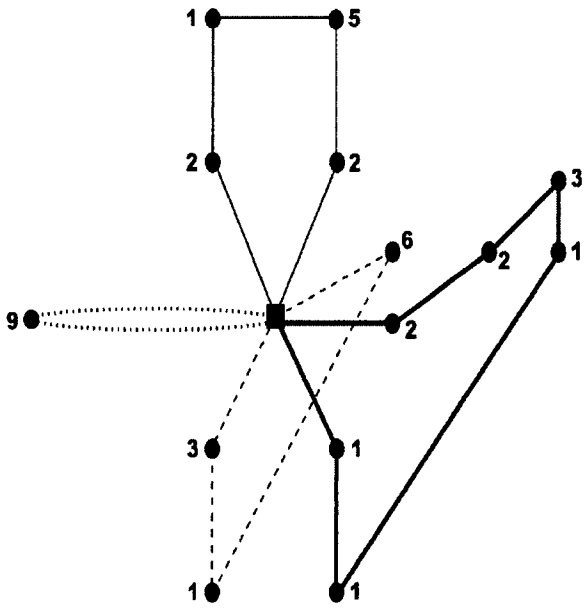


Figure 12 Solution of CVRP 14 Customers and Four Trucks Q 10. (Laporte, 2007)

The CVRP is one of the most studied combinatorial optimization problems. This attention led to many advances and variations to the original CVRP. The more real-life constrains, conditions, and/or assumptions are considered, the more VRP variants get introduced and studied. In an effort to categorize the VRP variants, the variants are divided based on the constraints that led to creating them: customer variants, distribution systems variants, and problem variants.

Customer variants:

In many real-world distribution problems the customers expect receiving demands within a given time window. The VRP with Time Windows (VRPTW) is introduced (Solomon, 1987). The Periodic VRP (PVRP) model and solves a VRP where the costumers require periodic delivery of products (Gaudioso & Paletta, 1992). On the other hand, to model a VRP where the customers' demands are not known in advanced, Psaraftis (1995) introduced the Stochastic VRP (SVRP). This variant covers more than unknown demands, but also unknown number of customers and unknown time windows. Another customer-induced constraint can be based on the product being delivered. The Site Dependent VRP (SDVRP) was introduced to model a VRP where the products being delivered require specific vehicles (Cordeau & Laporte, 2001).

Distribution systems variants:

The CVRP makes many assumptions about the distribution system. Distribution companies can have several fleets of vehicles to deliver products. Assuming that all vehicles have the same capacity is very simplistic. Hence, CVRP with heterogeneous fleets variants was introduced (Tarantilis et al., 2004). To model distribution systems with more than a single central depot, the CVRP with multiple depots was introduced (Filipec et al., 1997). In distribution systems where the truck does not have to return to the back to the central depot, the Open VRP is used (Tarantilis et al., 2005). Another VRP variant is the VRP with multiple scheduling (VRPM). In this instance, the distribution companies allow the vehicles to return back to the central depot(s) and replenish with more products (Tarantilis et al., 2005).

Problem variants:

There are VRP variants can be considered spinoffs of the VRP. The Arc Routing Problem (ARP) satisfies customers demand in a similar fashion as the VRP. However, the ARP delivers/picks up products from the edges or the arcs connecting the customers (Lacomme et al., 2003). Additionally, systems that require serving both customers and arcs are modeled using the General Routing Problem (GRP) (Gerhard & Dirk, 2005).

Though this is not a complete list of all VRP variants, the list aims at illustrating the wide spectrum of VRP variants and the attention this problem it received from researchers. Several VRP variants can fit in more than a single category such as the CVRP with Split Deliveries (CVRPSD) (Belenguer et al., 2000) and the VRP with Pickup and Delivery (VRPPD) (Savelsbergh & Sol, 1995). In the CVRPSD, both distribution companies and customers coordinate satisfying customers' demand via delivering products using more than one vehicle. The VRPPD models a system where the trucks not only delivers goods, but also pick up items from the customers. Additional VRP variants exist as a result of incorporating more aspects of the problem such as a Capacitated Arc Routing Problem (CARP) (Golden & Wong, 1981) or incorporating time window constraints to other VRP variants such as the Split Delivery Vehicle Routing Problem with Time Window (Dror & Trudeau, 1989).

4.2 Vehicle Routing Problem Solutions

As with many combinatorial optimization problems, the VRP was solved using complete/exact algorithms and approximate ones. Many reviews articles are written to give snapshots of the algorithms used to solve the VRP and its variants (Golden et al. 1988; Golden & Stewart 1985; Laporte, 1992, 2007, 2009; Laporte & Norbet 1987; Toth & Vigo 2002). Complete algorithms find optimal solution for the problem by searching the entire search space. Complete algorithms used to solve the VRP include Integer Programming algorithms, Dynamic Programming, Branch-and-Bound (Laporte, 2007). The downside of complete algorithms is the time they take to produce an optimal solution. This limits them to small (Toth & Vigo, 2002), mostly conceptual instances of

the problem. To overcome this limitation, many researchers have hybridized complete algorithms with approximate ones. Though they reliably provide optimal solution to the problems, complete algorithms will not be discussed in detail here. The focus on this brief review is on approximate algorithms used to solve the VRP. The approximate algorithms used to solve the VRP are divided into heuristics and metaheuristics.

4.2.1 Heuristic Algorithms

Blum and Roli (2003) classified heuristic algorithms into constructive and local search or improvement algorithms. The constructive algorithms incrementally build a solution by starting from scratch. On the other hand, improvement heuristic algorithms start with an initial solution and work on improving it by modifying portions of in an effort to find better versions of the solution. The similarity between the VRP and the TSP allows for TSP-specific constructive and improvement heuristics to be used to solve VRP. VRP-specific heuristics include the Clarke and Wright saving heuristics, Sweep and Petal heuristics, and Two-Phase Decomposition Procedure (Laporte, 2007).

4.2.1.1 Clarke and Wright (CW) Saving Heuristic

This algorithm was introduced 1964 to solves the CVRP by merging routes to reduce the total distance traveled (Clarke and Wright, 1964). The initial setup assumes that individual routes exist between the central depot and each customer. Reducing the total distance traveled is quantified using the concept of savings:

$$S_{ij} = d_{0i} + d_{0j} - d_{ij} \geq 0 \qquad \forall arc\ (i, j) \tag{5}$$

The distance between two destinations is referred to as d. Thus, the distance between customer i and the depot is d_{0i}, the distance between customers i and j is d_{ij}. S_{ij} represents the distance saved by directly connecting customers i and j. To use the CW algorithm to solve a CVRP instance, the distances between each customer must be calculated. The distances can be saved in a distance matrix. The distances are then used to create a savings matrix, where equation (2) is used to calculate the savings if an arc were to connect two customers directly. The savings matrix is then sorted in a descending

order, where the arc with the most savings is first. The outcome of sorting will be referred to as the sorted savings list. Every arc in the sorted savings list represents a possibility to merge two CVRP routes by connecting two customers together. However, to merge two routes using an arc from the sorted savings list, the arc is examined against three CW conditions. The first condition checks if the customers in the arc belong to two separate existing routes. The second condition verifies that the truck capacity is not exceeded if two routes are merged using the arc under consideration. The last condition checks if the both customers in the arc are either first or last in their existing routes. After evaluating every arc in the sorted savings list, several of the initial routes are expected to be merged.

This heuristic is both simple and very popular when constructing VRP solutions. Some of its main disadvantages are related to the relatively long time the sorting step consumes (Laporte, 2007) and the lack of flexibility evident by the poor results achieved when adding new constraints (Cordeau et al., 2002).

4.2.1.2 Sweep and Petal Heuristics

For VRP layouts where the routes do not interest (or planar graph VRP), the Sweep algorithm is used (Gillett & Miller, 1974). A route is generated starting with a line centered at the depot. Customers are included in a vehicle route gradually by rotating the line until constrains are violated (e.g. capacity is exceeded). The process repeats until all customers are served. The figure below shows the construction of feasible routes using the sweep algorithm with vehicle capacity Q = 10. Customers' demands are displayed on the vertices.

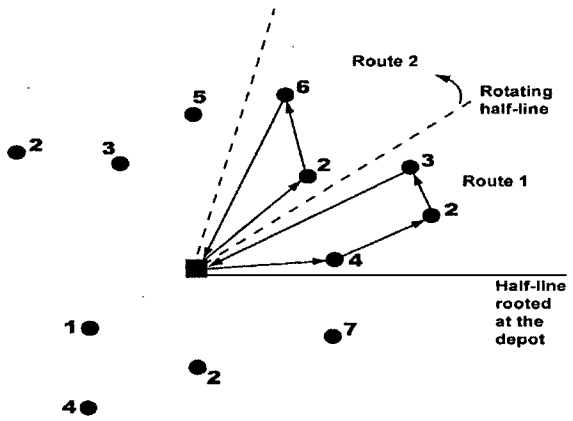


Figure 13 Sweep Algorithm (Laporte, 2007)

The algorithm is very simple, however it is limited to simple VRP instances especially ones where intersections and grids are expected.

The Petals heuristic algorithm (Foster & Ryan, 1976) generates routes based on the Sweep heuristic. A number of routes (or r-petals) are created within a specific area from the depot. A subset of the routes is selected and then a Set Partitioning Problem (a complete algorithm) is performed. Several extensions were introduced to the petals heuristic to allow for intersecting and embedded routes (Renaud et al., 1996; Ryan et al., 1993).

4.2.1.3 Two-Phase Decomposition Procedure

Fisher and Jaikumar (1981) proposed a procedure where the first phase consists of clustering customers and then vehicle routes are created in the second phase. The clusters are created using the General Assignment Problem (an NP-hard combinatorial optimization problem). A seed is located in a region and then the sum of distances between customers and the seed to which they are allocated is minimized. After creating the clusters, a vehicle route is generated by solving a TSP per cluster. Laporte (2007) reported that the Fisher and Jaikumar did not explicitly specify the method behind

placing the seeds in certain regions. Advances to this framework were done to improve seed selection, where Bramel and Simichi-Levi (1995) used actual customers as seeds.

4.2.2 Metaheuristics Algorithms

The VRP and its variants are very close to real-world distribution problems. With their ability to provide good-enough solutions in a short time, metaheuristics are a viable option to solve these problems as evident by the amount of literature produced. The objective of this section is to discuss the core concepts of some of the well-established metaheuristics used to solve the CVRP as reviewed by Laporte (2007). Due to its similarity with Meta-RaPS, the subsequent section focuses on showcasing literature that applied GRASP metaheuristic to solve the CVRP.

According to Laporte (2007), the best metaheuristics approaches used to solve benchmark CVRP instances deviate from best known value by a maximum of 1%. These approaches utilize population-based metaheuristics, local search, or a combination of the two. Table 4 highlights some of the latest best performing metaheuristics used to solve CVRP. The common test beds used by many researchers are:

1. CMT small VRP instances generated by Christofides, Mingozzi, and Toth (1979), and
2. GWKC large VRP instances generated by Golden, Wasil, Kelly, and Chao (1998)

The performance is typically measured by the deviation from best known solutions for the input benchmark problems.

Metaheuristic	Hybridization Approach
TABUROUTE (Gendreau et al., 1994)	Tabu search
AGES (Mester and Bräysy, 2005)	Genetic Algorithm, Guided Local Search
D-Ants (Reimann et al., 2004)	Ant Systems
BoneRoute (Tarantilis and Kiranoudis, 2002)	Tabu search, Adaptive memory
SEPAS (Tarantilis, 2005)	Tabu search, Adaptive memory
GRELS (Prins, 2009)	GRASP, ELS
MPNS-GRAPS (Marinakis, 2012)	GRASP, Multiple reactive methods
GRASPVRP (Layeb, 2013)	GRASP, SA
GRASP w/ evolutionary path-relinking (Usberti, et al., 2013)	GRASP, Multiple reactive methods

Table 4 Sample of Best Performing Metaheuristics to Solve CVRP

4.2.2.1 TABUROUTE

Various Tabu Search approaches were successfully utilized to solve the VRP. Taillard (1993) proposed a two-phased decomposition-like procedure to assist parallel iterative search methods applied to the VRP. The approach divided the problem into several regions. Tabu Search is then applied to each region individually. Inter-region moves are performed periodically for adjacent regions. Another successful Tabu Search approach was introduced by Gendreau et al. (1994) named TABUROUTE. This allows for infeasible solutions to be accepted, which is a result of incorporating penalties terms in the objective function. One of the factors that contributed to the success of TABUROUTE is the use of GENI (GENeralized Isertion) heuristic, which only allows for a customer to be inserted in a route that contains the customer's closest neighbors. GENI helped produce better routes and periodically perturbed solutions.

4.2.2.2 AGES

The Active-Guided Evolution Strategies or AGES (Mester & Bräysy, 2005) is a hybrid metaheuristic that employs basic heuristics, guided local search concepts, and genetic algorithm concepts to solve large CVRP. AGES consists of two phases, where a set of solutions are created in the first phase that get improved in the second phase. To create the initial solutions, cheapest insertion heuristics along with a filling procedure are used. The best solution is passed to the second phase, which starts with a guided local

search method. The goal behind this is to better improve the solution by using 3-5 different improvement heuristics. The second phase is concluded when the guided local search metaheuristic stops improving the solution. This kicks off the last stage during which the solution is further improved by utilizing evolution strategies, which consists of a series of removal and reinsertion moves.

4.2.2.3 D-ANT

Decomposition-ANT (Reimann et al., 2004) focused on using the natural clustering process that human dispatchers followed when handling the VRP. This clustering can occur based on geographical characteristics of an area or a basic zip-code, etc.

D-ANT starts with generating a master solution and improving it. This solution is divided into clusters after determining the center of gravity for each cluster of customers using the sweeping algorithm. Each cluster is then solved using the Ant Colony Algorithm. The construction heuristic used is the C&W heuristic, where the saving rule represents the ants' pheromone trail. After solving each cluster, a master solution is resembled.

4.2.2.4 BoneRoute

This is a population-based method that follows the adaptive memory concepts introduced by Taillard et al. (2001). The method builds solutions based on using sequences of nodes (or bones) extracted from previous solutions. The core concept is to identify frequently occurring sequence of nodes that don't only occur in high quality solutions, but also medium and low quality solutions. The method is organized based into three phases: pool generation phase based on a stochastic Paessens heuristic and Tabu search; pool exploitation phase, where two conditions are applied to identify the bones (user defined bone length and user defined bone frequency of occurring in the pool); and new solution generations phase that uses knowledge extracted from the second phase to construct a new solution.

4.2.2.5 SEPAS

One of BoneRoute's authors proposed Solutions' Elite PArts Search (SEPAS) metaheuristic (Tarantilis, 2005). SEPAS is an iterative method that makes used of Tabu search under the Adaptive Memory Programming concept by Taillard et al. (2001). SEPAS starts by following a systematic, diversified approach while generating initial solutions. These solutions are stored in adaptive memory and are used as the source of the 'elite parts'. 'Elite parts' are a sequence of the generated solutions. These sequences consist of a number of predefined nodes in a single route that occurs in a predefined number of solutions. The 'elite parts' are merged using a construction heuristic. The last phase of SEPAS improves the constructed solution by using a Tabu search approach.

4.2.2.6 GRELS

Prins (2009) hybridized GRASP with Evolutionary Local Search (ELS) metaheuristic named GRELS. Throughout the iterations, GRELS alternates the structure of the solution being constructed between a VRP solution structure and a giant tour. In handling the VRP as a giant tour, the author encoded VRP into TSP problem. A giant tour follows the route-first cluster-second approach.

In the initial GRASP iteration, the construction phase uses the Clarke and Wright heuristic to build a VRP solution. Subsequent GRASP iterations construct a giant tour using the Randomized Nearest Neighbor Heuristic, which gets converted to a VRP solution using a Split procedure. The improvement phase of GRASP is achieved using a fast local search procedure. The improved VRP solution gets converted back to a giant tour before kicking off ELS portion of GRELS, which represent the core of GRELS.

Within ELS, the giant tour gets mutated and split to VRP tours. The VRP tours go through local search (improvement phase). The best solution gets concatenated back to a giant tour, which is reused within ELS again. This process repeats until an upper iteration limit is reached, which happens by incrementing a variable every time ELS mutation does not produce a better solution. The variable is cleared once a new best solution is found. This algorithm was used to solve the CMT benchmark problems. On average, it was better than other metaheuristic methods (which in term on average reached 0.3% deviation from best known solutions). GRELS was comparable to AGES Fast (Mester & Bräysy, 2005).

4.2.2.7 MPNS-GRASP

The MPNS-GRASP (Marinakis, 2012) incorporates several features that allow GRASP to be more effective when solving the CVRP. The algorithm is divided into three phases: initialization, main algorithm, and main phase 2. Parameter setting and selection of both greedy and local search heuristics occur during the initialization phase. During the main algorithm phase, the solution gets progressively constructed by selecting an element from the Restricted Candidate list to use one of the greedy heuristics, which will be selected at random. The performance of the greedy heuristic is evaluated during main algorithm phase while the solution is constructed. If the quality of the currently constructed solution is not better than a threshold, then another greedy heuristic is used. The main algorithm is executed repeatedly until a stopping condition is met. Main phase 2 represent the improvement phase where a new metaheuristic named Expanding Neighborhood Search is introduced and used.

The algorithm was used to solve CMT and GWKC benchmark problems. The average deviation from optimal is 0.41% for the CMT problems and 1.34% for the larger GWKC problems. The various concepts introduced can be quantified as: (1) using more than a single construction heuristic to create an initial solution; (3) allowing construction and/or local search heuristics to be changed online;(2) existing based on the quality of the solution found (and not when a constant number of iterations are executed); (3) following Cardinality-based RCL, where best candidate make it to the RCL and one of the top candidates is selected randomly; (4) proposing and using the Expanding Neighborhood Search.

4.2.2.8 GRASPVRP

The GRASPVRP algorithm is a hybrid between two metaheuristics: GRASP and SA during the construction and improvement phases respectively. The construction approach follows the route-first cluster-second strategy, where they construct a mater/giant route first and then split it into feasible routes. The improvement phase uses the SA metaheuristic with several inter-route and intra-route heuristics.

To construct a tour, the authors introduce construction heuristic that is based on the density order heuristic used for the knapsack problem. This heuristic calculates a density value for the paths between customers. Density value is calculated by taking the difference between the vehicle capacity and customers' demands and then dividing the result by the distance between two customers. Density values are used to create a density matrix, which is used as a method to prioritize customers during the construction of the giant route. The giant route is then split into feasible routes based on the truck capacity. This is repeated until all customers are assigned to a truck. Prior to exiting the construction phase the nearest neighbor heuristic is performed on each tour. GRASPVRP is tested by solving the CMT problems. The performance was comparable to MPNS-GRASP (Marinakis, 2012), but it was outperformed by another GRASP method named GRELS (Prins, 2009).

