2006

Meta-raps: Parameter Setting And New Applications

Seyhun Hepdogan
University of Central Florida

Meta-RaPS: PARAMETER SETTING AND NEW APPLICATIONS

by

SEYHUN HEPDOGAN

B.S. Middle East Technical University, 2000
M.S. Middle East Technical University, 2001
M.S. University of Central Florida, 2004

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2006

Major Professors: Gary E. Whitehouse
Gail W. DePuy

ABSTRACT

Recently meta-heuristics have become a popular solution methodology, in terms of both research and application, for solving combinatorial optimization problems. Meta-heuristic methods guide simple heuristics or priority rules designed to solve a particular problem. Meta-heuristics enhance these simple heuristics by using a higher level strategy. The advantage of using meta-heuristics over conventional optimization methods is meta-heuristics are able to find good (near optimal) solutions within a reasonable computation time. Investigating this line of research is justified because in most practical cases with medium to large scale problems, the use of meta-heuristics is necessary to be able to find a solution in a reasonable time.

The specific meta-heuristic studied in this research is, Meta-RaPS; Meta-heuristic for Randomized Priority Search which is developed by DePuy and Whitehouse in 2001. Meta-RaPS is a generic, high level strategy used to modify greedy algorithms based on the insertion of a random element (Moraga, 2002). To date, Meta-RaPS had been applied to different types of combinatorial optimization problems and achieved comparable solution performance to other meta-heuristic techniques.

The specific problem studied in this dissertation is parameter setting of Meta-RaPS. The topic of parameter setting for meta-heuristics has not been extensively studied in the literature. Although the parameter setting method devised in this dissertation is used primarily on Meta-RaPS, it is applicable to any meta-heuristic's parameter setting problem. This dissertation not only enhances the power of Meta-RaPS by parameter tuning but also it introduces a robust parameter selection technique with wide-spread utility for many meta-heuristics.

Because the distribution of solution values generated by meta-heuristics for combinatorial optimization problems is not normal, the current parameter setting techniques which employ a parametric approach based on the assumption of normality may not be appropriate. The proposed method is Non-parametric Based Genetic Algorithms. Based on statistical tests, the Non-parametric Based Genetic Algorithms (NPGA) is able to enhance the solution quality of Meta-RaPS more than any other parameter setting procedures benchmarked in this research. NPGA sets the best parameter settings, of all the methods studied, for 38 of the 41 Early/Tardy Single Machine Scheduling with Common Due Date and Sequence-Dependent Setup Time (ETP) problems and 50 of the 54 0-1 Multidimensional Knapsack Problems (0-1 MKP).

In addition to the parameter setting procedure discussed, this dissertation provides two Meta-RaPS combinatorial optimization problem applications, the 0-1 MKP, and the ETP. For the ETP problem, the Meta-RaPS application in this dissertation currently gives the best meta-heuristic solution performance so far in the literature for common ETP test sets. For the large ETP test set, Meta-RaPS provided better solution performance than Simulated Annealing (SA) for 55 of the 60 problems. For the small test set, in all four different small problem sets, the Meta-RaPS solution performance outperformed exiting algorithms in terms of average percent deviation from the optimal solution value. For the 0-1 MKP, the present Meta-RaPS application performs better than the earlier Meta-RaPS applications by other researchers on this problem. The Meta-RaPS 0-1 MKP application presented here has better solution quality than the existing Meta-RaPS application (Moraga, 2005) found in the literature. Meta-RaPS gives 0.75% average percent deviation, from the best known solutions, for the 270 0-1 MKP test problems.

This dissertation is dedicated to my family.

ACKNOWLEDGMENTS

I would like to use the opportunity to thank the people whom helped me throughout my PhD. Study. Despite my efforts to thank these people, I know that I will always feel indebted to them.

I would like to thank God, for his blessings throughout my life. Without the strength he provided at times when I felt lonely, out-of-place and deserted, I know I wouldn't be able to carry on.

Without the support of my family I would have never accomplished this wish of mine. Apart from me and frankly more than me the burden of this dissertation is carried on their shoulders.

I would like to thank my main advisors Dr. Gary E. Whitehouse and Dr. Gail DePuy for continuously supporting me without any complaints throughout my dissertation. I feel blessed to have Dr. Reinaldo Moraga on my dissertation committee for the help and support he provided. I feel extremely lucky to have these three people around me not only for the sake of my dissertation but for being the model individuals whom I may wish to be in future. I also would like to thank my remaining committee members; Dr. Jose Sepúlveda, Dr. Charles Reilly III and Dr. Christopher D. Geiger for enriching my research experience and supporting me.

I also would like to thank all my friends for their extensive help during my studies.

Finally I wish that I would be worthy of all these people who supported me and be able to give back, serve with my work in future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

0-1 MKP	0-1 Multi-dimensional Knapsack Problem
COMSOAL	Computer Method of Sequencing Operations for Assembly Lines
ETP	Early/ Tardy Single Machine Scheduling Problem with a Common Due Date and Sequence Dependent Setup Times
GA	Genetic Algorithms
GAP	Generalized Assignment Problem
GPI	Generalized Pairwise Interchange
GRASP	Greedy Randomized Adaptive Search Procedure
K-W	Kruskal-Wallis Test
LPT	Longest Processing Time
Meta-RaPS	Meta-heuristic for Randomized Priority Search
NN	Neural Networks
NPGA	Non-Parametric Based Genetic Algorithm
RCPSP	Resource Constrained Project Scheduling Problem
RS	Reactive Search
R&S	Ranking and Selection Method
RSM	Response Surface Methodology
SA	Simulated Annealing
SE	Simulated Evolution
SPT	Shortest Processing Time
TS	Tabu Search

TE	Simple Trial-and-error Method
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

CHAPTER 1: INTRODUCTION TO COMBINATORIAL OPTIMIZATION PROBLEMS

1.1 Combinatorial Optimization Problems

Many practical and theoretical problems involve a search of the best configuration, or set of variables, to achieve a goal. Many of these problems can be structured to optimize some decision variables within a set of constraints (Papadimitriou, and Steiglitz, 1982). The formulations of one end of the hierarchy of such problems are given as a general nonlinear programming problem; for example:

where f, g_i and h_j are general functions of the variable The goal of the problem is to optimize the function f by setting the independent variables, $\mathbf{x}$ to a level within the allowable ranges of constraints g_i and h_j. The optimization of $f(x)$ can be a minimization, as shown above, or a maximization problem, depending on the nature and objective of the problem. The principal goal of optimization models is to mathematically express the problem to facilitate a solution method. Solution techniques for many optimization problems are almost always iterative in nature, and their convergence is studied using the mathematics of real analysis.

There are different classes of optimization problems to consider. By placing restrictions on the type of functions under consideration (f, g_i and h_j), and restricting the values that decision variables can take, different classes of optimization problems are formed (Reeves, 1993). If f is convex, g_i concave and h_j is linear, then the optimization problem is called a convex programming problem. This type of problem is convenient in the sense that local optimality implies global optimality. In a more general setting, when all f, g_i and h_j are linear the problem is called a linear programming problem.

Optimization problems are typically divided into two categories: those with continuous variables and those with discrete variables, which are also called combinatorial problems. In combinatorial problems, one is looking for a set of decision variables from a finite or possibly near-infinite set that satisfies constraints and optimizes one or more objective functions (Papadimitriou, and Steiglitz, 1982).

A wide variety of combinatorial optimization problems are studied in the literature as they have numerous practical applications. The combinatorial optimization problems most often addressed in the literature include Traveling Salesperson Problem (TSP), Quadratic Assignment Problem (QAP), Multiple Knapsack Problem (MKP), Bin Packing Problem, and the Vehicle Routing Problem (VRP).

1.2 Why Heuristics Are Needed

A well-known combinatorial optimization problem that is often cited in literature is the Traveling Salesperson Problem (TSP). In TSP, one is given an integer number n>1 number of cities and the distance between every pair of n cities. The objective of the problem is to find a tour or a closed path that visits each city exactly once and minimizes

total tour length. If the goal is to find the minimal tour by enumeration, one begins by computing the length of all the possible tours and picking the one with the smallest tour length. Since each city can be visited only once, the number of possible tours will be (n-1)! The number of possible routes can be enumerated when n is relatively small. However as the number of cities increases, the possible combinations of tours become impossible to enumerate even with the most advanced computers today. A TSP with 25 cities has 1.55E+25 possible tours which is not a reasonable number for an enumeration approach. TSP is considered a NP-Complete problem for the reason that there is no algorithm that can solve the problem in polynomial time (Reeves, 1993).

Since the enumeration technique seemed impractical for most combinatorial optimization problems, researchers propose simpler procedures for finding optimal solution. The techniques which guarantee optimal solutions, like complete enumeration, are called exact methods. Exact methods try to find a solution more efficiently than complete enumeration. Simplex algorithm (used for linear optimization problems), branch-and-bound and dynamic programming are some examples of exact methods. However, researchers determined that for large problems, most exact methods were not able to find optimal solutions in a reasonable amount of computing time. Although exact methods are usually more efficient than complete enumeration, they are still impractical in terms of computation time for large problems Reeves, 1993).

To be able to solve large problems, researchers began to investigate how the solution time for a problem varies with the size (i.e., number of cities in a TSP instance) of the problem (Reeves, 1993). Exact methods for some problems require a computation effort with polynomial time complexity. Unfortunately, for some other harder problem computational effort required is at a greater magnitude than the polynomial function of

the size of the problem, such as exponential function of the problem size (Reeves, 1993). However, in many practical cases the goal of achieving the optimal solution is not only unrealistic but also unnecessary; finding a "good enough" solution is often considered a success considering that the true optimal solution may never be reached and the optimal solution may not be necessary in practice. To find the "best" solution subject to the time restrictions, researchers devise simple rules of thumbs (i.e., heuristics). The term heuristic can be defined as "a technique which seeks good (i.e., near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is" (Reeves, 1993). It is this need to generate a fast, good enough solution that justifies why heuristics are merited.

1.3 Meta-Heuristics

Heuristics are classified into several broad categories with respect to the approach used to find a solution. Some of these categories are: greedy construction methods, neighborhood search (improvement heuristics), relaxation techniques, partial enumerations, decomposition and partition.

Many heuristics are problem specific. However, there are several general techniques that give high quality solutions over a wide array of problem types. The local neighborhood search strategy is considered to be an important technique for its performance flexibility on various types of problems. Some examples of the local search techniques are λ-optimal heuristics, in which a feasible solution is improved by exchanges (decomposition and reconstruction of a solution), uphill moves in which a

solution is allowed to get worse to for the intention that a local optimal solution to be "climbed out" in order to ultimately reach a better solution.

Despite the utility of these general search strategies, researchers have proposed meta-heuristics to further enhance the solution quality of heuristics. Meta-heuristics are defined as a top-level general strategy that guides other heuristics to search for feasible solutions in domains where the task is particularly hard. In such cases, meta-heuristics have been applied to problems classified as NP-Hard and achieved considerable success (Reeves, 1993). Some examples of modern meta-heuristics are: Meta-RaPS, tabu search, simulated annealing, genetic algorithms, neural networks and greedy randomized adaptive search procedure (GRASP). This research focuses on Meta-RaPS, a meta-heuristic developed by DePuy and Whitehouse (2001). However many of the insights gained in this work can be applied to other meta-heuristics.

Almost all meta-heuristics have decision parameters to be set, and the performance of a meta-heuristic is dependent on the choice of these parameters. There are many techniques available for meta-heuristic parameter selection, but there is no clear consensus on when and why to use a particular parameter selection technique. The main objective of this research is to design a robust parameter setting technique for the meta-heuristic studied in this research, Meta-RaPS. Other objectives of this research are to implement Meta-RaPS for 0-1 MKP and ETP combinatorial optimization problems and achieve comparable or better solution performance against the other existing meta-heuristic applications in the literature for these two combinatorial optimization problems.

This dissertation is organized as follows. In Chapter 2, several common meta-heuristics including the meta-heuristic used in this research, are described. The research objectives are also presented in Chapter 2. In order to best demonstrate the parameter

setting techniques developed in this research, two combinatorial optimization problems are studied in Chapter 3. Chapter 3 also includes the description and results of several modifications made to previous Meta-RaPS 0-1 MKP and Meta-RaPS ETP efforts. The enhanced performance of Meta-RaPS for these two application problems constitutes a contribution of this research in addition to the main focus of this work of parameter selection techniques for meta-heuristics. Parameter setting literature is discussed in Chapter 4 and the problem statement is identified based on the research gap found in the literature search. Chapter 5 provides the description and Meta-RaPS applications of several parameter setting techniques found in literature. Chapter 6 presents the analysis of statistical parameter setting comparisons and shows that this comparison should be done using distribution-free procedures. In Chapter 6, a new parameter setting method, Non-parametric Based Genetic Algorithms, is introduced and compared to the parameter setting methods in Chapter 5. Chapter 7 briefly summarizes the findings and contributions of this research as well as discusses future research directions.

CHAPTER 2: COMMON META-HEURISTICS AND INTRODUCTION TO Meta-RaPS

This chapter provides a discussion of several common meta-heuristics, their specific features, advantages and disadvantages. In addition, a detailed description of Meta-RaPS is presented.

2.1 Common Features of Meta-Heuristics

Meta-heuristics use different strategies to find solutions. However, almost all of the meta-heuristics use some combination of these common strategies to find a solution. Some examples of the common meta-heuristics strategies are described below.

A common characteristic in modern search heuristics is being able to accept moves that temporarily degrade the objective function value in an attempt to avoid being trapped at local optima and ultimately yield a better solution. This strategy is also called an "uphill move". For a minimization problem after finding a local minimum, the search is restarted or modified to move to a point with an inferior (higher) solution value than the local minimum at which heuristic has been trapped. It is expected that this uphill move will be able to move far enough away from the local minimum so that the search will not converge to this point and the global minimum may eventually be reached.

Modern meta-heuristics mostly use sampling and local search to improve solution quality. Sampling allows meta-heuristics to explore different regions of the solution space so that it is more probable that the final solution is the global optimum. By using randomness, sampling allows meta-heuristics to create a variety of solutions.

Local search is used to further improve an already constructed solution. An important issue in local search is search confinement. Confinement means, if the heuristic converges to a local minimum; the heuristic should avoid further searching in this area in order to save valuable computation time. To overcome this problem, some meta-heuristics use a strategy called diversification. Diversification strategy allows the heuristics to move away from the local optima. As previously mentioned, the uphill move described above is considered to be a typical diversification strategy. On the other hand, some search effort around the local minimum should be conducted under the assumption that neighborhoods may have correlated evaluation/objective function values, and a global optimum may be found in close proximity to this local optimum point. This search around a neighborhood is called intensification. Diversification and intensification are conflicting search strategy procedures. Yet, a good meta-heuristic should be able to balance the intensification and diversification, both of which are vital in finding a good or possibly optimal solution.

Some meta-heuristics, such as evolutionary algorithms, use a set or population of solutions. Instead of moving from one solution to another, these types of meta-heuristics try to characterize the solution space by determining the regions of the solution space that contain the best solutions. By way of comparison, most other meta-heuristics construct one solution within each iteration. In general, constructing a solution is done by adding feasible elements to the solution one by one until no other elements can be added to the solution without the violation of feasibility.