4.2.2.9 GRASP with Path Relinking

The authors employed several reactive concepts with GRASP to solve the CARP problem (Usberti et al., 2013). These concepts are: (a) reactive parameter tuning, where the value of certain parameters are updated based on performance; (b) using a statistical filter to avoid executing local search heuristics for poor solutions;(c) infeasible local search, which is used to explore the neighborhood of infeasible, high-quality solutions. Additionally, the augmented GRASP is hybridized by incorporating evolutionary Path Relinking to it. This allows elite solutions to be improved by relinking pairs of elite solutions.

CHAPTER 5

META-RAPS FOR CAPACITATED VEHICLE ROUTING PROBLEM

This chapter describes the details behind using the original Meta-RaPS algorithm to solve the CVRP. This algorithm is used as a baseline for the hybrid algorithms. The memory-less Meta-RaPS framework consisting of a construction and an improvement phase is used. The construction heuristic selected is the Clarke and Wright (CW) savings heuristic (Clarke & Wright, 1964). The improvement heuristic selected is a basic local search heuristic named adjacent pair-wise exchange local search heuristic.

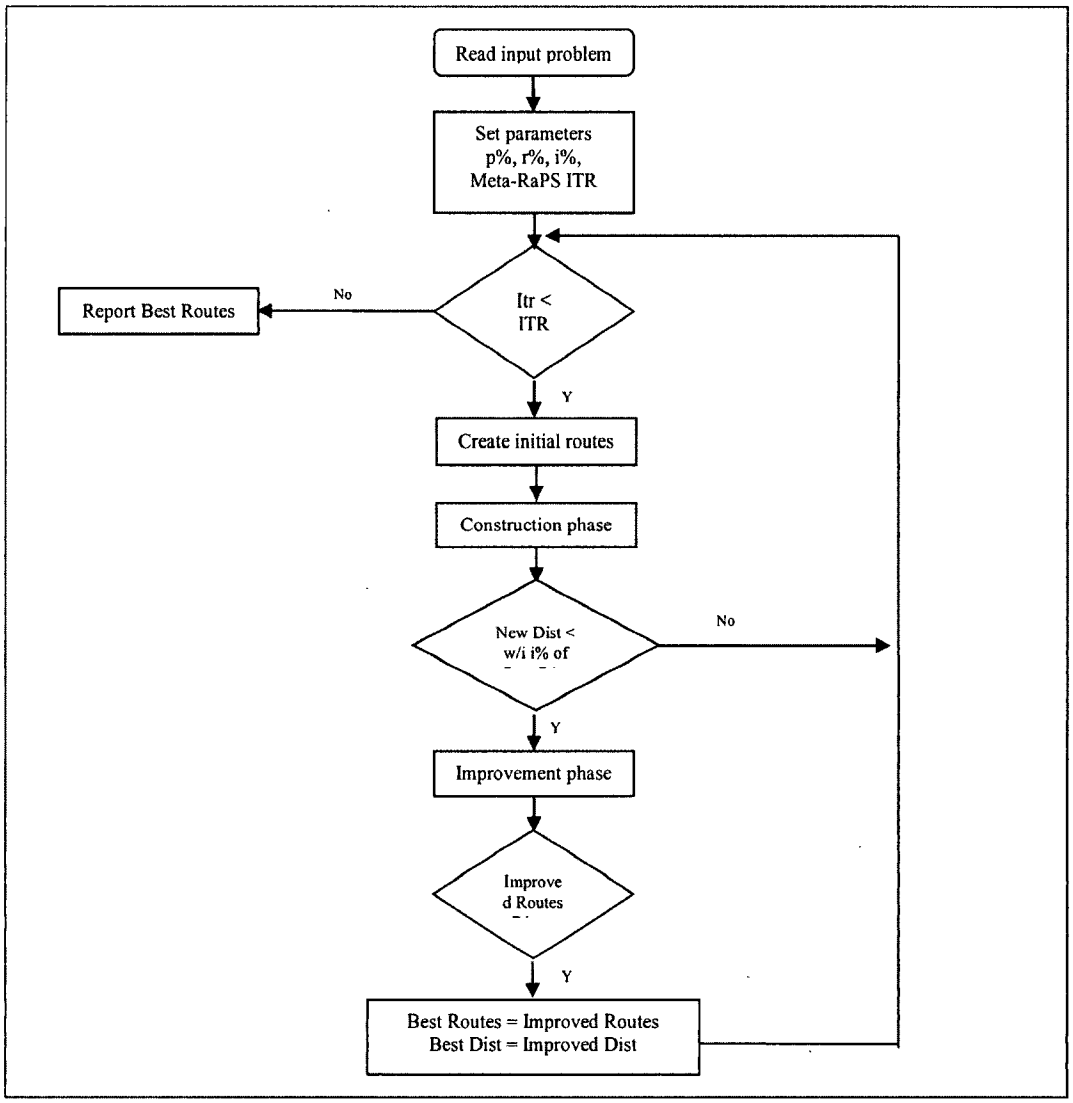


Figure 14 Original Meta-RaPS Framework

The steps executed can be summarized as follows:

1- Read input problem:

Benchmark input problems usually contain the size of the problem (in terms of number of customers), maximum load allowed per truck, (X,Y) coordinates for the central depot and the customers, and the demand associated with each customer. In CVRP with time constraint, the maximum time constraint is provided along with the service (or drop) time

2- Set Meta-RaPS parameters:

p%, *r*%, *i*%, and the number of Meta-RaPS iterations. The process behind determining the values to assign to the parameters is discussed in the design of experiment in later sections.

3- Create initial routes:

Initial routes consist of routes that serve a single customer. In other words, a route can be represented as (central depot, customer A, central depot). The number of initial routes is equal to the number of customers in the input problem

4- Determine the priority:

In this case, the distance is the priority as the overall objective is to minimize the total distance (the distance traveled by each truck). Based on this priority, the distance between each customer in the input problem is calculated.

5- Execute the construction phase:

If a random value is less than or equal to *p*%, then follow the CW heuristic in selecting the next customer to add to the solution. Else, follow a modified CW heuristic. Both options are discussed in details in this section. The construction phase ends when a solution is created. A solution consists of routes, where each route is handled by a truck. The number of customers served in a route depends on the capacity of the truck. A route could consist of a single customer if the customer's demand is equal to the truck's capacity.

6- Execute the improvement phase:

If the total distance of the constructed solution is within i% of the best constructed solution, then apply the local search improvement heuristic (details of the adjacent pair-wise exchange are discussed later in this section). The improvement limit is calculated by considering the values of the Best Constructed Solution (BCS) and Worst Constructed Solution (WCS) so far:

$$Improvement\ limit = WCS + i\%\ (BCS - WCS) \quad (6)$$

7- All the steps are repeated until all Meta-RaPS iterations are exhausted. The best solution found from all Meta-RaPS iterations is reported as the final solution.

5.1 Clarke and Wright Construction Heuristic

Within the construction phase, original and modified versions of the CW heuristic are executed. The following pseudo code describes the construction phase:

```
Do Until feasible solution generated
        Find priority value for each feasible component based on CW (Savings List)
        P = randomly generated value: RND (1,100)
        If P ≤ p% Then
                CW:
                Find component with best priority value based on CW
                New solution = New solution + component with best priority value
        Else
                Modified CW:
                Create an 'available list' of all feasible components with priority values within r%
                of component with best priority value
                Randomly choose component from available list
                New solution = New solution + randomly selected component
        End If
End Until
```

Figure 15 Meta-RaPS Construction Phase Detailed Pseudo Code

The steps followed within the construction phase are dependent on the construction heuristic used. In order to use the CW heuristic within the construction phase, the following building blocks must be understood:

1- Creating an ordered savings list:

In this step, the distances calculated between each customer are used to calculate the distance saved if an arc were to connect two customers directly. The CW saving equation (2) is used to calculate the savings arcs. The saving arcs are sorted from a descending order and stored in an ordered saving list. The first saving arc in the list is the arc that yields the most saving in distance, which makes it the arc with the highest priority.

2- Original CW:

The original CW heuristic could be used to select the next component (or arc) to add to the solution. If the random number $\leq$ p%, then the next arc to add to the solution must be the best, feasible arc. The feasibility of an arc depends on the CW conditions and the current state of the solution. The following are the possible outcomes for an *arc* (i,j) connecting two customers i and j:

1- If both customers belong to different routes ***and*** each customer is either first or last in its route ***and*** the truck capacity is not violated by merging the capacity of both route, then merge the two routes.

2- If one of the customers (e.g. customer i) belongs to an existing route ***and*** this customer is either first or last in its route ***and*** the truck capacity is not violated by adding the capacity of the other customer in the arc (e.g. customer j), then add customer j to the route.

3- If none of the conditions above is applicable, then neither customer in the arc belongs to an existing route. Therefore, a new route is created with both arc customers if the truck capacity is not violated.

If an arc passes the CW conditions, then it is removed from the ordered savings list. If an arc does not pass the CW conditions, then the next arc in the

ordered saving list is evaluated. This is repeated until an arc is feasible arc is found marking this step complete.

3- Modified CW:

If the random number is greater than p%, then instead of selecting the best feasible savings arc, a saving arc is randomly selected from an 'Available List'. The Available List contains savings arcs that are within restriction limit (r%) of the best available saving arc (the arc with the highest priority). To compute the restriction limit, the Most Saving Arc (MSA) and Least Saving Arc (LSA) are identified and the following equation is used:

$$Restriction\ limit = MSA + r\%\ (LSA - MSA) \quad (7)$$

The saving arc randomly selected from the 'Available List' is evaluated against the CW conditions (same condition as in the original CW heuristic). If an arc does not pass the CW conditions, then another arc from the 'Available List' is selected randomly.

The process above (original and modified CW) is repeated until all saving arcs are used to merge routes or create new routes.

5.2. Adjacent Pair-wise Exchange Improvement Heuristic

Following the construction phase, the improvement phase starts. If the constructed solution passes the improvement limit, then a local search heuristic is used to improve the solution. The heuristic selected is the adjacent pair-wise exchange heuristic, which was used for the CVRP by Mirza (2011).

```
For each route in constructed solution
        Old route distance = route distance
        For each adjacent customers in a route
                Replace customer i and customer i+1
                Calculate the route distance
                If new route distance < old route distance
                        Keep new route
                Old route distance = new route distance
        Next
Next
```

Figure 16 Adjacent Pair-Wise Exchange Improvement Heuristic

CHAPTER 6

HYBRIDIZING META-RAPS WITH INDUCTIVE DECISION TREE

The high-level goal behind hybridizing Meta-RaPS with an Inductive Decision Tree (IDT), a data mining technique, is to increase the effectiveness of the original Meta-RaPS by performing online parameter control (figure below). Characteristics (or attributes) about the solutions generated by Meta-RaPS represent the data to be mined by the IDT. After mining Meta-RaPS solutions, IDT is expected to identify Meta-RaPS parameter that was associated with Meta-RaPS' good solutions. Knowing the parameter (and its associated value) represent the knowledge that the IDT obtained. This knowledge is passed to future Meta-RaPS iterations. On the other hand, the values for the remaining Meta-RaPS parameters can be randomly selected.

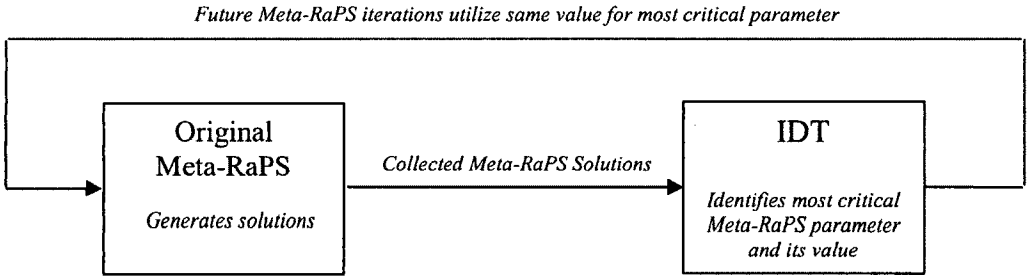


Figure 17 High-level Hybridization Approach

This approach of integrating a data mining technique with a metaheuristic can be classified as: High-Level Relay Hybrid (*HRH*), with *heterogeneous* algorithms (Meta-RaPS and IDT), *global* search domain to tackle, and *general* (same problem) to solve (Talbi, 2009). This hybridization with data mining can be classified as an online

hybridization with the goal of increasing the effectiveness of Meta-RaPS by setting its parameters using knowledge identified in previous iterations.

6.1 Meta-RaPS Inductive Decision Tree

The hybridized Meta-RaPS with Inductive Decision Tree, or MR IDT, starts with an initialization phase. This phase lasts for a predefined percentage of number of Meta-RaPS iterations (TE%). During the TE% iterations Training Examples (TE) are collected to be used in building the tree. During the initialization phase, Meta-RaPS uses no prior knowledge and performs as a multi-start metaheuristic (the same way the original Meta-RaPS). After every iteration in the initialization phase, a set of information or attributes about the solutions produced by Meta-RaPS is gathered. The attributes of interest are ranges of Meta-RaPS parameters p% value, r% value, i% value, and the category (or label) of the solution.

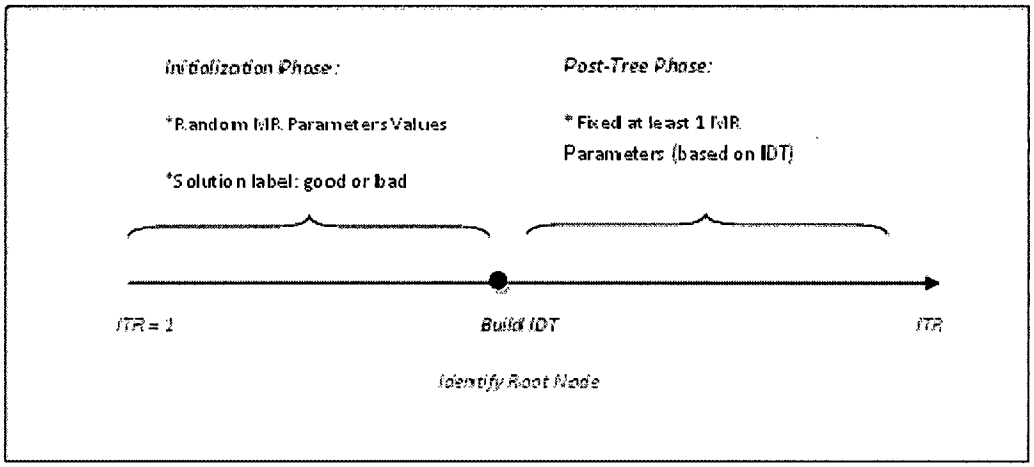


Figure 18 MR IDT Timeline

As previously mentioned, the attributes selected in this design are Meta-RaPS' parameters. To construct a tree, values for these parameters must be assigned. Due to the continuous nature of the parameters' values, categories or levels are assigned. These

levels reference the parameter values used in the original Meta-RaPS. Therefore, during the initialization phase of MR IDT, the values of p%, r%, and i% are randomly selected. For example, if a randomly selected value for p% is larger than the Meta-RaPS value for p%, then the level for p% attribute is set to 'High' for this solution. Otherwise, the level for p% is set to 'Low'. Note that several approaches can be taken to labeling parameters. A different approach can limit labeling a parameter as 'High' only when the randomly selected value falls within a specific range of Meta-RaPS' high values. This implies defining high and low values for Meta-RaPS parameters and defining a labeling range (e.g. a parameter is labeled as 'High' if it is within 5% of Meta-RaPS high p%).

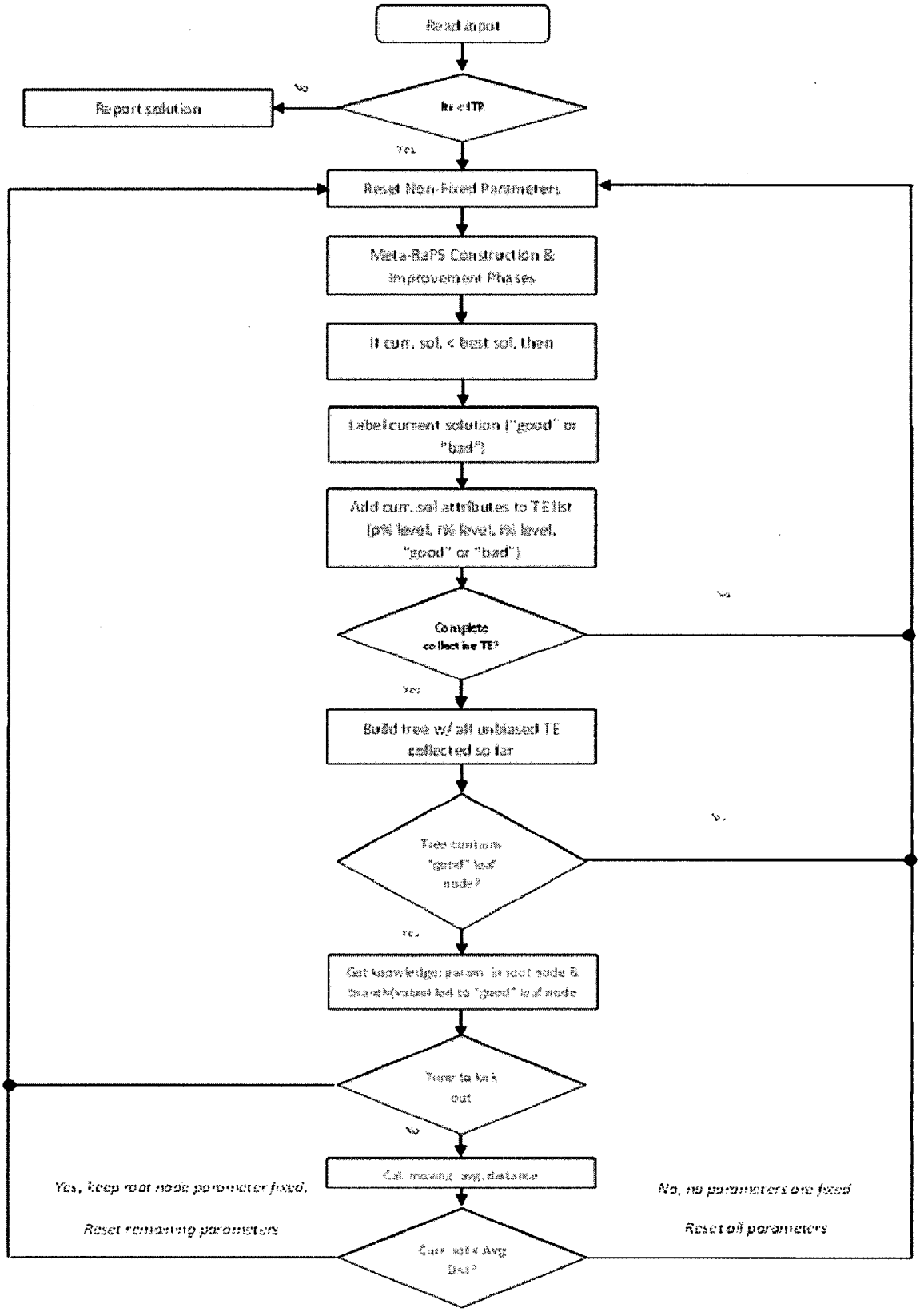


Figure 19 MR IDT Framework

The last attribute captured represents a label given to the solution produced. Several approaches can be used to label a solution as 'Good' or 'Bad'. A solution can be labeled as a 'Good' if its distance is within a percentage of the Best Known Value for the problem being solved (BKV%). A solution with a distance that is exceeds the BKV% limit is labeled 'Bad'. Alternatively, a solution can be labeled as 'Good' if it is better than the moving average value of a predetermined number of previous solutions.