Some meta-heuristics also may use adaptive memory in the way that the information obtained in one solution can be carried to the next, or in other words the good solution traits and characteristics can be identified and used in successive iterations.

To detail these specifics, some of the most common modern meta-heuristics are briefly described in the following section.

2.2 Different Types of Meta-heuristics

This section describes the genetic algorithms, simulated annealing and tabu search, which are most frequently used meta-heuristics in literature and in practice. Apart from these aforementioned meta-heuristics, greedy randomized adaptive search procesure (GRASP) is also described due to its similarity to Meta-RaPS.

2.2.1 Genetic Algorithms

Genetic algorithms are approaches to solve combinatorial optimization problems based on natural selection and evolution mechanics. Genetic algorithms (GA) were originally developed by John Holland at University of Michigan to mimic natural selection and evolution. The seminal work in this area was published in 1975 as Holland's book called "Adaptation in Natural and Artificial Systems". The main theme of the GA is robustness. Nature laws and evolution are thought to be the key elements of robustness where survival of the fittest rule is of primary priority.

The information about the GA applied problem is carried from one population to another population by individuals composed of chromosomes representing decision parameters of the problem. A GA works in the way that individuals having favorable characteristics are more likely to pass their chromosomes, or their traits, compared to those individuals who have less favorable characteristics. The random selection processes

and mutation enable GAs to incorporate randomness in their procedure so that sometimes the types of individuals introduced into the population are not the fittest. As a result, the search is not limited to the previously best performing chromosomes.

Different from most other optimization methods, genetic algorithms use a population of solutions to evaluate the performance of a system based on a fitness function value and screen the population for fit individuals that are more likely to survive. The underlying idea is mating of fit individuals will result in a yield of better fit offspring. The local optimum is avoided by a genetic operation called mutation where at a very low probability the solution may move to a completely different region. If this trial for exploring different region is not fit enough with respect to the fitness value then this solution (or population of solutions) will be eliminated (Goldberg, 1989).

While many variations exist, the main steps used by most of the GA-based heuristics are:

- Representation of individuals: Individuals are represented by chromosomes of gene strings of fixed length. Depending on the application, the genes are represented by binary numbers (0 and 1), or by real and/or integer numbers.
- Fitness evaluation function: Based on a function, the individuals are evaluated. The fitness function depends on the application. For TSP, the fitness of an individual (i.e. solution) will be the resulting length of a candidate tour. Therefore, the smaller the fitness value, the more fit the individual.
- Starting population: A suitable number of individuals are selected as a starting population from which the GA will explore many different generations. A good representative starting population that has good fitness values may be an important starting point for some GA applications.

- Selection: Parents are selected by a rule, which is related to their fitness functions, or fitness rank in the population. Then, they will produce the next generation after a series of genetic operations.
- Genetic operations: Crossover and mutation are the most common genetic operations used to modify the genetic information from parents to their offspring. In mutation, a gene string that is not found in any of the parents may be introduced to offspring. In crossover, where a parent's chromosome pieces are exchanged to form the offspring (Pham, and Karaboga 2000).

Genetic algorithms have proven their value of being robust in many diverse applications. In the literature, GA is applied to various kinds of combinatorial optimization problems. Some examples include the Traveling Salesperson Problem by Wang et al. (2006), the Knapsack Problem by Yuan (2005), and the Set Covering Problem by Vasko (1991).

Schaffer and Eshelman (1996) point out that, combinatorial optimization problems are difficult to solve with GA because representations that induce good schema are hard to find. Typically, GAs needs to be modified for each problem it is applied to (i.e., chromosome representation, elitism, crossover and mutation types etc.), and this lack of universality can be a limitation in optimization problems.

2.2.2 Simulated Annealing

Another popular meta-heuristic is simulated annealing (SA), which is based on the analogy of the annealing of solids. The principles of simulated annealing are discussed in Metropolis et al. (1953) in an algorithm to simulate the cooling of a material

in a heat bath (Rayward-Smith et al., 1996). The idea of using this thermodynamic process as an analogy to solve the combinatorial optimization problems is proposed by Kirkpatrick (1983).

The underlying philosophy of SA is as follows: in thermodynamics if the temperature is high, then the atoms will have higher energy and they are more mobile. The local search is able to move to many different solutions and the chance of moving to inferior solutions is high so that the feasible region is explored in detail. When the temperature decreases the atoms tend to have more rigid bonds between each other and mobility decreases and they become crystallized, for the SA close to the end of procedure the probability of moving to inferior solutions decreases. In other words the moves are accepted only if they enhance the solution.

The final goal is to have a perfectly crystallized material which has the perfect structure. Similarly for the SA the perfect structure is regarded as the optimal solution. If a material is annealed starting from a high temperature and let to cool slowly, the material crystallizes because by starting from high temperature all the atoms are mobile enough to move the desired locations and by cooling slowly the solidification happens gradually without distorting the structure of the crystal structure. In other words, the pieces will have time to move into their desired locations to be crystallized. In contrast, if a low starting temperature is selected and/or material is cooled fast, the resulting end material will not have the perfect crystalline structure, meaning a global optimal point may not be found for optimization purposes.

Simulated annealing is one of the simplest heuristic procedures to apply, yet it is very flexible to modifications which produce further decisions to make and parameters to be set in the application. A drawback of SA is the disappointingly long running times that

may be needed for good quality solutions or convergence to optimality. In the literature SA had been applied to many combinatorial optimization problems (Reeves, 1993) including Traveling Salesperson Problem (Lee, 2005), Quadratic Assignment Problem (Connoly, 1990 and Baykasoglu, 2004), Multi-level Lot Sizing (Kuik, and Salomon, 1990 and Tang, 2004) and Vehicle Routing Problem (Sohrabi, and Bassiri, 2004 and Osman, 1993).

2.2.3 Tabu Search

Tabu search (TS) was originally developed by Fred Glover (Glover and Laguna, 1997). TS is based on procedures designed to cross the boundaries of feasibility or local optimally which are treated as barriers for finding the global optimal solution. The method systematically removes the barriers mentioned which limit the exploration of otherwise forbidden regions. The feasibility barrier is removed by imposing and releasing the constraints. This idea was also developed by Pierre Hansen in the steepest/mildest descent method (Hansen, 1986).

The advantage of tabu search lies in this local search which uses adaptive memory. The local search method tabu search uses is based on an evaluation function that chooses the highest evaluation solution move, subject to tabu list restrictions at each iteration. From the available moves, the move with the best objective improvement, or if this is not a possible move, then the one with the least objective deterioration is made.

The concept of the tabu list and its restriction is simple. A tabu list keeps track of the recently accepted moves made so that these moves should be avoided so as to reduce

the probability of local optimality. The tabu list decides which moves are to be permitted for the next iteration(s).

Tabu search features a list of different strategies to overcome local optimality (Pham, and Karaboga 2000):

- Forbidding strategy is used to avoid cycling by forbidding certain moves. This strategy keeps track of the moves made so far in the local search and prohibits making the same moves again. The drawback of this strategy is it requires extensive computer memory. To overcome this effect instead of keeping all the moves in the list, some number of recent moves is kept. This number is also the size of the tabu list, the smaller the number, the greater the chance of cycling.
- Aspiration criteria make a tabu solution a candidate move if this solution is of sufficient quality and this criterion may prevent cycling. In tabu search, the move attributes are recorded, and if a move is found to yield good quality solutions even if this move is in the tabu list, the aspiration criteria may set this move free and re-use the move since move history shows the value of that move.
- Freeing Strategy deletes the restrictions on the solution so these moves can be made again.

In general tabu search evolved from ideas of the founders Glover and Laguna (1997) and various applications reviewed show that tabu search is an effective method in navigating through large and complex solution spaces. tabu search has been applied to many different kinds of problems: Flow Shop (Grabowski, and Wodecki, 2004; Taillard, 1990), Single Machine Scheduling (Hino et al., 2005; Laguna 1991), Traveling Salesperson Problem (Yang et al., 2006; Glover, 1992), Quadratic Assignment Problem (Misevicius, 2005; Skoring-Kapov, 1990).

2.2.4 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is a meta-heuristic founded by Feo and Resende(1995). GRASP has two main stages: a solution constructor, that generates randomized solutions, and a local search heuristic, that improves the solutions generated by the constructor (Gomes, 2001).

Stage 1: Construction

In the construction stage a feasible solution is constructed in an iterative fashion using an adaptive greedy heuristic. A greedy heuristic is an algorithm that always takes the best immediate, or local, solution while finding an answer (Black, 1998). The greedy heuristic by itself is deterministic and it does not allow variation of solutions. The greedy heuristic ranks the feasible moves to be made and selects the best possible move. The probabilistic component, or randomization, of GRASP is done by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the restricted candidate list (RCL). This RCL choice technique allows GRASP to have different solutions at each iteration and still does not compromise the power of adaptive greedy heuristic (Feo, and Resende 1995).

The greedy heuristic used in GRASP is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element (Feo, and Resende, 1995).

Stage 2: Improvement

After the construction stage is completed a local search is done to further improve the constructed solution. Like the construction stage heuristic, a local search algorithm also works in an iterative fashion but in a different way. Local search replaces the current solution with a better solution in the neighborhood of the current solution (Feo, Resende 1995).

At each iteration, GRASP samples the solution space by a randomized greedy function and then applies the local search to improve the constructed solution. The GRASP procedure is repeated for a large number of iterations and the best solution found is reported when all iterations are completed. An outline of the GRASP algorithm is shown in Figure 1:

GRASP has been used for several problems, such as scheduling problems, routing, facility planning, maximum independent set (Gomez, 2001). Resende and Festa (2001) lists many other applications of GRASP in their annotated bibliography of GRASP.

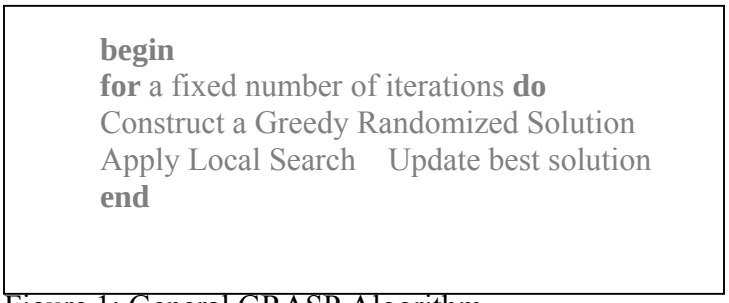

Figure 1: General GRASP Algorithm

2.3 Meta-heuristic for Randomized Priority Search (Meta-RaPS)

Meta-RaPS is a generic, high-level strategy used to modify greedy algorithms based on the insertion of a random element. Meta-RaPS integrates priority rules, randomness, and sampling in each iteration, As with other meta-heuristics, the randomness represents a device to avoid getting stuck in local optima (Moraga, 2002).

Meta-RaPS constructs and improves feasible solutions through the utilization of a greedy algorithm in a randomized fashion. After a number of iterations, Meta-RaPS reports the best solution found. Meta-RaPS is the result of research conducted on the application of a modified COMSOAL approach. COMSOAL (Computer Method of Sequencing Operations for Assembly Lines), which was developed by Arcus (1966), is a computer heuristic originally reported as a solution approach to the assembly line balancing problem. Whitehouse and Tidwell (1980) modify COMSOAL for the resource allocation problem. Although Meta-RaPS conserves Arcus's original idea, it differs considerably from the original COMSOAL. This significant difference led DePuy and Whitehouse (2001) to present their approach as Meta-RaPS.

Meta-RaPS, as well as COMSOAL, constructs solutions by generating a list of feasible elements that may be added to the partially constructed solution. The next element to be added to the solution is randomly chosen from the list referred to as the candidate or available list. This iterative process of building the solution is continued until all the feasible elements are included in the solution and no more feasible elements can be added to the solution (i.e., until the available list is empty). Many iterations are run and best solution is reported. One should note that because randomness is involved in the selection of the elements that are to be included in the solution, different iterations will

give different solutions most of the time. The difference between Meta-RaPS and COMSOAL comes from the way that the procedure chooses the next element to enter the solution. Meta-RaPS chooses the next feasible element from the available list either strictly by a priority rule or by a relaxed priority rule.

Meta-RaPS has two stages, construction and improvement. In the construction stage a feasible solution is built by adding elements to a solution based on a priority rule. Meta-RaPS modifies the construction algorithm such that the next element to be added to the solution does not always have to have the best priority value. The construction stage ends after no possible elements can be added to solution. In the improvement phase, a local search is applied. The local search may use a similar priority rule as the construction stage or a completely different procedure.

The Meta-RaPS technique involves the use of four parameters: number of iterations, *I*, the priority percentage, *%p,* the restriction percentage, *%r,* and the improvement percentage, *%I*. For a number of iterations *I*, Meta-RaPS constructs feasible solutions and Meta-RaPS will pick the best solution from *I* iterations. During each iteration, the parameter *%p* is used to determine the percentage of time the next activity will be scheduled using the base or unmodified priority rule. The remaining *(100%-%p)* of the time, the priority rule is modified by *%*restriction, *%r* parameter. In *(100%-%p)* of the time the next element added is randomly chosen from the feasible elements whose priority values are within *%*restriction of the best priority value. Improvement *%I* decides if the solution created at the construction stage is worthy to be improved. The improvement heuristic is used if the solution value at the end of construction stage is within *%I* of the best unimproved solution value found so far from the preceding iterations (Moraga, 2002).The solution quality of the Meta-RaPS depends on the *number*

of iterations I, %priority %p, %restriction %r and *%improvement %I* parameters. Meta-RaPS integrates priority rules, randomness by *%p, %r* and *%I* parameters and uses sampling by *I* parameter.

The general steps in applying the Meta-RaPS methodology to any combinatorial problem are as follows (Moraga, 2002):

1. Study the structure of the problem to be solved.
2. Find priority rules that construct feasible solutions.
3. Modify priority rules to incorporate randomness by adjustment of %p and %r parameters.
4. Construct feasible solutions using priority rule and randomness.
5. Improve selected solutions, keep the best solution found by Meta-RAPS for both construction and improvement stages.
6. Report the best solution found at the end of I iterations.

Apart from COMSOAL, the Meta-RaPS procedure is also similar to GRASP. However, there are two main differences between Meta-RaPS and GRASP. In the construction stage, Meta-RaPS either uses the pure greedy heuristic or uses randomness inserted (relaxed) into the greedy heuristic whereas GRASP always uses randomness inserted into the greedy function during the construction phase. The second difference is: in the improvement stage Meta-RaPS improves the solutions that are *%I (improvement percentage)* close to the best solution found so far. On the other hand, GRASP improves every solution produced by it's construction stage. In some GRASP applications (Feo et al.,1994; Prais and Ribeiro, 2000), which are referred as GRASP with filtering strategies, the similar idea of applying local search is applied only to some promising solutions. In conclusion, Meta-RaPS may be seen as a generalization of both COMSOAL and GRASP.