Following the Initialization Phase (after TE% Meta-RaPS iterations have been executed), the Training Examples (or TE) are passed to the IDT algorithm (Table 5). The TE set is used as input for the tree construction algorithm.

Count	p% Level	r% Level	i% Level	Solution Label
1	High	Low	High	"good"
2	High	High	High	"good"
3	Low	Low	Low	"bad"
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
TE	Low	Low	High	"bad"

Table 5 Example of TE List

The IDT algorithm processes the list of TE and produces a tree. The root node of the tree can either be p%, r%, or i%. While the leaf nodes present a solution label: 'Good' or 'Bad'. Though the tree will show various parameters and link them to the leaf nodes (both 'Good' and 'Bad'), the parameters of interest are the ones in the branch that leads to a leaf node labeled 'Good'. This relationship represents the parameter (root node) that is strongly associated with 'Good' leaf node. The knowledge represented by this

relationship is passed to future Meta-RaPS iteration by fixing the value of the root node parameter. Note that it is possible that the TE contain mostly 'Bad' solutions leading the IDT algorithm to build a tree without a 'Good' leaf node. In this case, the no knowledge is passed to future Meta-RaPS iterations.

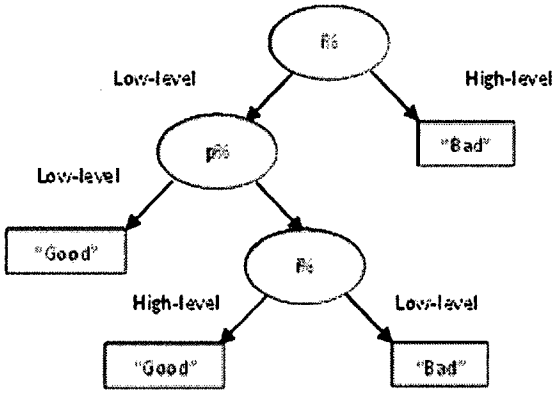


Figure 20 Example of Constructed IDT

Following the construction of the tree, two approaches can be followed: continue to use IDT knowledge until the end of Meta-RaPS iterations or routinely re-evaluate MR IDT performance and implement a kick-out mechanism. In the later approach, after a pre-defined number of iterations (avg%), the performance of MR IDT is evaluated in an effort to prevent continuing to use bad IDT knowledge. If the last MR IDT solution is improving relative to the performance of the moving average of the last avg% solutions, then continue to use the knowledge. Else, stop using the knowledge and revert back to randomly selecting values for Meta-RaPS' parameters. After executing the the kick-out mechanism, more TE are collected. This avoids collecting biased TE (when certain parameters are fixed). The IDT algorithm gets executed again after avg% iterations. A new tree is built leading to potentially discovering new parameters that contributed to good Meta-RaPS solutions.

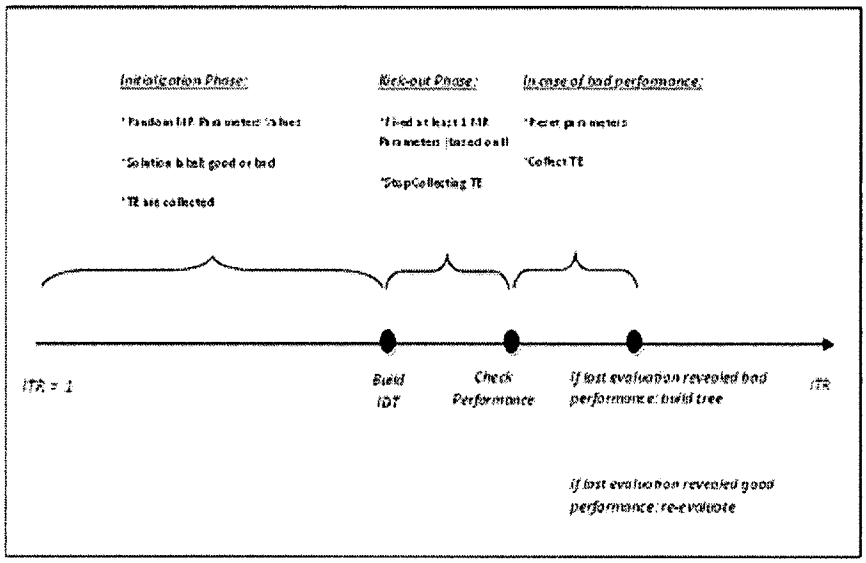


Figure 21 MR IDT Timeline with Kickout

6.2 ID3 Algorithm

The algorithm employed to construct an IDT is the ID3 (Quinlan, 1993). The input to the algorithm consist the Training Examples (TE). Each TE consists of a vector of attributes and a class (a.k.a. as label or target attribute).

A simple, non-Meta-RaPS-related example for using ID3 is mining data with the goal of identifying a consumer base. Each instance in the TE represents a consumer. The attributes gathered are: gender, age, income level, education, and marital status. The target attribute is "belong to consumer base" or "does not belong to consumer base". Given the values associated with these attributes in the TE, ID3 aims at identifying the attribute that leads to correctly classifying the consumer base, which can be income level as an example. If the incorrect attribute is identified (e.g. gender instead of income level), then an organization might dedicate resources to attract the wrong subset of the consumers. Identifying the "income level" attribute is viewed as identifying the most informative attribute about the target attribute.

The TE set in the MR IDT algorithm consist of Meta-RaPS attributes (p%, r%, and i%) and an associated target attribute that describes the solution as either 'Good' or 'Bad'. ID3 aims at identifying the parameter that leads to 'Good' performance.

ID3 recursively processes the TE with the goal of identifying a root node for the tree. The algorithm starts with calculating the Entropy (8) for the training examples.

$$H(P) = -\sum_{i=1}^{n} p_i \log(p_i)$$, where p_i represents the probably of each value of the target attribute (8)

Entropy represents the uncertainty (Shannon, 1948) in arriving at a classification given the TE. In other words, if all the instances in the TE were labeled as 'Good', then Entropy would be 1 and there would be no uncertainty.

After calculating the Entropy, ID3 calculates the information gain (9) for each attribute.

Information Gain (Att, TE) = Entropy (TE) - Entropy (Att, TE) (9)

The Entropy of a specific attribute is calculated using a similar entropy equation (10)

$$H(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} H(T_i) \quad (10)$$

Information gain is used to rank attributes. The attribute with the highest information gain value is the attribute that can classify the TE (and reach a specific target attribute) with the least uncertainty. This attribute correspond to the root node in a tree. The arcs of this root node represent the attribute possible values as found in TE.

The algorithm is repeated for the remaining attributes (while excluding the root node attribute from TE). The attributes identifies subsequently represent non-leaf nodes

in the tree. The algorithm reaches a leaf node when all attributes have been processed. Leaf nodes represent the various values of the target attributes (in this case: 'Good' or 'Bad'). Given the arcs connecting the root node, non-leaf nodes, and the leaf nodes, one should arrive at a leaf node that represents the correct target attribute.

CHAPTER 7

HYBRIDIZING META-RAPS WITH FREQUENT ITEMSET MINING

The high-level goal behind integrating Meta-RaPS with Apriori Algorithm (Argawal et al., 1993) is to increase the effectiveness of the original Meta-RaPS by making use of previous knowledge. To achieve this goal, frequent itemsets found in previous Meta-RaPS solutions are identified and utilized as starting point for future Meta-RaPS iterations.

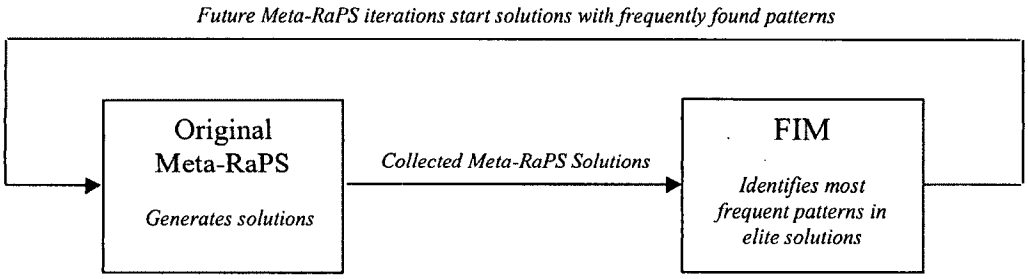


Figure 22 High-level Hybridization Approach

This approach of integrating a data mining technique with a metaheuristic can be classified as: High-Level Relay Hybrid (*HRH*), with *heterogeneous* algorithms (Meta-RaPS and the Frequent Itemsets Mining portion of the Apriori algorithm, *global* search domain to tackle, and *general* (same problem) to solve (Talbi, 2009). This hybridization with data mining can be classified as an online hybridization with the goal of increasing the effectiveness of Meta-RaPS by starting the construction of new solutions using knowledge identified in previous iterations. The hybridization is modeled after GRASP

hybridization with FIM (Plastino et al., 2011) and GA hybridization with Apriori Algorithm (Santos et al., 2006).

7.1 Meta-RaPS Frequent Itemset Mining

The framework of the Meta-RaPS with Frequent Itemset Mining (FIM) is illustrated in the following flowchart. Meta-RaPS FIM starts with an elite list initialization phase. During this phase, solutions that pass elite list eligibility conditions are accumulated until a predefined elite list size is reached (E%). During the initialization phase, Meta-RaPS uses no prior knowledge and performs as a multi-start metaheuristic (the same way the original Meta-RaPS). At the end of each iteration during the initialization phase, the solution generated is saved if it is better than the worst solution in the elite list.

Once the elite list size is reached, FIM is executed. The statistically significant patterns found in the elite solutions are produced. These patterns are identified if they pass Support% (S%) threshold. If no patterns are identified, then the next Meta-RaPS iterations are executed without previous knowledge (same as the original Meta-RaPS). If patterns are found by FIM, then future Meta-RaPS iterations start constructing new solutions with the patterns. The improvement phase is executed normally (no changes from the original Meta-RaPS). The same patterns are used until FIM is executed again. The next FIM execution time is based on the changes to the elite list. When a new item is added to the elite list, FIM is executed again.

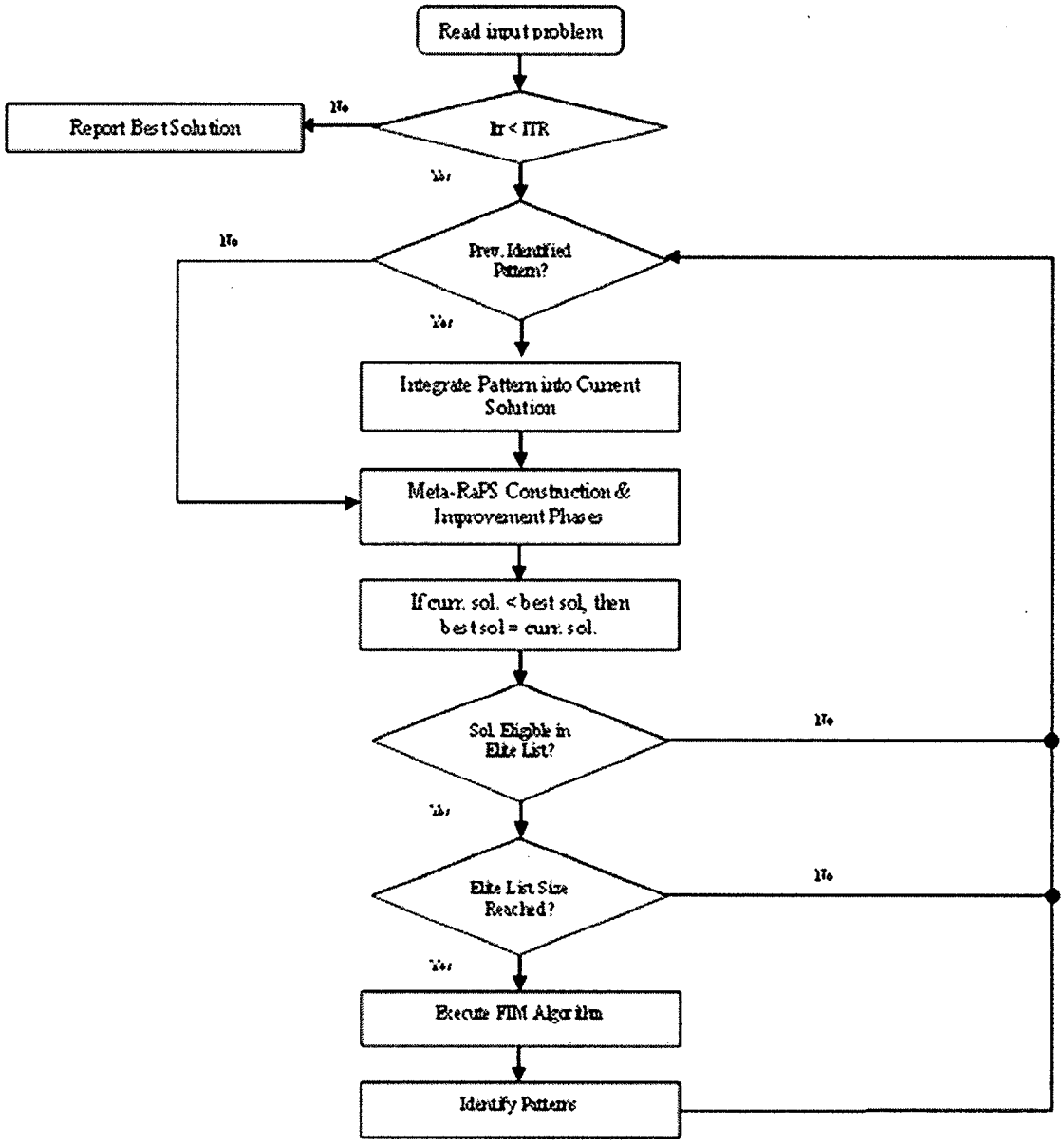


Figure 23 MR FIM Framework

7.2 Frequent Itemset Mining – Apriori Algorithm

The Apriori algorithm is designed to mine databases to discover relationships between items without any initial knowledge. The process starts with creating the database of candidates containing itemsets to be mined. Followed by identifying statistically significant itemsets and identifying the association rules between the itemsets.

With the Meta-RaPS FIM framework, the need to identify that a relationship exists between itemset is eliminated. The relationship of interest is defined by the elite list. With the list only containing best solutions Meta-RaPS generated so far, the purpose of learning becomes the identification of the patterns that are common between elite solutions. This makes identifying frequent itemsets that pass the Support% threshold is a sufficient (eliminating the need to calculate the Confidence and Lift between itemsets). Identifying the frequent itemsets follows the Apriori principles where

- Any subset of a frequent itemset is frequent
- No supersets of infrequent itemsets should be generated

To identify frequent patterns, the following process is executed:

1- Define the smallest itemsets. In the case of VRP, the smallest itemsets are arcs connecting two customers.
2- Mine elite list for frequent itemsets
3- Generate supersets by joining frequent itemsets. In the case of VRP, joining itemsets is done if the end of a smaller itemset (coordinates of the last customer customer) is the same as the start of another smaller itemset (coordinates of the first customer)
4- Repeat steps 2 and 3 until joining frequent itemsets is infeasible.
5- Return the last joined itemsets

The steps above identify the longest most frequent pattern among the solutions in the elite list. If the solutions in the elite list have a frequent, common pattern, then the next

step is to integrate this pattern into future Meta-RaPS solutions. This is done by starting future Meta-RaPS solutions with the pattern.

As described in the Meta-RaPS chapter, utilizing the C&W heuristic to construct VRP solution (original Meta-RaPS) starts with creating temporary short routes that start with the central depot, service a single customer, and end at the central depot. The temporary short routes are created for each customer in the input problem. The process of creating C&W temporary short routes is modified in Meta-RaPS FIM. With patterns provided from previous Meta-RaPS iteration, creating temporary short routes is limited to the customers that are not in the patterns. Temporary (though longer) routes are created for the patterns. This is done by starting and ending the patterns with central depot. This modification allows minimal change to the C&W heuristic. As with the original Meta-RaPS, the temporary routes will be evaluated against the savings list in order for temporary routes to be joined without violating the C&W conditions.

CHAPTER 8

COMPUTATIONAL EXPERIMENT DESIGN

The objective of this dissertation is to conduct controlled experimental studies to evaluate the impact of hybridizing Meta-RaPS with data mining machine learning algorithms. The data mining algorithms selected for this study are the Inductive Decision Tree (IDT) and the Frequent Itemset Mining (FIM) algorithms. The research hypothesis of this study specifies that hybridized Meta-RaPS outperform the original Meta-RaPS.

To conduct this study, Visual Basic programs are implemented using Microsoft Visual Studio 2010 development environment. The programs represent three major experiments: original Meta-RaPS (MR) hybridized Meta-RaPS with IDT (MR IDT), and hybridized Meta-RaPS with FIM (MR FIM). The same benchmark instances of the Capacitated Vehicle Routing Problem (CVRP) are used as input to all experiments. All experiments were run using Intel Xeon E5-2666v3 processors with a base speed of 2.9 GHz and 3.75 GB of memory.

8.1 Input Problems

The input problems used for both experiments belong to a set of CVRP benchmark problems: CMT benchmark problems. These problems are generated by Christofides, Mingozzi, and Toth (1979). The CMT benchmark problem set contains 14 CVRP instances. These problems represent smaller CVRP problems with number of customers ranging from 50 to 199. The Golden, Wasil, Kelly, and Chao (1998) or GWKC benchmark problem set contains 20 CVRP instance with more customers. The number of customers in the GWKC CVRP problems range from 200 to 480. The GWKC problems will not be used here.