Meta-RaPS has been applied to the Traveling Salesperson Problem (Moraga et al.,2001; DePuty et al., 2005), the Set Covering Problem (Lan et. al, forthcoming), the Resource Constrained Project Scheduling Problem (DePuy. and Whitehouse 2001), Vehicle Routing (Moraga, 2002), and the Knapsack Problem (Moraga et al., 2005). Meta-RaPS has demonstrated good performance in terms of both solution quality and computation time with respect to other meta-heuristics (genetic algorithms, neural networks, simulated annealing etc.) Table 1 (from Moraga, 2001) shows a comparison of Meta-RaPS and other meta-heuristics in terms of computation time and percent deviation from the best known solution value for five 100-city TSP test problems taken from TSPLIB (Reinelt and Bixby, 1995).

Table 1: Comparison of meta-heuristics for 5 TSP test problems (Moraga, 2002).

Solution Method	Problem					Run Time for KroA
	KroA	KroB	KroC	KroD	KroE	
Meta-RaPS TSP(Moraga, 2002)	0.00%	0.25%	0.00%	0.00%	0.17%	50 sec
Cheapest Insertion & Node Insertion	0.50%	2.46%	0.82%	1.43%	1.10%	-
GA (Chatterjee et al., 1996)	0.70%	-	1.78%	1.45%	-	1 hour
Neural Networks (Modares et al., 1999)	0.31%	1.43%	-	-	-	55 sec
SA(Voudouris & Tsang, 1999)	0.42%	-	0.80%	-	-	37 sec
TS (Voudouris & Tsang, 1999)	0.00%	-	0.25%	-	-	21 sec

Table 2 (from Moraga, 2002) compares the solution quality of Meta-RaPS to other heuristics for the Resource Constrained Project Scheduling Problem (RCPSP). Table 2 shows the %difference from the optimal solution for 480 RCPSP test problems.

These 480 problems are 30 node RCPSP test problems taken from PSPLIB (Kolisch and Sprecher, 1996). Meta-RaPS produces statistically significantly smaller average percent differences from the results of all the articles in Table 2 except Brucker (1998) in which branch and bound method is used. It is important to note that branch and bound has a much longer average run time than Meta-RaPS.

Table 2: Comparison of Meta-RaPS versus other heuristics for 30 node PSPLIB test problems

Solution Method	Average %Deviation from Optimal	Standard Deviation from Optimal	Maximum %Deviation from Optimal	%Optimal Solutions
Meta-RaPS (Moraga, 2002)	0.598%	1.22%	6.25%	76.00%
Khattab and Choobinch (1991)	3.29%	4.24%	23.68%	47.30%
Brown (1995)	2.05%	2.82%	15.79%	52.10%
Kolisch and Drexl (1996)	0.910%	1.83%	8.62%	75.00%
Brucker et al. (1998)	0.138%	-	4.00%	88.50%

As seen in Tables 1 and 2, Meta-RaPS achieves better or comparable results to other meta-heuristics. Moraga (2002) summarizes the pros and cons of Meta-RaPS in Table 3.

Table 4 compares many of the common meta-heuristics to Meta-RaPS. A meta-heuristic has adaptive memory if it uses the information from the previously found solution to produce a new solution. All meta-heuristics shown in Table 4, except GA, use priority rules to produce solutions, GA instead uses sample solutions (input and output pairs) to function. Although Table 4 summarizes the general attributes, each meta-heuristic procedure is flexible and can be designed in a way to implement different attributes of meta-heuristics depending on the specific implementation. Thus Table 4

shows the general properties of different meta-heuristic procedures when they are implemented in their conventional way.

Table 3: Meta-RaPS pros and cons (Moraga, 2002)

Pros	Cons
1. Global Search Method 2. Not problem dependent 3. High capability of avoiding local optima 4. Keep best solution after a number of iterations 5. Can incorporate intelligence as stopping rule 6. Reasonable runtimes 7. Good results for less-restricted problems, such as TSP 8. Easy to implement	1. Might duplicate solutions 2. Priority rule dependent 3. Not adaptive 4. Does not exploit parallelism 5. Parameter dependent

Table 4: Comparison of meta-heuristics

Meta-heuristic	Adaptive Memory	Use of Randomness	# Solutions per Iteration	Priority Rule Dependent	Need of sample solutions
Meta-RaPS	No	Yes	1	Yes	No
SA	No	Yes	1	Yes	No
GA	Yes	Yes	Many	No	Yes
TS	Yes	Yes	1	Yes	No
GRASP	No	Yes	1	Yes	No

2.4 Research Objectives

Meta-RaPS has been applied to Traveling Salesperson (Moraga et al., 2001), Set Covering Problem (Lan et. al, forthcoming), Resource Constrained Project Scheduling Problem (DePuy and Whitehouse 2001), Vehicle Routing (Moraga, 2002) and 0-1 Multidimensional Knapsack (Moraga et al. 2005) problems and yielded competitive

results with respect to other meta-heuristics. Although the Meta-RaPS procedure is straightforward and easy to implement, its performance is dependent on the parameters (*I, %p, %r, %I*) that it uses. So far numerous techniques have been used for Meta-RaPS parameter selection (Moraga, 2002) but there is no clear consensus on when and why to use a particular parameter selection technique.

Similar to the differences in meta-heuristics explained in Section 2.1, the parameter setting procedures have different strengths and weaknesses. An ideal parameter setting method is not only needed to effectively find the effective parameters settings, but it should also have the following properties:

- Use as few points in the parameter domain as possible and have simple procedure steps. These two points help the procedure to be fast in terms of the computation time.
- Robust for different types of combinatorial optimization problems; be able to suggest effective parameters when used for different type of problems
- Easy to use and ease of repeatability of the procedure. The procedure of application is desired to be straightforward. The amount of human effort required during the application should be minimal and the procedure should be repeatable without complication.

Having a combination of these properties enables a meta-heuristic to be effective and a user friendly procedure. The different attributes of parameter setting procedures will be discussed in detail in Chapter 5.

In order to best demonstrate the parameter setting techniques developed in Chapter 5, two combinatorial optimization problems are used: 0-1 Multidimensional Knapsack Problem (0-1 MKP) and Early/Tardy Single Machine Scheduling Problem with

Common Due Date and Sequence Dependent Setup Times (ETP). These two applications, 0-1 MKP and ETP, are first described in Chapter 3. Chapter 3 also includes the description and results of several modifications made to previous Meta-RaPS 0-1 MKP and Meta-RaPS ETP efforts. The enhanced performance of Meta-RaPS for these two application problems constitutes a contribution of this research in addition to the main focus of this work presented in Chapters 5 and 6 of parameter selection techniques for meta-heuristics.

The research objectives are as follows:

1. Design a robust parameter setting technique for Meta-RaPS, that can also be applied to other meta-heuristics.
2. Implementation of Meta-RaPS to 0-1 MKP and ETP combinatorial optimization problems. Achieve comparable or better solution performance against the other existing meta-heuristic applications in literature for these two combinatorial optimization problems.

CHAPTER 3: Meta-RaPS APPLICATIONS

To compare the different parameter setting methods presented in Chapters 5 and 6 and to demonstrate the parameter sensitivity of Meta-RaPS, two applications of Meta-RaPS are introduced and developed in this chapter. In this research, the 0-1 Multidimensional Knapsack Problem and the Early Tardy Single Machine Scheduling with a Common Due Date and Sequence Dependent Setup Times Problem are chosen as Meta-RaPS application areas.

3.1 Application to 0-1 Multidimensional Knapsack Problem (0-1 MKP)

The 0-1 Multidimensional Knapsack Problem (0-1 MKP) is one of the most studied combinatorial optimization problems. It had been extensively studied in literature and therefore it is a very good benchmark problem. Moreover, there are many simple greedy heuristics which makes the problem an easy application for Meta-RaPS.

3.1.1 Description of 0-1 MKP

The idea of the 0-1 MKP is to fill a knapsack with different types of objects to maximize the profit or the total worth of the objects in the knapsack. The knapsack has a set of m capacity constraints bounded by b_j where j=1...m. The knapsack constraints are often described as weight constraints which cannot exceed some upper limit. To fill the knapsack, n different types of objects of c_i worth and of a_{ij} weight is available i=1...n. The formulation of 0-1 MKP is as follows:

Where $\mathbf{x_i}$ is a binary variable; 1 if item *i* is selected, 0 if item *i* is not selected

The objective of the 0-1 MKP is to maximize the total worth or profit subject to the constraints. The solution of a 0-1 MKP is a vector **x** of size *n* which is composed of binary numbers. The real world applications of 0-1 MKP include: cargo loading (Shih, 1979), cutting stock (Gilmore and Gomory, 1966) and capital budgeting (Weingartner, 1967). Cargo loading aims to fill the designated area with the most valuable load. Cutting stock tries to partition an area into different sizes in the most profitable way. The capital budgeting problem tries to maximize the total payoff by selecting options from a list of possible investment options.

3.1.2 Meta-RaPS 0-1 MKP Application

Performance of the Meta-RaPS 0-1 MKP application relies on the priority rule selected. In this application the priority rule selected is found by Moraga (2003) and named as Dynamic Greedy Rule. Most priority rules for 0-1 MKP are based on a profit-weight ratio calculation for each item to be added to knapsack. This ratio is called pseudo-utility ratio. The pseudo-utility ratio is $\alpha_i = c_i/w_i$; where w_i is the penalty factor for item *i*. After pseudo-utility ratios are calculated for each item, the items are ordered in decreasing pseudo-utility order and the ordered items are added to the solution one by

one as long as they do not violate any constraints. There are different variations on the calculation of the penalty factor, w_i, in the literature (i.e. primal effective gradient method by Toyoda (1975); dual-effective gradient method by Senyu and Toyoda (1968) and lognormal point function using w_i by Cho et al. (2004a and 2004b).

In this application the Meta-RaPS priority rule selected uses the normalization of weights idea introduced by Cho et al. (2004a and 2004b). This heuristic uses a lognormal point function for the weight vector. This transformation gives more weight to the constraint with the least resource remaining. By this way the priority rule selects items that use the scarce constraints (resources) effectively, allowing the priority rule to include more items in the knapsack. This idea from Cho et al. (2004a and 2004b) is combined with a new weight ratio devised in this research, to yield the following weight formula:

In equation 3, Φ^{-1} is the inverse of standard normal cumulative density function and σ takes the value 3. σ is the shape parameter and is set empirically. The term inside the Φ^{-1} function (CW_j / b_j) is normalization by dividing the amount of resources remaining with the initial capacity of the knapsack constraints. In other words, CW_j is the amount of the j^{th} resource consumed by the items assigned so far and a_{ij} is the weight of item i for the constraint j. The idea of using lognormal point function is that this transformation assigns a higher priority to the constraints with less resources remaining, so that scarce resources (constraints with less capacity remaining) will be more effectively used.

For the Meta-RaPS improvement stage, the exchange neighborhood search is employed which tries to exchange the items in the solution with the items that are not in solution for all possible combinations. The improvement stage in Meta-RaPS 0-1 MKP

application has two different exchange improvement procedures; 2-way and 1-way exchange improvement. Initially any possible 2 items which are already included in the solution are exchanged with any 2 items that are not in the solution. If the solution value improves, the exchange is accepted, otherwise, the exchange is rejected. After 2-way exchange, 1-way exchange is done, any item in the solution is exchanged with an item not in the solution. Same as in the 2-way exchange, if the exchange improves the solution value it is accepted, else the exchange is rejected. The 0-1 MKP Meta-RaPS approach is detailed in Figure 2.

The Meta-RaPS 0-1 MKP application is coded in C++ and tested on a P4 2.2 GHz PC. The application is tested on 55 small-sized test problems from MP-TESTDATA (Skorobohathyj, 2002) and 270 large-sized test problems from OR-Library (Beasley, 1990). Both of the problem sets are commonly used benchmark problems used by many researchers.

Meta-RaPS is run for 1,000 iterations for each problem in the small problem set. The small set includes problems from various researchers. The problem sizes range from 2 to 30 constraints and 10 to 105 items. All the problems in this set have known optimal values. For each different problem set generated by different researcher, the Meta-RaPS parameters are selected using trial-and-error.

Table 5 shows the number of optimal solutions obtained, the average and the maximum percentage deviation from the optimal solution. Table 5 compares the solution quality of Meta-RaPS with the improvement phase versus the non-improved Meta-RaPS. The results show the enhancement gained by the improvement stage. For the entire 55 small problems, *%I* is set as 20. Meta-RaPS is able to solve all the 55 problems in the

small problem set optimally with improvement. The entire data set is optimally solved by most well designed heuristics.

```
Input Parameters (I, %p,%r,%I) and  Load Test Problem
Best Solution=0, Best Nonimproved Solution=0, S={x1,x2,…,xn}
While NofIterations ≤ I
        Xselected={Ø}, Xnonfeasible={Ø}
        DO
                p= Random(1,100)
                IF p≤%p
                        Select element k such that αk= max (ct / wt )
                                                        t ε S
                        IF xk feasible THEN xk → Xselected
                        ELSE  xk → Xnonfeasible
                        END_IF
                ELSE
                        Select element k randomly from
                                { t ε S | such that αt≥ max α [1-(%r/100)]}
                        IF xk feasible THEN xk → Xselected
                        ELSE  xk → Xnonfeasible
                        END_IF
                END_IF
        While Xselected + Xnonfeasible ≠ S
                IF currentsolutionvalue> Best Nonimproved Solution
                Best Nonimproved Solution= currentsolutionvalue
                END_IF
        IF currentsolutionvalue> Best Nonimproved Solution[1-(%I/100)]
                 FOR every pair of items of Xselected
                        IF they can be switched with a pair in Xnonfeasible
                        without violating feasibility THEN → Exchange
                        ELSE → No Exchange
                END_FOR

                FOR every  item of Xselected
                        IF they can be switched with an item in Xnonfeasible
                        without violating feasibility THEN → Exchange
                        ELSE → No Exchange
                END_FOR
        IF currentsolutionvalue>Best Solution
                Best Solution = currentsolutionvalue
        END_IF
END_WHILE
Report Best Solution
```

Figure 2: 0-1 MKP Pseudocode for Meta-RaPS

Table 5: Improved vs. Nonimproved Meta-RaPS for Small Problem Set

	Meta-RaPS without improvement	Meta-RaPS with improvement
# of Optimal	46/55	55/55
Avg. % Dev.	0.007	0
Max. % Dev.	0.99	0

Table 6 compares the new heuristic with the older version of Meta-RaPS 0-1 MKP applications as well as other meta-heuristics in literature.

Table 6: Comparison of Meta-RaPS to other techniques for small sized test problems

Solution Method	Optimal Solutions	Average %Deviation from optimal
Meta-RaPS New Heuristic	55/55	0.000%
Meta-RaPS with Oscillation Improvement (Moraga, 2003)	56/56	0.000%
Meta-RaPS with Insertion & Exchange (Moraga, 2002)	55/56	0.003%
SA PROEXC (Dammeyer, 1993)	31/57	0.328%
TS (Glover, 1995)	57/57	0.000%
GA (Chu, 1998)	55/55	0.000%

The large problem suite has nine different problem sets, each with different problem sizes. The number of constraints (m) in these problems is either 5, 10 or 30 and the number of items (n) is either 100, 250 or 500. For each n-m combination, 30 problems are generated by Chu and Beasley (1998) adding up to a total of 270 problems. For each set of problems, the same n-m combinations of 30 problems are generated. The first ten problems have a tightness ratio of 0.25, the second ten problems have a tightness (or constraint tightness) ratio of 0.50 and the last ten problems have a tightness ratio of 0.75. The objective function coefficients are correlated with the constraint coefficients, because

the correlated problems are more difficult to solve than uncorrelated problems (Pirkul, 1987). For each m-n combination, the Meta-RaPS parameters are set by non-parametric based genetic algorithms as described in Chapter 6. Table 7 shows the Meta-RaPS parameters used to solve large problems. In Table 7, *m* represents the number of constraints and *n* represents the number of items available for each 30 problem data set. Notice in Table 7 that as the problem size increases (i.e., as number of items increases) a smaller *%I* and *I* is used in order to have reasonable computation time.