The CVRP instances for both sets are available online (http://www.bernabe.dorronsoro.es/vrp/). With each X, Y coordinates, the demands for

each customer in the problem are listed. Each file also contains the maximum load allowed on the truck.

In addition to the truck capacity constraints, some of the instances enforce a maximum route time duration constraint. It is assumed that the time and distance between each customer are the same (Ursani, 2009). To account for the route time, the instances include a service (or drop) time. The service time represents the time associated with servicing each customer.

Instance	Number of Cities	Constraints	City Distribution	Best Known Value
VRPNC1	50	Truck Capacity	Random	524.6
VRPNC2	75	Truck Capacity	Random	835.3
VRPNC3	100	Truck Capacity	Random	826.1
VRPNC4	150	Truck Capacity	Random	1028.4
VRPNC5	199	Truck Capacity	Random	1291.4
VRPNC6	50	Truck Capacity & Total Tour Time	Random	555.4
VRPNC8	75	Truck Capacity & Total Tour Time	Random	909.7
VRPNC9	100	Truck Capacity & Total Tour Time	Random	865.9
VRPNC10	150	Truck Capacity & Total Tour Time	Random	1162.5
VRPNC11	120	Truck Capacity	Clustered	1395.8
VRPNC12	100	Truck Capacity	Clustered	1042.1
VRPNC13	120	Truck Capacity & Total Tour Time	Clustered	819.6
VRPNC14	100	Truck Capacity & Total Tour Time	Clustered	1541.1

Table 6 CMT Input Problems

8.2 Parameter Tuning

Metaheuristics' parameters represent an essential part of the algorithms. Determining the values of these parameters has a great impact on the quality of solutions produced. Parameter values vary based on the problem solved by the metaheuristic. Even

when solving the same problem, the optimal parameter values are likely to vary from one input instance to the other (Wolpert & Macready, 1997). The process of discovering suitable parameter values is referred to as parameter tuning.

To tune the parameters used in the original Meta-RaPS algorithm, an offline two-level factorial design of experiment was followed (Table 7). Deciding the values of each level can either be determined by a Subject Matter Expert or selected randomly. Though the values here were selected randomly, other than the values used by other researchers (Moraga, 2002; Song, 2005).

Levels	Original Meta-RaPS Parameters			
	p%	r%	i%	ITR
Low	40	30	20	250
High	80	70	60	500

Table 7 Two-level Factorial Design Values for Meta-RaPS Parameters

To identify parameter values, five randomly selected problem instances were run for 20 times with every combination of design factors. The combination of parameters that led to best solution quality (least deviation from best known value) is 80% for p%, 30% for r%, 60% for i%, and 500 iterations for ITR.

8.3 Algorithm Performance

To demonstrate the performance in the algorithms, the average solution quality is used as an indicator. Solution quality is defined as the percent deviation or relative error in relation to the Best Known Values (BKV) for the input instances. Another indicator of quality is the percentage that the hybridized Meta-RaPS algorithms deviate from the original Meta-RaPS algorithm. The same percent deviation equation can be used:

$$\frac{f(s')-f(s)}{f(s)} * 100 \quad (11)$$

where s represent the solution value for the BKV and s' represent the solution value found by the algorithms in this dissertation. When considering deviating from the original MR, it is critical to indicate that the deviation is due to learning.

All solutions represent the total distance traveled by all the trucks used to serve all the customers in the input problem. However, due to the large role randomness plays in Meta-RaPS, algorithms are repeated 10 times (Arin, 2012). However, when analyzing performance, values are averaged to describe the quality of the solutions. Averaging will not be used when describing the best solution produced (out of the 10 runs).

It is worth noting that the performance of metaheuristics is typically measured with CPU run time. This performance indicator is critical for metaheuristics, as they enjoy arriving at acceptable solutions for complex problems in reasonable amount of time. However, CPU run times can depend on many factors and usually vary from one study to the other, which makes the comparison illogical. CPU run time can be viewed as an advantage (or disadvantage) if it was directly improved (degraded) by an algorithm.

8.4 Meta-RaPS Inductive Decision Tree Experiment

Implementing the MR IDT algorithm involved evaluating various approaches (Figure 24). They consisted of varying major aspects of the hybridization algorithm, which are:

1- Kick-out mechanism or frequency of running ID3 algorithm: whether or not to employ a kick-out mechanism. This aspect can also be viewed as the minimum frequency of ID3 execution, which without a kick-out mechanism is one.

2- Makeup of knowledge: whether to pass to future iteration (or fix) the root node parameter only or fix all parameters that lead to a 'Good' leaf node.

3- Labeling solutions: when collecting TE, a Meta-RaPS solution is to be labeled as either 'Good' or 'Bad'. Deciding that a solution is 'Good' can be done by comparing it to the best known value of the problem being solved. Labeling a solution can also be based on comparing its value to a moving average value.
4- Attribute values:
    a. Knowledge Attribute(s) can either
        i. Be set to the original Meta-RaPS parameter value, or
        ii. Randomly selected within range of original Meta-RaPS parameter value
    b. Non-Knowledge Attributes can either
        i. Be randomly selected, or
        ii. Randomly selected within range of original Meta-RaPS parameter value

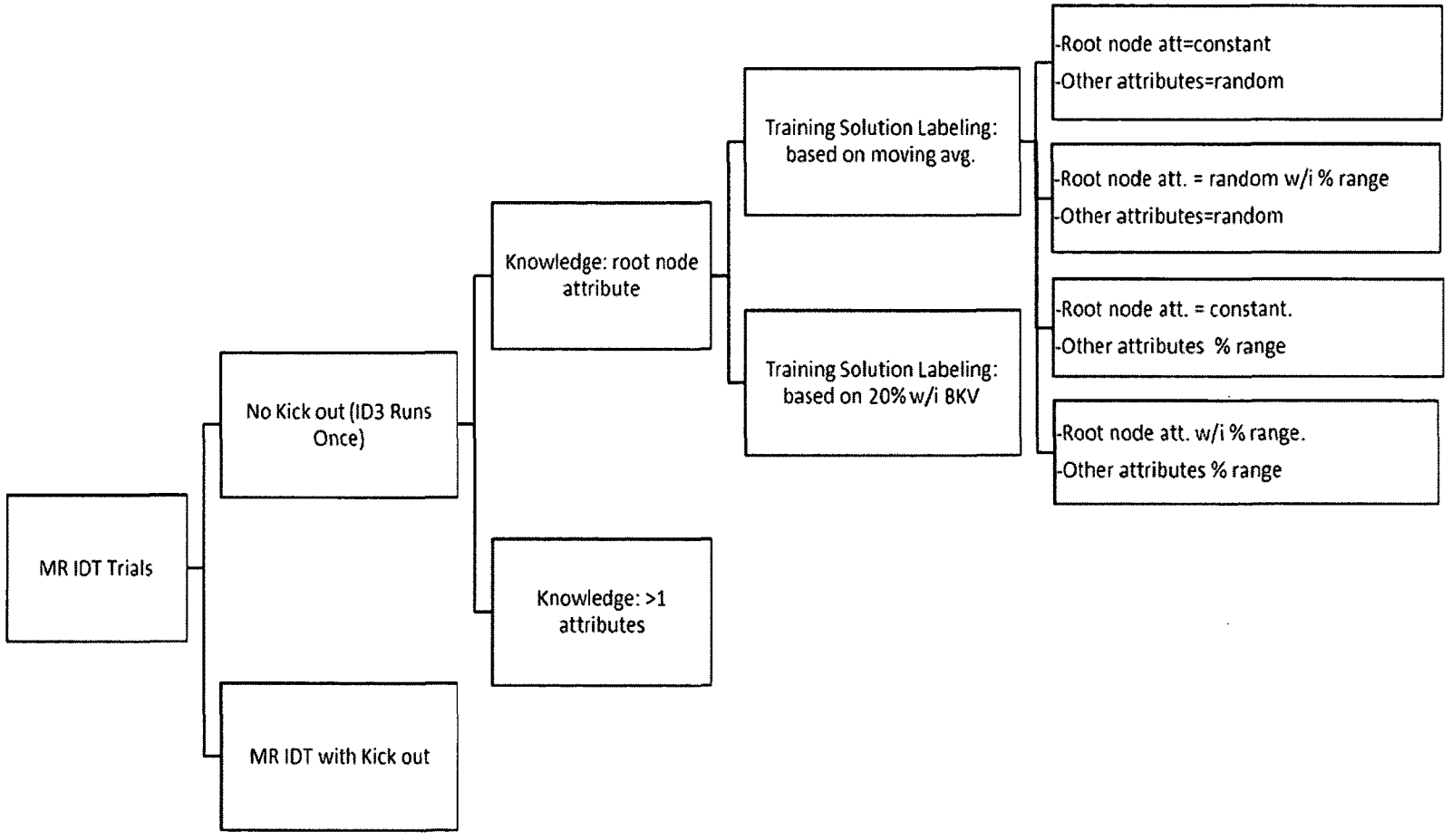


Figure 24 MR IDT Trials

Note, that every combination of the aspects listed above (and illustrated in the figure above) was examined. All of the MR IDT trials make use of Meta-RaPS parameters. In addition to the p%, r%, i%, and ITR parameters, MR IDT introduces TE% and avg% parameters. The TE% parameter is used to determine the minimum number of training examples to be used as input for the ID3. The avg% parameter is associated with evaluating the impact of the knowledge on the performance of Meta-RaPS.

To set the values for these parameters, the same two-level factorial design of experiment approach was followed. The same subset of input problems used to tune parameters in the original Meta-RaPS are used (vrpnc1, vrpnc2, vrpnc6, vrpnc7, vrpnc11). The original Meta-RaPS parameters values were not changed. The values for the levels of MR IDT parameters are randomly selected as follows (Table 8).

Experiment	Design Factors	Levels
MR IDT	*TE%*	25%
		50%
	Avg%	5
		10

Table 8 MR IDT Design Factors Values

To select the appropriate design factor levels, every combination of design factors was run 20 times. The combination of parameters that led to best solution quality (least deviation from optimal value) is 50% for TE% and 10 for Avg%. Due to its relative small value (relative to the TE% parameter), avg% overloaded as both the time when evaluating tree performance is needed and as the size of the moving average window. In trials where fixed or non-fixed parameters are randomly selected within a range of the original Meta-RaPS value, avg% value is used as the range cutoff.

8.5 Meta-RaPS Inductive Decision Tree Computational Results

The following tables (9-18??) show the computational results associated with all the MR IDT trials. The summary table shows two values:

1- The number of times MR IDT outperformed the original MR with learning.

2- The number of times MR IDT was on average better than the original MR.

Note "number of times" is relative to the number of input problems. In the case of CMT problems: maximum "number of times" is 14. The approaches columns represent:

1- Approach 1: single IDT run; TE labeling based on moving average or Best Known Value; knowledge: single attribute (root node)

2- Approach 2: multiple IDT runs; TE labeling based on moving average or Best Known Value; knowledge: single attribute (root node)

3- Approach 3: single IDT run; TE labeling based on moving average or Best Known Value; knowledge: multiple attributes

4- Approach 4: multiple IDT runs; TE labeling based on moving average or Best Known Value; knowledge: multiple attributes

The tables show that the simplest approach with a single IDT run / knowledge represented by the root node / TE labeled based on BKV is the best performing approach.

Trial	TE Label Based on Moving Average				TE Label Based on BKV			
	Approach 1	Approach 2	Approach 3	Approach 4	Approach 1	Approach 2	Approach 3	Approach 4
Knowledge Attribute: Constant; **Other Attributes:** Random	8; 13	10; 12	9; 11	8; 11	**10; 13**	9; 12	3; 13	8; 11
Knowledge Attribute: Random w/i % range; **Other attributes:** Random	7; 14	8; 12	11; 11	8; 13	7; 13	7; 12	9; 12	8; 13
Knowledge Attribute: Constant; **Other Attributes:** Random w/i % range;	1; 1	0; 2	10; 2	9; 12	1; 11	3; 1	0; 11	9; 12
Knowledge Attribute: Random w/i % range; **Other Attributes:** Random w/i % range;	0; 2	2; 0	8; 2	10; 3	0; 11	0; 0	7; 13	7; 2

Table 9 MR IDT Summary Results

The following table shows the results in terms of summary of averages. The values presented here show the average deviation from BKV for the various MR IDT trials. An average value is based on the best results produced by the MR IDT trial and spans over all the input problems. The average deviation for the original MR for the CMT input problems is 4.15.

Trial	TE Label Based on Moving Average				TE Label Based on BKV			
	Approach 1	Approach 2	Approach 3	Approach 4	Approach 1	Approach 2	Approach 3	Approach 4
Knowledge Attribute: Constant; **Other Attributes:** Random	3.47	3.53	3.50	3.60	**3.18**	3.61	3.60	3.60
Knowledge Attribute: Random w/i % range; **Other attributes:** Random	3.58	3.58	3.62	3.62	3.26	3.47	3.54	3.62
Knowledge Attribute: Constant; **Other Attributes:** Random w/i % range;	7.60	7.86	3.62	3.56	3.64	7.19	3.72	3.56
Knowledge Attribute: Random w/i % range; **Other Attributes:** Random w/i % range;	8.18	5.55	3.62	3.59	3.95	8.49	3.46	6.52

Table 10 MR IDT Average Summary Results

VRPNC	BKV	MR				MR IDT: Single IDT Run; TE Label Based on Moving Average; Knowledge: Single Attribute (Root Node)															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	533.42	1.68	547.10	4.29	545.04	3.89	549.90	4.82	544.67	3.82	552.69	5.35	546.27	4.13	551.58	5.14
2	835.26	845.75	1.26	866.89	3.79	850.32	1.80	863.87	3.42	855.84	2.46	864.88	3.55	868.69	4.00	877.60	5.07	864.21	3.47	877.34	5.04
3	826.14	858.32	3.90	865.27	4.74	858.32	3.90	862.16	4.36	856.85	3.72	861.34	4.26	871.98	5.55	889.44	7.66	885.01	7.13	890.36	7.77
4	1028.42	1089.35	5.92	1093.42	6.32	1072.69	4.30	1086.73	5.67	1074.86	4.52	1087.94	5.79	1152.21	12.04	1174.82	14.24	1141.35	10.98	1161.02	12.89
5	1291.29	1389.18	7.58	1397.87	8.25	1371.07	6.18	1382.91	7.10	1379.40	6.82	1389.24	7.59	1503.21	16.41	1524.58	18.07	1501.75	16.30	1518.05	17.56
6	555.43	574.52	3.44	582.34	4.84	576.72	3.83	582.12	4.80	567.48	2.17	579.81	4.39	565.66	1.84	582.22	4.82	572.84	3.13	580.19	4.46
7	909.68	944.74	3.85	952.59	4.72	942.49	3.61	948.22	4.24	942.83	3.64	948.70	4.29	955.25	5.01	963.04	5.87	948.92	4.31	961.87	5.74
8	865.94	927.19	7.07	936.00	8.09	902.10	4.18	922.90	6.58	917.94	6.00	924.98	6.82	925.15	6.84	958.15	10.65	931.28	7.55	958.68	10.71
9	1162.55	1243.96	7.00	1252.23	7.71	1233.83	6.13	1244.05	7.01	1224.96	5.37	1236.89	6.39	1315.90	13.19	1337.69	15.07	1298.64	11.71	1335.26	14.86
10	1395.85	1531.73	9.73	1543.66	10.59	1520.86	8.96	1535.76	10.02	1503.31	7.70	1528.08	9.47	1625.08	16.42	1693.13	21.30	1656.74	18.69	1692.95	21.28
11	1042.11	1058.81	1.60	1065.67	2.26	1054.73	1.21	1065.60	2.25	1054.25	1.16	1065.16	2.21	1158.54	11.17	1213.21	16.42	1174.23	12.68	1221.96	17.26
12	819.56	826.36	0.83	829.65	1.23	822.62	0.37	825.74	0.75	822.78	0.39	825.45	0.72	855.77	4.42	860.08	4.94	850.57	3.78	856.85	4.55
13	1541.14	1570.38	1.90	1579.15	2.47	1574.65	2.17	1581.39	2.61	1573.63	2.11	1577.77	2.38	1579.98	2.52	1686.38	9.42	1659.93	7.71	1693.33	9.88
14	866.37	868.50	0.25	871.07	0.54	869.10	0.32	870.78	0.51	868.85	0.29	870.89	0.52	894.46	3.24	903.13	4.24	892.29	2.99	904.15	4.36

Table 11 MR IDT: 1 IDT Runs; TE Label: Moving Average; Knowledge: Root

VRPNC	BKV	MR				MR IDT: Multiple IDT Run; TE Label Based on Moving Average; Knowledge: Single Attribute (Root Node)															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	534.68	1.92	548.08	4.47	539.12	2.77	550.33	4.90	539.29	2.80	550.04	4.85	556.85	6.15	577.92	10.16
2	835.26	845.75	1.26	866.89	3.79	849.89	1.75	859.10	2.85	849.89	1.75	860.24	2.99	864.55	3.51	876.51	4.94	877.48	5.05	890.89	6.66
3	826.14	858.32	3.90	865.27	4.74	849.91	2.88	858.95	3.97	857.08	3.75	861.64	4.30	883.33	6.92	892.66	8.05	869.64	5.27	869.64	5.27
4	1028.42	1089.35	5.92	1093.42	6.32	1083.40	5.35	1088.90	5.88	1072.98	4.33	1088.90	5.88	1152.61	12.08	1175.62	14.31	1085.53	5.55	1095.33	6.51
5	1291.29	1389.18	7.58	1397.87	8.25	1375.52	6.52	1386.90	7.40	1373.60	6.37	1386.36	7.36	1497.06	15.94	1527.21	18.27	1404.74	8.79	1411.14	9.28
6	555.43	574.52	3.44	582.34	4.84	576.14	3.73	582.21	4.82	574.99	3.52	584.17	5.17	566.30	1.96	579.60	4.35	594.11	6.96	611.00	10.00
7	909.68	944.74	3.85	952.59	4.72	941.23	3.47	949.04	4.33	942.83	3.64	947.24	4.13	949.11	4.33	961.64	5.71	971.72	6.82	971.72	6.82
8	865.94	927.19	7.07	936.00	8.09	910.44	5.14	927.10	7.06	910.88	5.19	923.87	6.69	953.21	10.08	965.19	11.46	944.31	9.05	949.62	9.66
9	1162.55	1243.96	7.00	1252.23	7.71	1224.81	5.36	1242.66	6.89	1228.66	5.69	1240.68	6.72	1329.86	14.39	1344.38	15.64	1237.22	6.42	1289.73	10.94
10	1395.85	1531.73	9.73	1543.66	10.59	1519.94	8.89	1534.22	9.91	1518.02	8.75	1531.86	9.74	1680.58	20.40	1698.48	21.68	1546.76	10.81	1578.22	13.07
11	1042.11	1058.81	1.60	1065.67	2.26	1058.40	1.56	1064.90	2.19	1058.97	1.62	1067.58	2.44	1078.57	3.50	1175.51	12.80	1067.96	2.48	1069.87	2.66
12	819.56	826.36	0.83	829.65	1.23	824.21	0.57	827.18	0.93	824.43	0.59	828.46	1.09	845.34	3.15	854.62	4.28	831.04	1.40	835.60	1.96
13	1541.14	1570.38	1.90	1579.15	2.47	1574.77	2.18	1582.65	2.69	1570.88	1.93	1578.86	2.45	1662.04	7.85	1719.13	11.55	1574.14	2.14	1580.02	2.52
14	866.37	868.50	0.25	871.07	0.54	868.50	0.25	871.49	0.59	868.50	0.25	870.34	0.46	893.89	3.18	904.40	4.39	873.45	0.82	874.17	0.90