Table 7: Meta-RaPS Parameter Settings for Large Problems

m	*n*	No. of instances	*I*	*%p*	*%r*	*%I*
5	100	30	1000	20	10	2
5	250	30	500	10	5	1
5	500	30	250	80	5	0.5
10	100	30	1000	80	5	2
10	250	30	500	80	3	1
10	500	30	250	90	20	0.5
30	100	30	1000	95	70	2
30	250	30	500	80	5	1
30	500	30	250	80	3	0.5

Table 8 shows the Meta-RaPS solution performance for the large problems. The optimal solutions for these large problems are not available. The table shows the results as the percentage gap from the LP optimal values. LP optimal values serve as the theoretical lower bounds for the Knapsack Problem.

Table 8: Average % Deviation of Meta-RaPS vs. Other Methods

m	n	Meta-RaPS	Moraga (2005)	GA (1998)	ADP (2002)	Haul & Voss (1997)
5	100	0.63	0.6	0.59	-	0.72
5	250	0.22	0.17	0.14	-	0.36
5	500	0.1	0.09	0.05	-	0.34
10	100	1.16	1.17	0.94	-	1.26
10	250	0.45	0.45	0.30	-	0.74
10	500	0.21	0.20	0.14	-	0.64
30	100	2.04	2.23	1.69	-	2.14
30	250	0.99	1.38	0.68	0.97	1.36
30	500	0.99	0.82	0.35	0.52	1.20
Overall		0.75	0.77	0.53	0.74	0.93

The average CPU time for Meta-RaPS 0-1 MKP is mainly dependent on the number of items (n) of an instance. For the 100, 250 and 500 item problems, the average CPU times are 18.2, 53.7 and 184.6 seconds, respectively. Genetic algorithms (GAs) by Chu and Beasley (1998) have one of the best solution performances so far on these problems, however, their computation time ranges between 6 and 65 minutes on a Silicon Graphics Indigo workstation. Moraga (2005) use a Pentium 4 1.6GHz PC with run times of 7 to 35 minutes per problem. The approximate dynamic programming solution (Bertimas and Demir, 2002) is denoted by ADP on Table 8. The ADP has an average computation time of 87.06 seconds on a Dell Precision 410 machine. Haul and Voss (1997) who used GAs for their solution approach, report their algorithm takes a long time to solve instances, in some cases more than four hours. Since the 0-1 MKP applications are tested on different platforms and machines, the direct comparison of computation time is not deemed particularly relevant.

The Meta-RaPS application implemented in this research provides better performance than the other existing Meta-RaPS application in the literature by Moraga (2005) in terms of both computation time and solution performance for the overall

percentage gap from the LP optimal values across all test problems. However a direct comparison of computation may not be appropriate because computers with different configurations are used for testing. In Table 5, for five test problem sets (5-100, 5-250, 5-500, 10-500 and 30-500) the Meta-RaPS application by Moraga (2005) gives better solution quality than findings of the Meta-RaPS used in this research. When Meta-RaPS is run for as many iterations as Moraga's application (for 5, 100, 500 item problems 10000, 5000 and 1000 iterations respectively), Meta-RaPS outperformed Moraga's results in all cases except the 5-100 problem set which yielded the same percent deviation from optimal. When compared with the other techniques, Meta-RaPS provide better solution performance than the others except for GA (Chu and Beasley, 1998) and ADP (Bertsimas and Demir, 2002). It should be noted that Meta-RaPS is a general solution approach that is applied to various combinatorial optimization problems, and its performance is dependent on the construction and the improvement stage algorithms used (Moraga, 2005).

The reason Meta-RaPS did not perform as well as the other two existing methods can be attributed to either the underlying construction heuristic, or the local search which may not be effective enough to carry the promising constructed solution to optima. It should also be noted meta-heuristics work in different ways and a meta-heuristic's solution and time performance are dependent on many factors such as type of problem, type of test problem and the specific features of the application (i.e., for Meta-RaPS the construction heuristic used and for GA the type of genetic operations used).

3.2 Application to Early/Tardy Single Machine Scheduling Problem with Common Due Date and Sequence-Dependent Setup Times (ETP)

The Early/Tardy Problem with Common Due Date and Sequence-Dependent Setup Times (ETP) variation of Single Machine Scheduling problem is selected as the second combinatorial optimization for the experimentation of different parameter setting techniques discussed later in Chapters 5 and 6. The single machine scheduling problem is studied with many different variations in the literature. In this version of the problem, the objective is to minimize the total amount of earliness and tardiness.

3.2.1 Description of ETP

The objective of the ETP is to complete a set of n jobs $\mathbf{\{j_1,j_2...j_n\}}$ on a single machine as close as their due dates as possible (Rabadi et al., 2004). All jobs are available at time zero and each job has a processing time p_j, and has a sequence-dependent setup time s_{ij}, which depends on the predecessor job $\mathbf{j_j}$. All jobs have a common due date d. The machine is able to work on one job at a time without preemption. After a job j is completed at completion time C_j, it will have an earliness (E_j=max(0,d-C_j)) and tardiness (T_j=max(0,C_j-d)). The objective function of ETP is to minimize the sum of the earliness and tardiness for all jobs (Rabadi, 1999):

In the objective function, earliness can be thought of as a holding or deterioration cost and the tardiness can be thought of as a penalty or loss of goodwill for missing the deadline.

Exact methods, like mixed integer programming, branch and bound, can solve the ETP optimally; however, the computation time becomes impractical as problem size increases. The reason for the extensive computation time is the sequence dependent setup property of ETP makes the problem NP-Complete. As a result, the ETP problem is commonly solved by meta-heuristic methods rather than exact methods. The ETP problem has numerous variations studied in the literature. Each different variation of the problem focuses on a special case of the problem. The problem studied in this study, ETP with common due date and sequence-dependent setup times, has not been addressed by many researchers in literature. In many real life applications, the setup time for a job is dependent on the type of job previously completed. Although sequence dependent setup times have not been extensively studied, this variation of the ETP is important as it often is the case for just-in-time or low batch size production schemes.

According to Rabadi (1999), ETP is addressed by Coleman (1992) with 0/1 mixed integer programming and by Chen (1997) using polynomial dynamic programming. Both of these methods are exact methods. Some meta-heuristic applications to single machine early tardy problem are: Koksalan and Ahmet (2003) who use GA and Song et al. (2005) who use ant colony optimization. Although these problems are not exactly the same problem studied in this research from the meta-heuristic procedure point of view, the closest application to Meta-RaPS had been done by Feo et al. (1996). Feo et al. uses GRASP to solve the sequence dependent single machine scheduling problem but the problem is different than ETP used in this study because it does not involve earliness penalties; their objective function minimizes only tardiness.