Table 12 MR IDT: >1 IDT Run; TE Label: Moving Average; Knowledge: Root

VRPNC	BKV	MR				MR IDT: Single IDT Run; TE Label Based on Moving Average; Knowledge: Multiple Attributes															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	545.93	4.06	551.65	5.16	537.29	2.42	550.08	4.85	533.42	2.42	546.15	4.11	534.68	1.92	549.72	4.79
2	835.26	845.75	1.26	866.89	3.79	856.93	2.59	868.39	3.97	857.41	2.65	862.60	3.27	858.77	2.65	868.43	3.97	849.91	1.75	867.00	3.80
3	826.14	858.32	3.90	865.27	4.74	855.66	3.57	864.41	4.63	852.77	3.22	859.99	4.10	855.59	3.22	871.15	5.45	855.39	3.54	873.44	5.73
4	1028.42	1089.35	5.92	1093.42	6.32	1075.37	4.56	1088.55	5.85	1079.67	4.98	1089.69	5.96	1074.13	4.98	1114.40	8.36	1079.79	4.99	1121.42	9.04
5	1291.29	1389.18	7.58	1397.87	8.25	1370.04	6.10	1390.66	7.70	1377.09	6.64	1390.91	7.71	1378.08	6.64	1476.25	14.32	1379.84	6.86	1472.44	14.03
6	555.43	574.52	3.44	582.34	4.84	561.52	1.10	580.45	4.50	574.13	3.37	583.84	5.12	562.63	3.37	578.10	4.08	566.94	2.07	580.50	4.51
7	909.68	944.74	3.85	952.59	4.72	942.49	3.61	947.00	4.10	944.63	3.84	950.76	4.52	944.63	3.84	952.80	4.74	948.92	4.31	955.69	5.06
8	865.94	927.19	7.07	936.00	8.09	916.28	5.81	929.76	7.37	910.16	5.11	927.40	7.10	916.36	5.11	944.82	9.11	926.56	7.00	947.94	9.47
9	1162.55	1243.96	7.00	1252.23	7.71	1207.12	3.83	1244.00	7.01	1220.72	5.00	1238.18	6.51	1232.22	5.00	1279.23	10.04	1217.59	4.73	1282.64	10.33
10	1395.85	1531.73	9.73	1543.66	10.59	1525.94	9.32	1541.67	10.45	1515.42	8.57	1535.69	10.02	1520.22	8.57	1575.40	12.86	1512.58	8.36	1583.92	13.47
11	1042.11	1058.81	1.60	1065.67	2.26	1060.01	1.72	1066.42	2.33	1063.17	2.02	1066.86	2.37	1075.75	2.02	1135.16	8.93	1065.74	2.27	1137.74	9.18
12	819.56	826.36	0.83	829.65	1.23	823.11	0.43	827.53	0.97	824.13	0.56	828.43	1.08	823.55	0.56	839.98	2.49	823.84	0.52	834.35	1.81
13	1541.14	1570.38	1.90	1579.15	2.47	1573.63	2.11	1580.85	2.58	1573.08	2.07	1579.27	2.47	1578.39	2.07	1611.57	4.57	1573.74	2.12	1617.73	4.97
14	866.37	868.50	0.25	871.07	0.54	868.50	0.25	870.84	0.52	868.50	0.25	870.19	0.44	868.50	0.25	888.77	2.59	869.31	0.34	887.59	2.45

Table 13 MR IDT: 1 IDT Run; TE Label: Moving Average; Knowledge:>1 Node

VRPNC	BKV	MR				MR IDT: Multiple IDT Runs; TE Label Based on Moving Average; Knowledge: Multiple Attributes															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	534.51	1.89	550.20	4.88	534.68	1.92	547.68	4.40	533.42	1.68	548.14	4.48	541.17	3.16	550.35	4.91
2	835.26	845.75	1.26	866.89	3.79	850.99	1.88	864.18	3.46	862.79	3.30	864.87	3.54	852.32	2.04	862.91	3.31	856.78	2.58	866.91	3.79
3	826.14	858.32	3.90	865.27	4.74	853.55	3.32	859.74	4.07	858.05	3.86	860.26	4.13	855.95	3.61	858.76	3.95	855.53	3.56	874.57	5.86
4	1028.42	1089.35	5.92	1093.42	6.32	1076.75	4.70	1085.03	5.50	1076.10	4.64	1082.68	5.28	1072.26	4.26	1083.27	5.33	1079.58	4.97	1121.29	9.03
5	1291.29	1389.18	7.58	1397.87	8.25	1375.72	6.54	1384.69	7.23	1377.17	6.65	1385.54	7.30	1377.65	6.69	1384.71	7.23	1376.93	6.63	1410.74	9.25
6	555.43	574.52	3.44	582.34	4.84	572.88	3.14	582.40	4.86	568.03	2.27	581.27	4.65	576.14	3.73	581.59	4.71	562.63	1.30	579.91	4.41
7	909.68	944.74	3.85	952.59	4.72	945.97	3.99	950.89	4.53	946.10	4.00	949.36	4.36	943.63	3.73	948.61	4.28	941.08	3.45	950.35	4.47
8	865.94	927.19	7.07	936.00	8.09	921.62	6.43	927.66	7.13	910.85	5.19	924.78	6.80	908.12	4.87	919.43	6.18	917.45	5.95	936.98	8.20
9	1162.55	1243.96	7.00	1252.23	7.71	1228.30	5.66	1236.25	6.34	1228.03	5.63	1239.18	6.59	1227.98	5.63	1236.94	6.40	1230.35	5.83	1285.56	10.58
10	1395.85	1531.73	9.73	1543.66	10.59	1518.60	8.79	1528.74	9.52	1519.63	8.87	1528.35	9.49	1519.35	8.85	1529.64	9.58	1509.50	8.14	1559.26	11.71
11	1042.11	1058.81	1.60	1065.67	2.26	1056.33	1.36	1064.77	2.17	1058.26	1.55	1064.79	2.18	1060.81	1.79	1066.13	2.31	1065.11	2.21	1101.00	5.65
12	819.56	826.36	0.83	829.65	1.23	823.11	0.43	826.51	0.85	823.97	0.54	826.91	0.90	824.71	0.63	827.18	0.93	823.71	0.51	837.31	2.17
13	1541.14	1570.38	1.90	1579.15	2.47	1573.28	2.09	1580.77	2.57	1574.04	2.13	1579.69	2.50	1574.41	2.16	1580.69	2.57	1569.96	1.87	1602.61	3.99
14	866.37	868.50	0.25	871.07	0.54	868.50	0.25	871.64	0.61	868.50	0.25	869.85	0.40	868.50	0.25	869.99	0.42	868.41	0.24	874.86	0.98

Table 14 MR IDT: >1 IDT Runs; TE Label: Moving Average; Knowledge: >1 Node

VRPNC	BKV	MR				MR IDT: Single IDT Runs; TE Label Based on % BKV; Knowledge: Single Attribute (Root Node)															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	**533.42**	**1.68**	**545.75**	**4.03**	534.68	1.92	545.24	3.93	537.29	2.42	546.92	4.25	542.35	3.38	548.77	4.60
2	835.26	845.75	1.26	866.89	3.79	**854.71**	**2.33**	**862.16**	**3.22**	849.91	1.75	859.35	2.88	849.89	1.75	862.31	3.24	849.89	1.75	860.78	3.05
3	826.14	858.32	3.90	865.27	4.74	**852.92**	**3.24**	**859.15**	**4.00**	855.66	3.57	858.93	3.97	854.88	3.48	859.43	4.03	852.31	3.17	856.65	3.69
4	1028.42	1089.35	5.92	1093.42	6.32	**1076.10**	**4.64**	**1085.02**	**5.50**	1076.81	4.71	1084.07	5.41	1080.49	5.06	1085.42	5.54	1077.88	4.81	1084.44	5.45
5	1291.29	1389.18	7.58	1397.87	8.25	**1368.94**	**6.01**	**1380.70**	**6.92**	1364.78	5.69	1383.17	7.12	1381.27	6.97	1397.20	8.20	1384.03	7.18	1394.78	8.01
6	555.43	574.52	3.44	582.34	4.84	**560.81**	**0.97**	**578.86**	**4.22**	560.81	0.97	579.25	4.29	568.03	2.27	579.08	4.26	573.41	3.24	580.76	4.56
7	909.68	944.74	3.85	952.59	4.72	**938.46**	**3.16**	**944.93**	**3.87**	946.10	4.00	948.12	4.23	933.97	2.67	943.59	3.73	933.64	2.63	940.02	3.33
8	865.94	927.19	7.07	936.00	8.09	**909.23**	**5.00**	**922.44**	**6.52**	909.06	4.98	920.66	6.32	890.70	2.86	917.65	5.97	896.64	3.55	919.22	6.15
9	1162.55	1243.96	7.00	1252.23	7.71	**1223.31**	**5.23**	**1235.94**	**6.31**	1224.96	5.37	1232.59	6.02	1237.97	6.49	1243.38	6.95	1239.02	6.58	1244.34	7.04
10	1395.85	1531.73	9.73	1543.66	10.59	**1498.38**	**7.35**	**1529.42**	**9.57**	1510.64	8.22	1523.86	9.17	1527.97	9.47	1536.93	10.11	1536.16	10.05	1540.52	10.36
11	1042.11	1058.81	1.60	1065.67	2.26	**1064.02**	**2.10**	**1066.43**	**2.33**	1059.86	1.70	1065.69	2.26	1072.90	2.95	1088.95	4.50	1088.22	4.42	1093.88	4.97
12	819.56	826.36	0.83	829.65	1.23	**823.11**	**0.43**	**825.80**	**0.76**	822.82	0.40	825.17	0.68	826.25	0.82	828.78	1.12	825.87	0.77	829.30	1.19
13	1541.14	1570.38	1.90	1579.15	2.47	**1572.33**	**2.02**	**1578.00**	**2.39**	1574.09	2.14	1579.11	2.46	1596.14	3.57	1602.69	3.99	1592.56	3.34	1601.24	3.90
14	866.37	868.50	0.25	871.07	0.54	**869.62**	**0.37**	**871.04**	**0.54**	868.83	0.28	869.64	0.38	868.34	0.23	871.60	0.60	870.82	0.51	872.11	0.66

Table 15 MR IDT: 1 IDT Run; TE Label: BKV; Knowledge: Single Node

VRPNC	BKV	MR				MR IDT: Multiple IDT Runs; TE Label Based on % BKV; Knowledge: Single Attribute (Root Node)															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	537.29	2.42	547.78	4.42	539.12	2.77	551.89	5.20	541.59	3.24	550.27	4.89	546.19	4.11	553.99	5.60
2	835.26	845.75	1.26	866.89	3.79	849.89	1.75	857.26	2.63	849.89	1.75	860.59	3.03	849.91	1.75	872.62	4.47	864.21	3.47	874.98	4.76
3	826.14	858.32	3.90	865.27	4.74	853.36	3.29	860.25	4.13	855.06	3.50	860.62	4.17	854.75	3.46	888.16	7.51	881.46	6.70	893.92	8.20
4	1028.42	1089.35	5.92	1093.42	6.32	1077.97	4.82	1086.02	5.60	1078.68	4.89	1084.29	5.43	1138.11	10.67	1162.94	13.08	1147.50	11.58	1161.58	12.95
5	1291.29	1389.18	7.58	1397.87	8.25	1377.78	6.70	1384.92	7.25	1376.12	6.57	1384.68	7.23	1499.92	16.16	1523.88	18.01	1493.73	15.68	1512.52	17.13
6	555.43	574.52	3.44	582.34	4.84	576.48	3.79	581.61	4.71	564.70	1.67	584.29	5.20	577.17	3.91	583.92	5.13	579.10	4.26	582.55	4.88
7	909.68	944.74	3.85	952.59	4.72	943.50	3.72	949.42	4.37	946.10	4.00	948.66	4.29	948.52	4.27	961.76	5.72	954.25	4.90	960.36	5.57
8	865.94	927.19	7.07	936.00	8.09	909.85	5.07	924.06	6.71	912.88	5.42	928.71	7.25	919.91	6.23	951.68	9.90	939.51	8.50	954.37	10.21
9	1162.55	1243.96	7.00	1252.23	7.71	1233.54	6.11	1241.12	6.76	1216.84	4.67	1234.83	6.22	1310.79	12.75	1338.22	15.11	1313.69	13.00	1336.42	14.96
10	1395.85	1531.73	9.73	1543.66	10.59	1518.68	8.80	1531.94	9.75	1521.50	9.00	1535.23	9.99	1653.32	18.45	1693.43	21.32	1672.18	19.80	1688.42	20.96
11	1042.11	1058.81	1.60	1065.67	2.26	1056.86	1.42	1064.29	2.13	1060.52	1.77	1065.06	2.20	1173.35	12.59	1199.21	15.08	1151.26	10.47	1199.22	15.08
12	819.56	826.36	0.83	829.65	1.23	822.95	0.41	825.26	0.70	822.44	0.35	827.02	0.91	828.60	1.10	852.31	4.00	856.33	4.49	862.01	5.18
13	1541.14	1570.38	1.90	1579.15	2.47	1572.36	2.03	1580.43	2.55	1572.03	2.00	1580.06	2.53	1632.90	5.95	1691.54	9.76	1674.34	8.64	1704.91	10.63
14	866.37	868.50	0.25	871.07	0.54	868.50	0.25	871.69	0.61	868.50	0.25	870.45	0.47	868.50	0.25	897.15	3.55	895.14	3.32	904.04	4.35

Table 16 MR IDT: >1 IDT Run; TE Label: BKV; Knowledge: Single Node

VRPNC	BKV	MR				MR IDT: Single IDT Run; TE Label Based on BKV; Knowledge: Multiple Attributes															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	546.48	4.17	550.85	5.00	542.15	3.34	549.90	4.82	543.77	3.65	550.13	4.86	537.09	2.38	547.67	4.39
2	835.26	845.75	1.26	866.89	3.79	854.78	2.34	863.42	3.37	862.79	3.30	866.00	3.68	849.89	1.75	859.48	2.90	849.91	1.75	862.81	3.30
3	826.14	858.32	3.90	865.27	4.74	854.20	3.40	858.69	3.94	856.93	3.73	859.61	4.05	850.01	2.89	858.15	3.88	854.68	3.45	859.60	4.05
4	1028.42	1089.35	5.92	1093.42	6.32	1081.90	5.20	1086.16	5.61	1079.05	4.92	1084.58	5.46	1076.08	4.63	1084.52	5.46	1078.15	4.84	1087.04	5.70
5	1291.29	1389.18	7.58	1397.87	8.25	1381.59	6.99	1387.46	7.45	1368.37	5.97	1383.02	7.10	1382.59	7.07	1394.02	7.96	1367.66	5.91	1383.69	7.16
6	555.43	574.52	3.44	582.34	4.84	561.66	1.12	581.43	4.68	560.35	0.89	583.55	5.06	576.11	3.72	581.62	4.72	576.46	3.79	583.77	5.10
7	909.68	944.74	3.85	952.59	4.72	944.63	3.84	950.33	4.47	945.97	3.99	950.44	4.48	934.24	2.70	943.09	3.67	937.30	3.04	947.36	4.14
8	865.94	927.19	7.07	936.00	8.09	914.03	5.55	924.91	6.81	919.13	6.14	926.19	6.96	895.33	3.39	920.30	6.28	904.57	4.46	921.12	6.37
9	1162.55	1243.96	7.00	1252.23	7.71	1226.64	5.51	1233.71	6.12	1216.36	4.63	1233.97	6.14	1203.76	3.54	1235.56	6.28	1228.57	5.68	1240.37	6.69
10	1395.85	1531.73	9.73	1543.66	10.59	1513.40	8.42	1527.17	9.41	1515.03	8.54	1529.41	9.57	1527.18	9.41	1536.60	10.08	1517.15	8.69	1531.63	9.73
11	1042.11	1058.81	1.60	1065.67	2.26	1054.81	1.22	1066.53	2.34	1058.97	1.62	1065.69	2.26	1089.30	4.53	1093.07	4.89	1059.63	1.68	1065.32	2.23
12	819.56	826.36	0.83	829.65	1.23	822.72	0.39	826.24	0.81	823.11	0.43	826.40	0.83	825.94	0.78	828.75	1.12	824.08	0.55	826.30	0.82
13	1541.14	1570.38	1.90	1579.15	2.47	1572.77	2.05	1575.27	2.21	1569.85	1.86	1577.85	2.38	1598.69	3.73	1602.93	4.01	1573.51	2.10	1578.65	2.43
14	866.37	868.50	0.25	871.07	0.54	868.50	0.25	870.35	0.46	868.50	0.25	869.44	0.35	868.83	0.28	871.71	0.62	868.50	0.25	870.13	0.43

Table 17 MR IDT: 1 IDT Run; TE Label: BKV; Knowledge: >1 Nodes

VRPNC	BKV	MR				MR IDT: Multiple IDT Run; TE Label Based on BKV; Knowledge: Multiple Attributes															
						Knowledge Att: constant Other attributes: random				Knowledge Att: rand w/i % range Other attributes: random				Knowledge Att: constant Other attributes: % range				Knowledge Att: w/i % range Other attributes: % range			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	534.51	1.89	550.20	4.88	534.68	1.92	547.68	4.40	533.42	1.68	548.14	4.48	543.46	3.59	552.36	5.29
2	835.26	845.75	1.26	866.89	3.79	850.99	1.88	864.18	3.46	862.79	3.30	864.87	3.54	852.32	2.04	862.91	3.31	864.15	3.46	873.84	4.62
3	826.14	858.32	3.90	865.27	4.74.	853.55	3.32	859.74	4.07	858.05	3.86	860.26	4.13	855.95	3.61	858.76	3.95	855.29	3.53	876.15	6.05
4	1028.42	1089.35	5.92	1093.42	6.32	1076.75	4.70	1085.03	5.50	1076.10	4.64	1082.68	5.28	1072.26	4.26	1083.27	5.33	1078.71	4.89	1149.01	11.73
5	1291.29	1389.18	7.58	1397.87	8.25	1375.72	6.54	1384.69	7.23	1377.17	6.65	1385.54	7.30	1377.65	6.69	1384.71	7.23	1498.41	16.04	1523.77	18.00
6	555.43	574.52	3.44	582.34	4.84	572.88	3.14	582.40	4.86	568.03	2.27	581.27	4.65	576.14	3.73	581.59	4.71	560.10	0.84	582.33	4.84
7	909.68	944.74	3.85	952.59	4.72	945.97	3.99	950.89	4.53	946.10	4.00	949.36	4.36	943.63	3.73	948.61	4.28	944.63	3.84	957.14	5.22
8	865.94	927.19	7.07	936.00	8.09	921.62	6.43	927.66	7.13	910.85	5.19	924.78	6.80	908.12	4.87	919.43	6.18	913.01	5.44	923.82	6.68
9	1162.55	1243.96	7.00	1252.23	7.71	1228.30	5.66	1236.25	6.34	1228.03	5.63	1239.18	6.59	1227.98	5.63	1236.94	6.40	1280.59	10.15	1324.12	13.90
10	1395.85	1531.73	9.73	1543.66	10.59	1518.60	8.79	1528.74	9.52	1519.63	8.87	1528.35	9.49	1519.35	8.85	1529.64	9.58	1665.01	19.28	1681.96	20.50
11	1042.11	1058.81	1.60	1065.67	2.26	1056.33	1.36	1064.77	2.17	1058.26	1.55	1064.79	2.18	1060.81	1.79	1066.13	2.31	1159.02	11.22	1195.12	14.68
12	819.56	826.36	0.83	829.65	1.23	823.11	0.43	826.51	0.85	823.97	0.54	826.91	0.90	824.71	0.63	827.18	0.93	823.37	0.46	851.43	3.89
13	1541.14	1570.38	1.90	1579.15	2.47	1573.28	2.09	1580.77	2.57	1574.04	2.13	1579.69	2.50	1574.41	2.16	1580.69	2.57	1669.95	8.36	1710.08	10.96
14	866.37	868.50	0.25	871.07	0.54	868.50	0.25	871.64	0.61	868.50	0.25	869.85	0.40	868.50	0.25	869.99	0.42	868.50	0.25	895.33	3.34

Table 18 MR IDT: >1 IDT Runs; TE Label: BKV; Knowledge is >1 Nodes

8.6 Meta-RaPS Frequent Itemset Mining Experiment

Implementing the MR FIM algorithm involved studying the behavior of the hybridized algorithm and examining the following aspects:

1- Negative impact of intensification: identifying the longest, most frequent sequence of customers in the elite list represents valuable knowledge. However, this knowledge is based solely on the content of the elite list. An elite list with easy to satisfy eligibility conditions lead to patterns based on average solutions. Integrating these patterns into future Meta-RaPS iterations negatively impacts the performance. Foregoing the use of the patterns with the incorporation of a kick out mechanism is one way to allow diversification.

2- Pattern integration and diversification: starting Meta-RaPS with patterns is an intentional intensification mechanism. However, patterns can lead to very little randomness in during solution construction. This ultimately leads to a repetitive performance where the same solution is produced for most of the iterations after FIM is executed. To allow for more randomness, the first step in constructing a route is forced to be a modified C&W. The goal is avoid using the same (best) arc in the saving list and allow for more randomness by choosing from the restricted candidate list.

3- Elite List characteristics: if the execution of the FIM algorithm is based on reaching the elite list size, then the size of the list should be relative to the elite list eligibility conditions. The FIM algorithm will likely not get executed if the elite list is large with very strict conditions. The following conditions (ordered from least to most strict) were examined.

   a. Include if new solution better than the worst solution in the elite list
   b. Include if new solution is better than the best solution in the elite list
   c. Include if new solution is within range of the best known value for the input problem

4- Support value vs. elite list eligibility condition(s): in the case of strict elite list eligibility conditions, having a high support value may not of value. The

solutions in the elite list may not have long sequence of routes in common. Enforcing a high support value will likely result in no patterns.

5- Supported patterns vs. rare/unique solutions: obtaining a pattern is a result of its frequent appearances among the solutions in the elite list. A side effect of obtaining patterns as a result of their frequency is omitting sequences belonging to very good, yet unique solutions. Since a very good solution is usually rare, its sequences are not likely to pass the Support% threshold.

With the features listed above, the following MR FIM trials were built and executed. They consisted of varying the following factors:

1- FIM Frequency: running FIM and identifying patterns is computationally expensive. However, running FIM once (based on a bad elite list) and identifying a pattern once is likely to lead to a bad, repetitive performance. Therefore, allowing for multiple runs is examined. Re-running the FIM algorithm is triggered by the update of the elite list once a new solution is added to the list. A new solution is added to the list if it is better than the best solution in the list. When a new solution is added, the worst solution is removed from the list.

2- The kick-out mechanism: if employed, it consists of comparing the latest solution to a moving average. If the latest solution is worse, then both the patterns and the content of the elite list are discarded. This allows the Meta-RaPS to revert to being a multi-start method. It also allows for new solutions to make to the elite list.

3- Reacting to very good solutions: if employed, then the content of the elite list is examined. When a newly added solution is within Elite% range of the best known value of the input problem, then the longest route in the solution is used as a starting pattern for future Meta-RaPS iterations. Executing the FIM is omitted.

Note the elite list eligibility condition used consisted of including a new solution if it better than the best solution in the elite list.

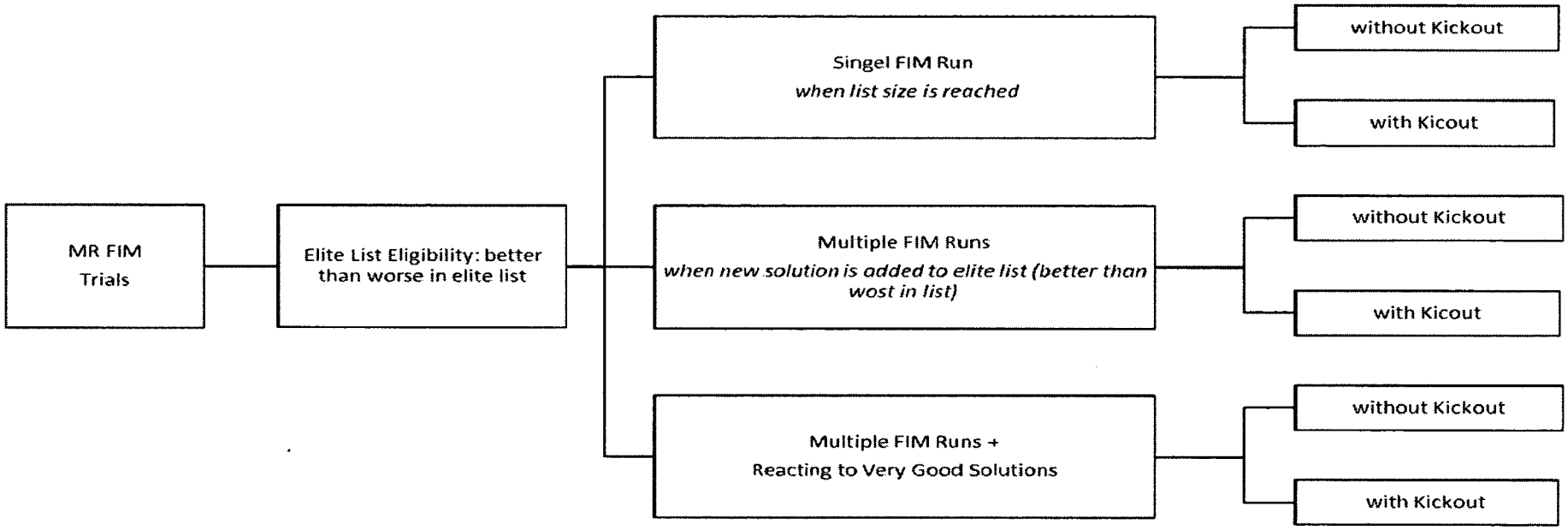


Figure 25 MR FIM Trials

All of the MR FIM trials make use of Meta-RaPS parameters. In addition to the p%, r%, i%, and ITR parameters, MR FIM introduces support (S%) and elite list size (E%) parameters. The Support% parameter is used to determine the frequency threshold percentage that a sequence needs to meet for it to count as a pattern. The E% represents the size of the elite list as a percentage of Meta-RaPS ITR. Some of the MR FIM trials use a moving average as a kick out mechanism. These trials use an Avg% parameter, which is also as a percentage of Meta-RaPS ITR. The trials with reaction to best solutions introduce a parameter (Bst%) to measure the range of deviation from best known problems. Based on this range, FIM might not be executed.

To set the values for these parameters, the same two-level factorial design of experiment approach used to tune the original Meta-RaPS was followed. The original Meta-RaPS parameters values were not changed. The values for the levels of MR FIM parameters are randomly selected as follows (Table 19).

Experiment	Design Factors	Levels
MR FIM	*FIM Pattern Support (S%)*	40
		60
	Elite List Size (E%)	1
		5
	Kick out Moving Avg. (Avg%)	5
		10
	Deviation from BKV (Bst%)	5
		10

Table 19 MR FIM Design Factors Values

To select the appropriate design factor levels, every combination of design factors was run 20 times. The combination of parameters that led to best solution quality (least

deviation from optimal value) is 40, 1, 5, and 5 for S%, E%, Avg%, and Bst% parameters.

8.7 Meta-RaPS Frequent Itemset Mining Computational Results

Similar to MR IDT, the following tables show the computational results associated with all the MR FIM trials. The summary table shows the following values:

1- The number of times MR FIM outperformed the original MR with learning.
2- The number of times MR FIM was on average better than the original MR.
3- Average deviation values, which is based on the best results produced by the MR FIM trial and spans over all the input problems. The average deviation for the original MR for the CMT input problems is 4.15

Note "number of times" is relative to the number of input problems. In the case of CMT problems: maximum "number of times" is 14. The approaches columns represent:

Trial	MR FIM outperforming original MR w/ learning	MR FIM avg. better than original MR	Avg. Deviation from BKV
Single FIM w/o Kick-out	2	2	8.96
Single FIM w/ Kick-out	3	2	10.35
Multiple FIM w/o Kick-out	2	0	8.87
Multiple FIM w/ Kick-out	3	2	9.07
Multiple FIM + good solution reaction w/ Kick-out	1	0	9.15
Multiple FIM + good solution reaction w/ Kick-out	2	2	9.42

Table 20 MR FIM Trials

VRPNC	BKV	MR				Single FIM run w/o Kick-out				Single FIM run w/ Kick-out			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	2.13	-1.60	549.46	4.74	2.48	-1.27	550.41	4.92
2	835.26	845.75	1.26	866.89	3.79	3.46	2.17	874.78	4.73	4.39	3.10	880.19	5.38
3	826.14	858.32	3.90	865.27	4.74	6.94	2.93	894.87	8.32	7.82	3.78	900.84	9.04
4	1028.42	1089.35	5.92	1093.42	6.32	14.39	7.99	1192.63	15.97	14.82	8.40	1198.32	16.52
5	1291.29	1389.18	7.58	1397.87	8.25	19.48	11.06	1566.69	21.33	19.82	11.38	1570.67	21.64
6	555.43	574.52	3.44	582.34	4.84	0.89	-2.47	575.23	3.56	0.84	-2.51	577.71	4.01
7	909.68	944.74	3.85	952.59	4.72	4.80	0.92	963.71	5.94	3.46	-0.38	958.47	5.36
8	865.94	927.19	7.07	936.00	8.09	9.01	1.81	963.80	11.30	10.07	2.80	969.65	11.98
9	1162.55	1243.96	7.00	1252.23	7.71	13.60	6.17	1367.07	17.59	17.04	9.38	1372.72	18.08
10	1395.85	1531.73	9.73	1543.66	10.59	20.35	9.67	1728.32	23.82	24.68	13.62	1753.46	25.62
11	1042.11	1058.81	1.60	1065.67	2.26	14.65	12.84	1259.65	20.87	19.78	17.89	1274.22	22.27
12	819.56	826.36	0.83	829.65	1.23	3.96	3.11	867.33	5.83	3.70	2.84	866.50	5.73
13	1541.14	1570.38	1.90	1579.15	2.47	8.48	6.46	1769.40	14.81	12.52	10.43	1776.95	15.30
14	866.37	868.50	0.25	871.07	0.54	3.33	3.07	909.88	5.02	3.59	3.33	909.75	5.01

Table 21 Single FIM Run

VRPNC	BKV	MR				Multiple FIM runs w/o Kick-out				Multiple FIM runs w/o Kick-out			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	3.13	-0.64	553.55	5.52	3.05	-0.72	547.49	4.36
2	835.26	845.75	1.26	866.89	3.79	2.74	1.46	875.50	4.82	4.27	2.97	879.73	5.32
3	826.14	858.32	3.90	865.27	4.74	5.42	1.47	889.58	7.68	6.37	2.38	894.11	8.23
4	1028.42	1089.35	5.92	1093.42	6.32	12.82	6.51	1187.74	15.49	10.81	4.61	1186.84	15.40
5	1291.29	1389.18	7.58	1397.87	8.25	20.02	11.56	1563.50	21.08	19.73	11.29	1568.26	21.45
6	555.43	574.52	3.44	582.34	4.84	1.39	-1.97	583.94	5.13	1.46	-1.91	575.05	3.53
7	909.68	944.74	3.85	952.59	4.72	3.88	0.03	960.00	5.53	2.99	-0.83	958.88	5.41
8	865.94	927.19	7.07	936.00	8.09	8.66	1.48	964.78	11.41	8.91	1.72	964.54	11.39
9	1162.55	1243.96	7.00	1252.23	7.71	15.60	8.04	1370.66	17.90	16.86	9.22	1377.54	18.49
10	1395.85	1531.73	9.73	1543.66	10.59	21.53	10.75	1738.33	24.54	20.23	9.56	1742.75	24.85
11	1042.11	1058.81	1.60	1065.67	2.26	12.16	10.39	1219.85	17.06	16.23	14.39	1250.10	19.96
12	819.56	826.36	0.83	829.65	1.23	1.65	0.82	859.05	4.82	2.83	1.99	855.17	4.35
13	1541.14	1570.38	1.90	1579.15	2.47	12.11	10.02	1755.51	13.91	10.03	7.98	1756.83	14.00
14	866.37	868.50	0.25	871.07	0.54	3.18	2.93	907.67	4.77	3.30	3.05	908.56	4.87

Table 22 Multiple FIM Runs

VRPNC	BKV	MR				Multiple FIM + runs w/o Kick-out				Multiple FIM runs w/o Kick-out			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	3.53	-0.26	553.12	5.44	3.31	-0.47	549.31	4.71
2	835.26	845.75	1.26	866.89	3.79	3.70	2.42	876.11	4.89	3.01	1.74	875.17	4.78
3	826.14	858.32	3.90	865.27	4.74	5.69	1.73	892.34	8.01	6.22	2.24	894.72	8.30
4	1028.42	1089.35	5.92	1093.42	6.32	13.80	7.44	1183.24	15.05	13.46	7.11	1186.23	15.34
5	1291.29	1389.18	7.58	1397.87	8.25	19.44	11.03	1566.55	21.32	18.83	10.46	1565.61	21.24
6	555.43	574.52	3.44	582.34	4.84	3.75	0.31	583.23	5.01	2.95	-0.47	577.91	4.05
7	909.68	944.74	3.85	952.59	4.72	4.38	0.51	960.37	5.57	4.36	0.49	960.49	5.59
8	865.94	927.19	7.07	936.00	8.09	9.50	2.26	967.77	11.76	9.19	1.97	961.21	11.00
9	1162.55	1243.96	7.00	1252.23	7.71	11.70	4.39	1354.27	16.49	16.65	9.01	1374.90	18.27
10	1395.85	1531.73	9.73	1543.66	10.59	22.13	11.30	1741.89	24.79	22.80	11.90	1739.64	24.63
11	1042.11	1058.81	1.60	1065.67	2.26	16.44	14.60	1249.42	19.89	16.03	14.20	1249.22	19.87
12	819.56	826.36	0.83	829.65	1.23	2.82	1.98	858.01	4.69	3.18	2.33	860.36	4.98
13	1541.14	1570.38	1.90	1579.15	2.47	10.38	8.33	1762.41	14.36	9.43	7.39	1784.62	15.80
14	866.37	868.50	0.25	871.07	0.54	0.88	0.63	905.37	4.50	2.50	2.25	907.39	4.73

Table 23 Multiple FIM Runs with Reacting to Good Solutions

Albeit discouraging, the poor performance of MR FIM led to adjustment in the following aspects of the algorithms:

1- Elite list eligibility: solutions are added to the elite list if they are better than the best. If a solution is better than the worst solution in the elite list, then it replaces the worst solution in the list. This strategy further improves the quality of the solutions in the elite list and potentially improving the quality of the patterns identified by FIM. This strategy, however, delays reaching the elite size and starting the FIM process.

2- Content of the pattern: instead of choosing the longest route out of the pattern, three? other options were evaluated: selecting the most effective route, dividing the routes in the pattern by half, and combination of the aforementioned options (fixing the most effective route in the pattern and fixing half of the remaining routes). The most effective route is the route with the most customers visited in the shortest distance. In an effort to check the performance of these options, they were evaluated by running MR FIM without any reaction to finding very good solutions. The best results were found when the pattern was divided in half. A pattern consisting of the longest or most effective route never included enough customers to make a difference in performance. A frequent single route is constrained by the capacity of the truck, thus preventing the pattern from being very long. In other words, a pattern may include 7 customers. This leaves MR to construct routes for the remaining 43 customers (in the case of the smallest CMT instance) without any influence from the patterns. Note that the option of combining most effective route in the pattern and fixing half of the remaining routes was not noticeably better. So, the approach of dividing routes by half was selected.

3- More reactive kick-out: in addition to kicking out after a certain number of solutions were accumulated after the pattern is identified, a more reactive kick-out mechanism is included. If a rare, yet very good solution was found, then half of its routes are fixed for future iterations. If this leads to bad solutions, then Meta-RaPS reverts to its normal multi-start behavior.

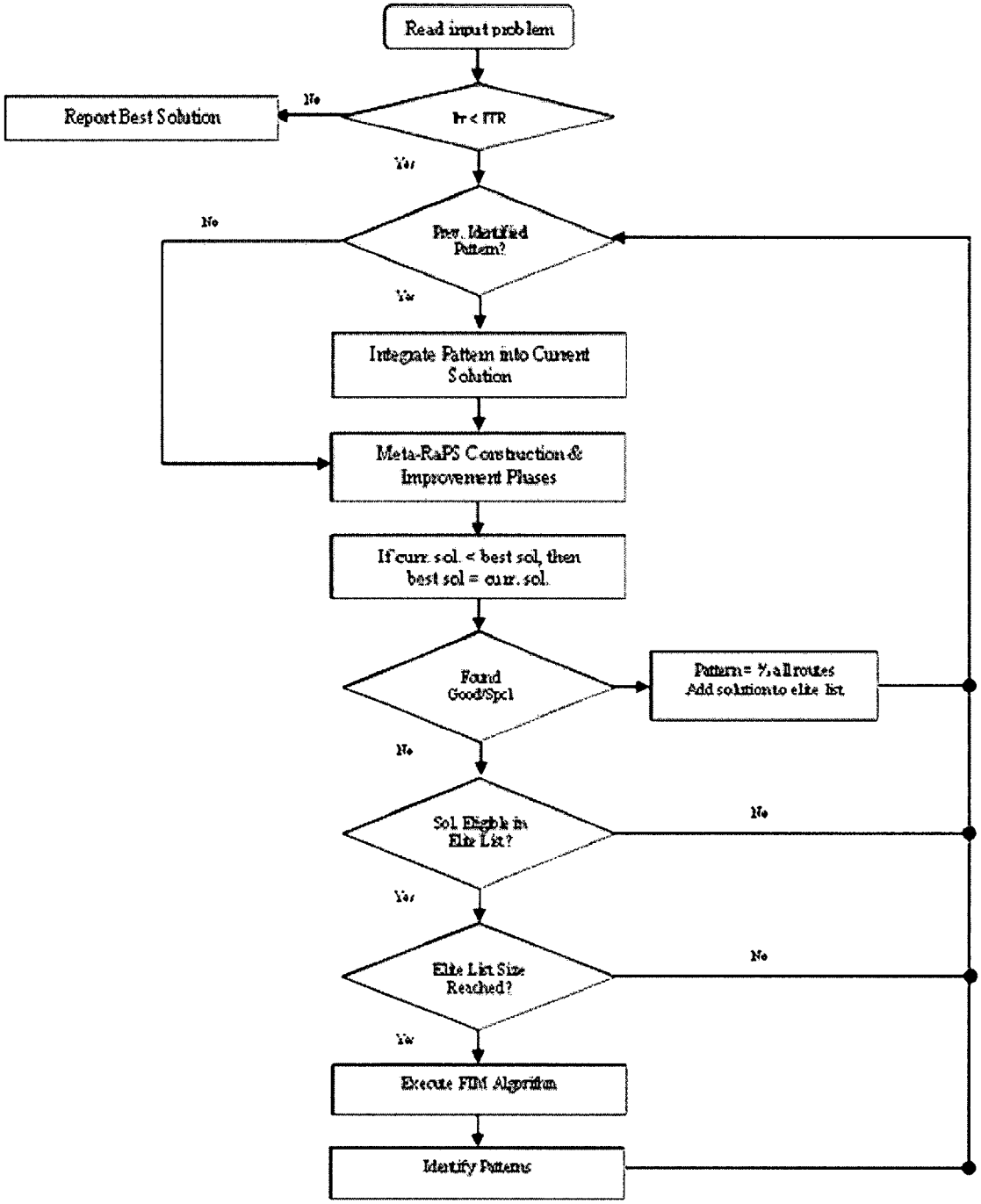


Figure 26 Update MR FIM Framework

The performance of the updated MR FIM is illustrated by showcasing the different approaches to identifying knowledge to pass to future Meta-RaPS iterations:

1- MR FIM only: this approach excludes reacting to finding very good, rare solution. A kick-out based on moving average is employed.
2- MR with special solutions only: this approach excludes FIM algorithm. If a very good, rare solution is found, then half of its routes will be passed to future iterations. A reactive kick-out mechanism is employed.
3- MR FIM including reacting to special solutions: this approach combines the previous approaches. Both kick-out mechanisms are employed.

Similar to previous summary results table, the values included are:

1- The number of times the trail outperformed the original MR with learning.
2- The number of times the trial was on average better than the original MR.
3- Average deviation values, which is based on the best results produced by the MR FIM trial and spans over all the input problems. The average deviation for the original MR for the CMT input problems is 4.15

The "number of times" is relative to the number of input problems. In the case of CMT problems: maximum "number of times" is 14. Note that FIM was executed in the MR FIM with kick-out trials. However, none of the iterations that made use of FIM pattern outperformed the performance of MR.

Trial	MR FIM outperforming original MR w/ learning	MR FIM avg. better than original MR	Avg. Deviation from BKV
MR FIM + Kick-out	0	0	3.79
MR w/ Special Solutions Only + Kick-out	8	0	4.30
MR FIM + Special Solutions + Kick-out	9	0	3.933

Table 24 Performance of Knowledge Approaches in MR FIM

The following table shows the details behind the summary table.

VRPNC	BKV	MR				MR FIM				MR w/ Special Solutions Only				MR FIM			
		Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%	Best	Best Dev%	Avg.	Avg. Dev%
1	524.61	544.53	3.80	552.05	5.23	544.02	3.70	580.83	10.72	539.32	2.80	572.81	9.19	537.38	2.43	569.98	8.65
2	835.26	845.75	1.26	866.89	3.79	849.89	1.75	899.81	7.73	862.40	3.25	891.27	6.71	858.16	2.74	901.72	7.96
3	826.14	858.32	3.90	865.27	4.74	852.85	3.23	898.41	8.75	850.78	2.98	894.39	8.26	851.60	3.08	899.95	8.93
4	1028.42	1089.35	5.92	1093.42	6.32	1076.74	4.70	1143.64	11.20	1072.24	4.26	1143.81	11.22	1069.08	3.95	1136.86	10.54
5	1291.29	1389.18	7.58	1397.87	8.25	1389.09	7.57	1457.99	12.91	1389.69	7.62	1457.38	12.86	1385.91	7.33	1456.37	12.78
6	555.43	574.52	3.44	582.34	4.84	557.89	0.44	615.77	10.86	576.14	3.73	606.75	9.24	573.89	3.32	607.22	9.32
7	909.68	944.74	3.85	952.59	4.72	935.62	2.85	985.03	8.28	940.99	3.44	978.29	7.54	943.33	3.70	983.66	8.13
8	865.94	927.19	7.07	936.00	8.09	902.75	4.25	974.18	12.50	905.30	4.55	972.11	12.26	885.92	2.31	962.59	11.16
9	1162.55	1243.96	7.00	1252.23	7.71	1222.89	5.19	1318.14	13.38	1237.06	6.41	1315.76	13.18	1227.03	5.55	1302.23	12.02
10	1395.85	1531.73	9.73	1543.66	10.59	1514.53	8.50	1620.72	16.11	1524.40	9.21	1622.65	16.25	1529.13	9.55	1621.46	16.16
11	1042.11	1058.81	1.60	1065.67	2.26	1099.97	5.55	1165.08	11.80	1104.32	5.97	1163.06	11.61	1088.81	4.48	1159.17	11.23
12	819.56	826.36	0.83	829.65	1.23	825.58	0.73	858.72	4.78	824.62	0.62	856.57	4.52	833.39	1.69	865.03	5.55
13	1541.14	1570.38	1.90	1579.15	2.47	1602.30	3.97	1693.52	9.89	1611.34	4.55	1727.17	12.07	1603.20	4.03	1729.37	12.21
14	866.37	868.50	0.25	871.07	0.54	871.89	0.64	911.65	5.23	873.84	0.86	909.00	4.92	874.27	0.91	918.67	6.04

Table 25 MR FIM Results

8.8 Algorithm Comparison

To demonstrate the performance of the various algorithms, a comparison is done between:

1- The original Meta-RaPS (without any hybridization) and both MR IDT and MR FIM with the goal of showcasing the impact of hybridization.
2- MR, MR IDT, and MR FIM with other applications in the literature that used Meta-RaPS to solve similar VRP instances.

Comparing the original Meta-RaPS with the hybridized ones aims at examining the research hypothesis of behind this effort specifying that a hybridized Meta-RaPS outperforms the original Meta-RaPS. The statistical hypotheses used in the experiments are based on paired t-test since the same input problems are solved with the original Meta-RaPS and with the hybridized ones.

The null hypothesis states that there is no difference in performance when a problem is solved with the original Meta-RaPS or with the hybridized Meta-RaPS.

$$H_0 : \mu_d = 0 \tag{12}$$

$$H_a : \mu_d \neq 0 \tag{13}$$

where d is the difference between paired values. The following tables show the paired t-test results for the best performing trails in both MR IDT and MR FIM. Paired t-tests are performed using the best deviation from BKV for the original MR and the hybrid MR algorithms. For MR IDT, the difference between means is 0.97 and the two-tail P value is 0.00735 indicating that that the statistical difference between the samples is significant. On the other hand, the difference between the mean of MR and the mean of MR FIM is 0.22 and the two-tail P value is 0.674167 indicating that there is no statistical difference between the samples.

	Original MR (Best Dev%)	*MR IDT (Best Dev%)*
Mean	4.152143	3.18071429
Variance	8.431664	4.68273022
Observations	14	14
Pearson Correlation	0.939094	
Hypothesized Mean Difference	0	
df	13	
t Stat	3.172461	
P(T<=t) one-tail	0.003673	
t Critical one-tail	1.770933	
P(T<=t) two-tail	0.007347	
t Critical two-tail	2.160369	

Table 26 MR IDT Approach 1 - Paired t-test Results (Best Values)

	Original MR (Best Dev%)	*MR FIM w. Spcl (Best Dev%)*
Mean	4.15203	3.933664
Variance	8.445334	5.200103
Observations	14	14
Pearson Correlation	0.757258	
Hypothesized Mean Difference	0	
Df	13	
t Stat	0.4301	
P(T<=t) one-tail	0.337083	
t Critical one-tail	1.770933	
P(T<=t) two-tail	0.674167	
t Critical two-tail	2.160369	

Table 27 MR FIM with Special Solution - Paired t-test Results (Best Values)

When comparing the performance of MR, MR IDT, and MR FIM to other metaheuristics, the focus was on top performing metaheuristics as listed below.

Metaheuristic	Hybridization Approach	CMT BKV Dev%	GWKC BKV Dev%
TABUROUTE (Gendreau et al., 1994)	Tabu search	0.05	N/A
AGES (Mester and Bräysy, 2005)	Genetic Algorithm, Guided Local Search	0.03	0.11
D-Ants (Reimann et al., 2004)	Ant Systems	N/A	0.71
BoneRoute (Tarantilis and Kiranoudis, 2002)	Tabu search, Adaptive memory	0.23	0.76 *avg over 8 not 20 input*
SEPAS (Tarantilis, 2005)	Tabu search, Adaptive memory	0.18	N/A
GRELS (Prins, 2009)	GRASP, ELS		
MPNS-GRAPS (Marinakis, 2012)	GRASP, Multiple reactive methods		
GRASPVRP (Layeb, 2013)	GRASP, SA		
GRASP w/ evolutionary path-relinking (Usberti, et al., 2013)	GRASP, Multiple reactive methods		

Table 28 Top Performing Hybrid Metaheuristics Solving VRP

The table below shows the performance of GRASP due to its similarity to Meta-RaPS. The algorithms are: GRELS (Prins, 2009) hybridized GRASP with Evolutionary Local Search metaheuristic, MPNS-GRAPS (Marinakis, 2012) incorporates multiple reactive mechanisms into GRASP, and GRASPVRP (Layeb, 2013) combines GRASP with Simulated Annealing.

CMT Input	GRELS Dev %	MPNS-GRAPS Dev %	GRASPVRP Dev%
VRPNC 1	0.00	0.00	0.00
VRPNC 2	0.00	0.14	0.11
VRPNC 3	0.00	0.00	0.00
VRPNC 4	0.10	0.37	0.33
VRPNC 5	0.22	1.77	1.34
VRPNC 6	0.00	0.00	0.00
VRPNC 7	0.00	0.00	0.53
VRPNC 8	0.00	0.00	1.03
VRPNC 9	0.00	1.14	1.32
VRPNC 10	0.40	1.16	1.20
VRPNC 11	0.00	0.00	0.00
VRPNC 12	0.00	0.19	0.00
VRPNC 13	0.28	0.48	0.44
VRPNC 14	0.00	0.26	0.00
Avg	**0.07**	**0.39**	**0.45**

Table 29 Performance of GRASP-based Metaheuristics

There are no other publications that used Meta-RaPS to solve the full set of CMT instances. None of the algorithms below attempted the larger GWKC benchmark problems. The following table shows the results of applying Meta-RaPS to solve different VRP benchmark input problems.

Source	Metaheuristic	Heuristic	Tuning	Input (name, #nodes)	%Dev BKV
(Moraga, 2002)	Meta-Raps	C&W	P%=90 R%=10 Itr=5000	(Eil, 51)	5.64
	Meta-Raps	C&W	P%=60 R%=10 Itr=5000	(Eil, 76)	2.64
	Meta-Raps	C&W	P%=80 R%=10 Itr=5000	(Eil, 101)	3.33
(Song, 2005)	Meta-Raps	C&W + improvement *repeated for 500 iterations*	5 repetitions P%=20 R%=15 I%=5 Improvement itr = 500 MR Itr=500	(Eil, 51)	2.4
	Meta-Raps	C&W + improvement *repeated for 500 iterations*	5 repetitions P%=20 R%=5 I%=5 Improvement itr = 500 MR Itr=500	(Eil, 76)	1.0
	Meta-Raps	C&W+ improvement *repeated for 100 iterations*	5 repetitions P%=80 R%=5 I%=5 Improvement itr = 500 MR Itr=500	(Eil, 101)	3.31
(Mirza, 2011)	Meta-Raps	C&W + adjacent pairwise	%p = 80, %r = 20, %i=100 MR itr= 2000	(VRPNC1, 50)	3.12
				(VRPNC2, 75)	3.80
				(VRPNC3, 100)	5.19
				(VRPNC4, 150)	7.45
				(VRPNC11, 120)	2.77
				(VRPNC12, 100)	1.03
(Mirza, 2011)	Meta-Raps + Neural network Hebbian learning	C&W + adjacent pairwise + neural network learning	%p = 80, %r = 20, %i=100 MR itr= 2000	(VRPNC1, 50)	3.12
				(VRPNC2, 75)	4.04
				(VRPNC3, 100)	3.86
				(VRPNC4, 150)	8.32
				(VRPNC11, 120)	2.68
				(VRPNC12, 100)	1.76

Table 30 Performance of Meta-RaPS Solving VRP

The tables above show that the performance of published Meta-RaPS does not compete with the top performing metaheuristics or GRASP-based algorithms that solved the classic VRP instances. The table below compares the performance of the published Meta-RaPS (Mirza, 2011) used to solving CMT input with algorithms developed in this effort (the last four columns show the results from the algorithms developed during this effort).

CMT Input	**MR** *2000 itr*	**MR Neural Network** *2000 itr*	**MR** *500 itr*	**MR IDT** *500 itr*	**MR FIM** *500 itr*
VRPNC1	3.12	3.12	3.8	1.68	2.43
VRPNC 2	3.80	4.04	1.26	2.33	2.74
VRPNC 3	5.19	3.86	3.9	3.24	3.08
VRPNC 4	7.45	8.32	5.92	4.64	3.95
VRPNC 11	2.77	2.68	1.6	2.1	4.48
VRPNC 12	1.03	1.76	0.83	0.43	1.69
Avg	**3.96**	**3.89**	**2.88**	**2.40**	**3.06**

Table 31 Performance Relative to Existing MR Implementation

CHAPTER 9

CONCLUSIONS

This effort was conducted around three interests: contributing to the efforts that aim at evolving Meta-RaPS, incorporating machine learning to metaheuristics, and solving classic VRP benchmark instances using Meta-RaPS. However, without the simplicity and adaptability of the classic Meta-RaPS algorithm, transforming these interests into a research effort would have been more challenging. Thus, perusing hybridization while preserving the simplicity of Meta-RaPS became another core interest. These interests combined formulate the research hypothesis of this study, which is to improve the performance of Meta-RaPS by hybridizing it with machine learning algorithms and demonstrate the performance by solving VRP instances.

The main contribution of this work lays in the employment of machine learning algorithms to enhance a metaheuristic algorithm. This contribution departs from the normal use of metaheuristic algorithms to enhance the performance of machine learning algorithms. The scope of this hybridization included two machine learning algorithms: Inductive Decision Tree and Frequent Itemset Mining portion of Association Rules. The selected machine learning algorithms conducted different learning approaches supervised and unsupervised respectively. The resulting hybridized Meta-RaPS algorithms are: MR IDT and MR FIM. Both hybridized algorithms followed the High-level Relay Hybridization classes, part of the hierarchical hybridization taxonomy. Additionally, when categorized based on the flat hybridization taxonomy, MR IDT and MR FIM are heterogeneous hybridized algorithms both applied to the global search domain with a general/same problem to solve. These hybridization approaches are aligned with the interest of continuing to keep Meta-RaPS relatively simple. Though both algorithms belong to the same hybridization classes (based on both hierarchical and flat hybridization taxonomies), the roles played by each machine learning algorithm were different. The IDT aimed at effectively guiding MR by performing online parameter control. FIM increased the efficiency of managing the search space by identifying

patterns and forgoing MR multi-start. Though hybridizing two or more algorithms has a negative impact of the simplicity enjoyed by the original/pure algorithms, both MR IDT and MR FIM are relatively simple and problem independent especially when compared with other metaheuristics (non-hybridized ones) used in solving VRP.

The lessons learned from hybridizing MR with IDT and FIM are reflected in the performance of MR IDT and MR FIM. MR IDT helped improve the performance when compared to the original Meta-RaPS due to the known, critical impact metaheuristics' parameters have on their performance. Fixing the values of these parameters while solving different problems (in this case VRP problems varied in size, the layout of the coordinates, and the constraints: truck capacity with and without route time) has a negative impact on performance as demonstrated in the performance of the original MR. Though multiple trials were conducted, the simple idea behind taking advantage of the root node of the tree proved to be sufficient enough to show improvement in performance. It is worth noting that MR IDT helped in guiding Meta-RaPS without impacting its multi-start approach for constructing solutions. In other words, MR IDT kept one of MR core features, which is allowing for less than best (highest-priority) moves to be considered when constructing a solution. This was not the case for MR FIM, which intentionally impacted MR multi-start by fixing coordinates passed as patterns from previous iterations. Despite including mechanisms to balance the intensification resulting from FIM patterns, MR FIM did not have a statistically significant improvement over the performance of Meta-RaPS.

The potential for future studies in this field are abundant as many combinations of algorithms can be examined. A systematic study aiming at showcasing how machine learning algorithms can be hybridized with Meta-RaPS or other metaheuristics at various hybridization levels based on the hybridization taxonomy would be an interesting study that adds to this not-well studied field.

Future work related to MR IDT may evaluate other classification algorithms. Another use of MR IDT can be to tune MR parameters offline to allow for MR to be

combined with other algorithms (possible MR FIM with the following sequence: MR IDT followed by MR FIM). MR FIM can benefit from more diversification methods to help maintain the randomness of Meta-RaPS such as methods to modify the priority of the solution being constructed. Both algorithms can also be evaluated with more robust heuristics.

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993) "Mining association rules between sets of items in large databases", Proceedings of the ACM SIGMOD Conference on Management of Data, pp.207 -216

Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez Pérez, M., Roli, A., & Sampels, M. (eds), Proceedings of HM 2006 – Third International Workshop on Hybrid Metaheuristics, volume 4030 of Lecture Notes in Computer Science, pp.1–12. Springer Verlag, Berlin, Germany, 2006

Alpaydin, E. (2010). *Introduction to machine learning.* Cambridge, MA: MIT Press.

Arcus, A.L. (1965). COMSOAL: A computer method of sequencing operations for assembly line. *International Journal of Production Research. 4*(4), 25-32

Arin, A. (2012). Incorporating memory and learning mechanisms into Meta-RaPS. Doctoral thesis, Old Dominion University, Norfolk, VA

Belenguer, J., Martinez, M., & Mota, E. (2000). A lower bound for the split delivery vehicle routing problem, *Operations Research, 48*, pp. 801-810.

Bianco, L., Dell'Olmo, P., Giordani, S., Sharaiha, Y.M., & Beasley, J. E. (1999). Minimizing total completion time subject to release dates and sequence dependent processing times, *Annals of Operations Research, 86*, pp. 393–415.

Blum, C. (2005). Ant colony optimization: Introduction and trends. *Physics of Life Reviews, 2*, pp. 353 – 373

Blum, C., Puchinger, J., Raid,l G.R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*(6), pp. 4135-4151

Blum, C., Blesa Aguilera, M. J., Roli, A., & M. Sampels, (eds). (2008). Hybrid metaheuristics – An emerging approach to optimization", vol. 114 of Studies in Computational Intelligence. City: Springer

Blum, C., Puchinger, J., Raidl, G.R., & Roli, A. (2010). A brief survey on hybrid metaheuristics", In B. Filipič & J. Šilc (eds.), Proceedings of BIOMA 2010—4th International Conference on Bio-Inspired Optimization Methods and their Applications, Jozef Stefan Institute, Ljubljana, Slovenia, 2010, pp. 3–18

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Computing Surveys, 35*(3), pp. 268–308.

Bramel, J., & Simchi-Levi D. (1995). A location based heuristic for general routing problems. *Operations Research, 43*, pp. 649-660

Chen, C.C., Yih, Y., & Wu, Y.C. (1999). Auto-bias selection for learning-based scheduling systems", *International Journal of Production Research, 37*(9), pp. 1987–2002

Chen, M.S., Han, J., & Yu, P.S. (1996). Data mining: An overview from database perspective. *Transactions on Knowledge and Data Engineering, 8*(6) pp. 866 – 883

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem, in N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), *Combinatorial Optimization* (pp. 315-338), Chichester: Wiley.

Chvatal, V. (1979). A greedy heuristic for the set covering problem. *Mathematics of Operations Research, 4*, pp. 233-235

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicle from a central depot to a number of delivery points. *Operations Research, 12*, pp. 568-581.

Cordeau, J. F., & Laporte G. (2001). A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFORM*, 39, pp. 292-298

Dantzig, G.B. & Ramser, H.H. (1959). The truck dispatching problem. *Management Science, 6*(1), pp. 80-91

de Lima, F.C., de Melo, J.D., & Neto, A.D.D. (2008). Using the Q-learning algorithm in the constructive phase of the GRASP and reactive GRASP metaheuristics, *International Joint Conference on Neural Networks*, pp. 4169-4176

den Besten, M., Stutzle, T., & Dorigo, M. (2001). Design of iterated local search algorithms: An example application to the single machine total weighted tardiness problem. In Proceedings of EvoWorkshops, LNCS. Springer, Berlin, pp. 441-451

DePuy, G.W., Whitehouse, G.E., & Moraga, R.J. (2001). Meta-RaPS: A simple and efficient approach for solving combinatorial problems. 29th International Conference on Computers and Industrial Engineering, November 1-3, Montreal, Canada, pp. 644-649

DePuy, G.W., Moraga, R.J., & Whitehouse G.E. (2005). Meta-RaPS: A simple and effective approach for solving the traveling salesman problem. *Transportation Research Part E, 41*, pp. 115–130

DePuy, G.W., & Whitehouse G.E. (2000). Applying the COMSOAL computer heuristic to the constrained resource allocation problem. *Computers & Industrial Engineering, 38*(3), pp. 413-422

DePuy, G.W., & Whitehouse, G.E. (2001). A simple and effective heuristic for the resource constrained project scheduling problem. *International Journal of Production Research, 39*(14), pp. 3275-3287

DePuy, G.W., Whitehouse G.E., Schulman, N.T., AlMazid, A.M., & Moraga R.J. (2000a). Modification of the COMSOAL approach to solve various combinatorial problems. Proceedings of the 4th International Conference on Engineering Design and Automation, July 30 – August 2, Orlando, Florida, pp. 349-354

DePuy, G.W., Whitehouse, G.E., & Tabler, D.K. (2000b). Using priority rules within the COMSOAL heuristic to solve the resource allocation problem. Proceedings of the 4th International Conference on Engineering Design and Automation, July 30 – August 2, Orlando, Florida, pp. 87-93

Dorigo, M., Bonabeau, E., & Theraulaz, G. (2000). Ant algorithms and stigmergy, *Future Generation Computer Systems, 16*, pp. 851- 871

Dror, M., & Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science, 23*, pp. 141- 145.

Fayyad, U.M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: An overview, Advances in knowledge discovery and data mining. Menlo Park, CA: American Association for Artificial Intelligence.

Feo T.A., & Resende M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*, pp. 109- 133

Feo, T.A., & Resende, M.G.C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters8*, pp. 67–71

Filipec, M., Skrlec, D., & Krajcar, S. (1997). Darwin meets computers: New approach to multiple depot capacitated vehicle routing problem. *IEEE International Conference Systems, 1Man & Cybernetics*, Orlando, FL, pp. 421–426

Fisher, M.L., & Jaikumar, R. (1981).A generalized assignment heuristic for vehicle routing. *Networks, 11*, pp. 109-124.

Fogel, L. J. (1962). Toward inductive inference automata. Proceedings of the International Federation for Information Processing Congress, Munich, pp. 395 – 399.

Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. *Opl Res., 27*, pp. 307-384

Gaudioso, M., & Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transport Sci, 26*, pp. 86–92.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science, 40*(10), pp. 1276–1290.

Gendreau, M., & Potvin, J. (2010). Handbook of Metaheuristics. New York: Springer.

Gillett, B., & Miller, L. (1974). A heuristic for the vehicle dispatching problem. *Operations Research, 22*, pp. 340-349.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research, 13* (5), pp. 533–549.

Golden, B. L., & Wong, R. T. (1981). Capacitated Arc Routing Problems. *Networks, 11*, pp. 305–315

Golden, B. L., & Stewart, W. R. (1985). Empirical analysis of heuristics". In The Traveling Salesman Problem (pp. 207-249). Chichester: John Wiley & Sons..

Golden, B. L., Wasil, E.A., & Kelly, J.P., & Chao I. M. (1998). Metaheuristics in Vehicle Routing In T. G. Crainic and G. Laporte, eds, Fleet Management and Logistics (pp. 33-56). Boston: Kluwer.

Golden, B.L., & Assad, A.A. (1988). Vehicle Routing: Methods and studies. North-Holland: Amsterdam.

Gomes, L. M., Diniz, V. B., & Martinhon, C. A. (2000). An hybrid GRASP + VNS Metaheuristic for the Prize-Collecting Traveling Salesman Problem. *Simposio Brasileiro de 'pesquisa Operacional (SBPO), 32*, pp. 1657–1665

Goodman, M.D., Dowsland, K.A., & Thompson, J.M. (2009). A grasp-knapsack hybrid for a nurse-scheduling problem. *J Heuristics,15*, pp. 351–379

Hancerliogullari, & G., Rabadi, G. (2012). Approximate Algorithms for Aircraft Rescheduling Problems Under Operation Disruptions. *INFORMS 2012, Phoenix, AZ (October 14-17)*

Hepdogan, S., Moraga, R.J, DePuy, G.W., & Whitehouse, G.E. (2009). A Meta-RaPS Solution for the Early/Tardy Single Machine Scheduling Problem. *International Journal of Production Research, 47*(7), pp. 1717-1732

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM, 3*, pp. 297–314.

Jourdan, L., Dhaenens, C., & Talbi, E.G. (2006). Using data mining techniques to help metaheuristics: A short survey. Hybrid Metaheuristics (HM'2006), Gran Canaria, Spain, LNCS vol. 4030, pp. 57–69

Kennedy, J., & Eberhart, R.C. (2001). Swarm intelligence. San Francisco, CA: Morgan Kaufmann.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science new series, 220*, pp.671 -680

Kotsiantis, S. B., (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica, 31*, pp. 249-268

Koza, J. R. (1992). Genetic programming. Cambridge, MA: MIT Press.

Lacomme, P., Prins, C., & Sevaux, M. (2003). Multi-objective capacitated arc routing problem. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., & Thiele, L. (eds.), *Evolutionary multi-criterion optimization* (Proceedings of EMO 2003, Faro, Portugal). Lecture Notes in Computer Science, Berlin: Springer, 2632, pp. 550-564

Lan, G., & DePuy, G.W. (2006). On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the Set Covering Problem. *Computers & Industrial Engineering,51*, pp. 362-374

Lan, G., DePuy, G.W., & Whitehouse, G.E. (2007). An Effective and Simple Heuristic for the Set Covering Problem. *European Journal of Operational Research, Vol. 176* (3), pp. 1387-1403

Laporte, G. (1992). The Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*, pp. 345-358

Laporte, G. (2007) . What you should know about the vehicle routing problem. *Naval Res Logist, 54,* pp. 811–819

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43* (4), pp. 408–416

Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. In S. Martello, G. Laporte, M. Minoux & C. Ribeiro (eds.), *Surveys in combinatorial optimization* (pp. 147-184). Amsterdam: North-Holland

Layeb, A., Ammi M., & Chikhi A. (2013). A GRASP algorithm based on new randomized heuristics for vehicle routing problem. *CIT, 21*(1), pp. 37-48

Lessmann, S., Caserta, M., & Arango, I. M. (2011). Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Syst. Appl., 38*(10), pp. 12826-12838

Lourenco, H. R., Martin, O., & Stutzle, T. (2002). Iterated local search, In Glover, F., & G. Kochenberger (eds.), *Handbook of metaheuristics. International Series in Operations Research and Management Science, 57*. Amsterdam: Kluwer Academic Publishers, pp. 321–353.

Marinakis, Y. (2012). Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem. *Expert Systems with Applications, 39*(8), pp. 6807-6815

Mester, D. I., & Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research, 32*, pp.1593-1614

Mirza, A. A. B. (2011). Improved Meta-RaPS approach with learning concepts for solving capacitated vehicle routing problem. Master's thesis, Northern Illinois University, DeKalb, IL

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Ressearch, 24* (11), pp. 1097 – 1100

Moraga, R.J. (2002). Meta-RaPS an effective solution approach for combinatorial problems. Doctoral thesis, University of Central Florida, Orlando, FL

Moraga, R.J., DePuy, G.W., & Whitehouse, G.E. (2005). Meta-Raps approach for the 0-1 multidimensional knapsack problem. *Computers & Industrial Engineering, Vol., 48*(1), pp. 83-96

Park, S.C., Paman, N., & Shaw, M.J. (1997). Adaptive scheduling in dynamic flexible manufacturing systems: A dynamic rule selection approach. *IEEE transactions robotics automation, 13*, pp. 486–502

Plastino, A., Fuchshuber, R., Martins, S. D. L., Freitas, A. A., & Salhi, S. (2011). A hybrid data mining metaheuristic for the p-median problem., *Statistical Analysis and Data Mining, 4*(3), pp. 313-335

Prins, C., (2009). A GRASP x evolutionary local search hybrid for the vehicle routing problem. Bio-inspired algorithms for the vehicle routing problem. In F.B. Pereira & J. Tavares (eds.), *Studies in Computational Intelligence*. Dordecht: Springer, pp. 35–53

Psaraftis, H.N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of Operations Research, 61*, pp. 143–164.

Quinlan, J.R., (1993). *C4.5: Programs for machine learning*. Los Altos, CA: Morgan Kaufmann,

Rabadi, G., Moraga, R., & Al-Salem, A (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times., *Journal of Intelligent Manufacturing,17*, pp. 85-97

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In F. Almeida, M. Blesa, C. Blum, J. M. Moreno, M. Pérez, A. Roli, & M. Sampels (eds.), *Proceedings of HM 2006 – 3rd International Workshop on Hybrid Metaheuristics*, vol. 4030 of *Lecture Notes in Computer Science*, pp. 1–12. Berlin: Springer-Verlag.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem, Translation 1122, Royal Aircraft Establishment Library

Reimann, M., Doerner, K., & Hartl, R.F. (2004). D-Ants: Savings based ants divide and conquer the VRP. *Computers and Operations Research, 31*(4), pp. 563 – 591

Reinelt, G., & Theis, D. R. (2005). Transformation of facets of the general routing problem polytope. *SIAM Journal on Optimization, 16*(1), pp. 220-234

Renaud, J., Boctor, F., & Laporte, G. (1996). An improved petal heuristic for the vehicle routeing problem. *Journal of the Operational Research Society, 47*(2), pp. 329–336

Ribeiro, M., Plastino, A., & Martins, S. (2006). Hybridization of grasp metaheuristic with data mining techniques. *Special Issue on Hybrid Metaheuristic of the Journal of Mathematical Modelling and Algorithms, 5*(1): pp. 23–41

Ribeiro, M., Trindade, V., Plastino, A., & Martins S. (2004). Hybridization of GRASP metaheuristic with data mining techniques. In *Workshop on Hybrid metaheuristics 16th European Conference on Artificial Intelligence* (ECAI), pp. 69–78

Ryan, D., Hjorring, C., & Glover, F. (1993). Extensions of the petal method for vehicle routeing. *The Journal of the Operational Research Society, 44*(3), pp. 289–296

Santos, L. F., Ribeiro, M. H., Plastino, A., & Martins S. L. (2005). A hybrid GRASP with data mining for the maximum diversity problem. In LNCS 3636, *Hybrid Metaheuristic*, pp. 116–128

Santos, H. G., Ochi, L. S., Marinho, E. H., & Drummond, L. M. D. A. (2006). Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing, 70*(1), pp. 70-77

Savelsbergh, M., & Sol, M. (1995). The general pickup and delivery problem. *Transportation Science, 29* (1), pp. 17–29

Shannon, C. (1948). A mathematical theory of communication. *Bell System Tech. J., 27*, pp. 379-423

Solomon, M. M. (1987).Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*(2), pp. 254-265

Song, J., (2005). Meta-heuristics for the vehicle routing problem. Doctoral thesis, University of Louisville, Louisville, Kentucky

Stuetzle, T. (1998). Local search algorithms for combinatorial problems --- Analysis, improvements, and new applications. Doctoral thesis, Darmstadt University of Technology, Darmstadt, Germany

Taillard, E.D., Gambardella, L.M, Gendreau, M., & Potvin, J.Y., (2001). Adaptive memory programming, A unified view of metaheuristics. *European Journal of Operational Research, 135*, pp. 1-16

Taillard, E.D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks, 23*(8), pp. 661–673

Talbi, E. G. (2009). Metaheuristics, from design to implementation. New Jersey: John Wiley & Sons, Inc.,.

Talbi, E.G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics, 8*(5), pp. 541–564

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research, 32*, pp. 2309-2327

Tarantilis, C. D., & Kiranoudis, C. T. (2002). BoneRoute: An adaptive memory-based method for effective fleet management. *Annals Operations Research, 115*(1), pp. 227-241

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. (2004). A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research, 152*(1), pp. 148-158

Tarantilis, C., Ioannou, G., Kiranoudis, C., & Prastacos, G. (2005). Solving the open vehicle routing problem via a single parameter metaheuristic algorithm. *Journal of the Operational Research Society 56*(5), pp. 588–596.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem (monographs on discrete mathematics and applications).* Philadelphia: SIAM

Ursani, Z. (2009). Localized genetic algorithm for the vehicle routing problem. Doctoral thesis, University of New South Wales at Australian Defence Force Academy. Retrieved from: http://unsworks.unsw.edu.au/vital/access/manager/Repository/unsworks (2009) 7240

Usberti, F. L., França, P. M., & França, A. L. M. (2013). GRASP with evolutionary path-relinking for the capacitated arc routing problem. *Computers & Operations Research, 40*(12), pp. 3206-3217

Villegas, J.G., Prins, C., Prodhon, C., Medaglia, A.L., & Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers and Operations Research,33*, pp. 1319-1334

Voudouris, C. (1997). Guided local search for combinatorial optimisation problems. Doctoral thesis, University of Essex, Colchester, UK

Wolpert, D.H., & Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, pp. 67-82

VITA

Fatemah Al-Duoli received both her Bachelors and Masters of Science degrees in Computer Engineering from Old Dominion University in 2003 and 2005. She worked as a teaching assistant for a variety of courses during both degrees. She taught introductory logic and circuit courses at ECPI, Newport News, VA. Her research interests include metaheurisitcs, optimization methods, machine learning, modeling and simulation, and scheduling. Mrs. Al-Duoli worked as a project engineer at Luna Innovation, Hampton, VA between 2005 and 2008. She has been working as Principle Systems engineering at Rockwell Collins ARINC since 2008.