Rabadi (1999) proposes a heuristic called shortest adjusted processing time (SAPT). Instead of using processing time, which is constant for each job, and setup time,

which is dependent on sequence of the jobs, SAPT uses the advantage of constant processing time for each job and merges setup and processing time by adding each setup time, and the constant processing time. This new value is called adjusted processing time (APT). The idea behind SAPT comes from a property of Early/Tardy Problem without setup times (or Early/Tardy Problem with sequence independent constant setup times). It is known for the Early/Tardy Problem without setup times, that in the optimal schedule the early jobs should follow longest processing time (LPT) order and tardy jobs should follow shortest processing time (SPT) order (see Figure 3). This scheduling is called V-shaped scheduling because the job processing times increase as a job is scheduled far from the due date regardless if it is an early job or tardy job and vice versa (Rabadi, 1999).

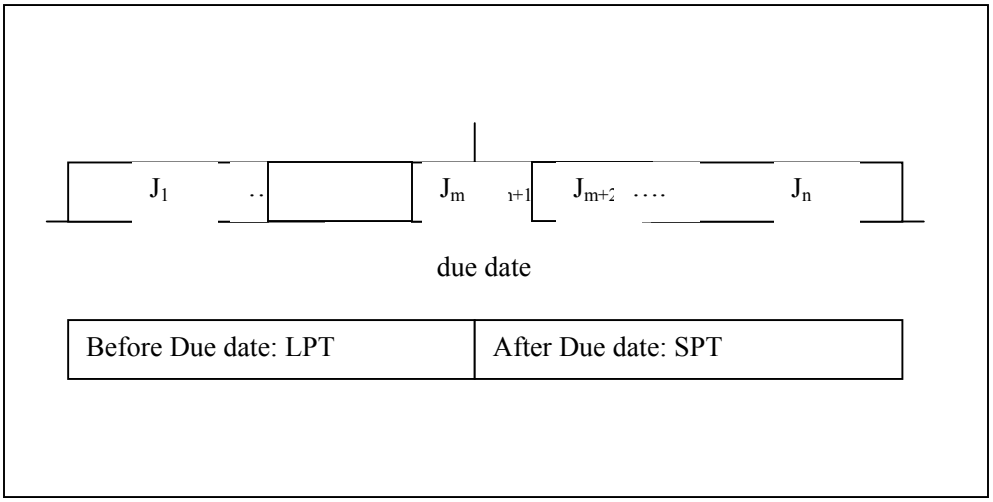

Figure 3: Optimal sequence for E/T Problem without setup times

For the sequence dependent setup time case considered in this work, the V-shaped scheduling does not guarantee optimality because the dynamics of the sequence dependence may force scheduling a job that does not follow LPT rule before the due date

or SPT rule after the due date for the optimal job schedule. In other words, in some cases a job with longer APT time can be scheduled close to the middle (or due date) so that the setup times for other jobs may be minimized. The discussion above concludes that the dynamics of the sequence dependent setup times may not allow a perfect V-shaped sequence to have the optimal schedule configuration. However, even though a perfect V-shaped sequence may not be the optimal configuration, a V-shaped like or distorted V-shaped sequence will likely result in a better objective function value (Rabadi, 1999). Rabadi (1999) bases his heuristic on this idea and develops with SAPT which builds distorted V-shaped solutions.

SAPT builds three different schedules: early schedule first (E), tardy schedule first (T) and early and tardy schedules simultaneously (ET). In the case of E schedule, initially an early schedule, (jobs scheduled before the due date) is built by selecting jobs with the smallest APT combination for the due date and then adding remaining jobs by choosing the smallest remaining APT values. After the early schedule is built, the tardy schedule, (jobs schedules after the due date) is built by the same rule. In the T schedule initially the tardy schedule is built and the early schedule is constructed after that. T schedule uses just the opposite sequence of scheduling of the E schedule. Different from the E and T schedules, the ET schedule starts building solutions from the due date, and then both early and tardy jobs are added to the solution simultaneously depending on which APT in the list is smallest. In other words ET schedule simultaneously builds both early and tardy schedule.

SAPT is carried out for the number of jobs of an instance (n) times, each time starting with a smaller AP in the AP matrix. SAPT is executed n times for n entries in the AP matrix and each time comes up with 3 schedules. In the end, SAPT evaluates three

schedules (E,T and ET) by starting with n different initial AP value, which is equal to 3*n* schedules in total.

3.2.2 Meta-RaPS ETP Application

The SAPT algorihm is used as the priority rule for the Meta-RaPS ETP application. In order to have a variation of different answers within a type of schedule, SAPT starts with a different job pair for the first assignment, and then the rest of the algorithm is carried out in a greedy fashion. Instead of using 3 different types of schedule forming strategies (E, T and ET schedules), only ET scheduling is randomized for Meta-RaPS application. The reason why only the ET schedule is coded is because it is expected that the randomness introduced by Meta-RaPS can actually be able to construct the E and T schedule solutions as well as the ET schedule solutions.

In Meta-RaPS ETP at any point in the solution construction, %priority of the time the greedy SAPT ET rule is used. In the remaining *(100%-%priority)* of the time the job to be assigned is randomly selected from the list of candidate jobs which are *%restriction* close to the job with the smallest AP time. Incorporating randomness in SAPT in this manner does not further necessitate running the heuristic *n* different times starting with a job combination that has different AP. Instead if Meta-RaPS is to be run for *n* times, the randomness will produce a variety of results.

Rabadi (1999) uses generalized pairwise interchange (GPI) for the local search as part of the SAPT heuristic to enhance the solution quality further more. GPI considers all the possible 2-job swap combinations and checks if the objective function improves, if this is the case, jobs are swapped. This type of local search heuristic searches $n(n-1)/2$

different neighborhoods for a problem size of *n* jobs. Meta-RaPS ETP application also uses GPI after the SAPT heuristic. Using the same local search provides a basis for a fair comparison of SAPT and Meta-RaPS if both heuristics are run for same number of iterations.

Meta-RaPS ETP application is coded in C++ and the application is tested on P4 2.2GHz PC. The ETP test problems are generated by Rabadi in two sets; smaller (Rabadi, 1999) and larger instances (Rabadi et al., forthcoming). The small ETP test problems have 10, 15, 20 and 25 number of jobs and they have three different settings of APT: low, medium and high setting. For each setting and number of job combination, 15 problems are generated by Rabadi (1999). For the smaller test set generation, the APT values (only set of values that is needed to be generated to create an ETP test problems) are independently drawn from uniform distribution as: unif(10, 10+*R*). *R* adjusts the three level of settings: low, medium and high. *R* is set as 50 for low, 100 for medium and 150 for high settings. The larger ETP test problems have 40 and 50 number of jobs and two different settings, low/high. The larger set is generated in the same way as the smaller problems. Similar to the small problem set, for the larger problem sets for each job-setting combination 15 test problems are generated. The low, medium and high settings refer to the range of random numbers from which the AP values are sampled.

The optimal values for the small problems are known; however, for the larger problems the simulated annealing solution (Rabadi et al., forthcoming) is taken as the best known solution for the Meta-RaPS. A trial-and-error of priority% and %restriction parameters of Meta-RaPS had been done for each job size and for each APT value range of problems.

The Meta-RaPS ETP approach is shown in Figure 4.

The parameter tuning of Meta-RaPS is done by trial-and-error for each problem set of 15 problems. Table 9 shows the parameters used for Tables 10, 11 and 12.

Table 9: Meta-RaPS Parameter Settings for ETP Problems

Number of Jobs	Setting	No. of instances	*%p*	*%r*
10	low	15	30	30
10	medium	15	50	50
10	high	15	60	60
15	low	15	20	20
15	medium	15	50	50
15	high	15	20	20
20	low	15	40	20
20	medium	15	40	40
20	high	15	50	50
25	low	15	40	10
25	medium	15	40	40
25	high	15	40	40
40	low	15	70	5
40	high	15	70	7
50	low	15	70	3
50	high	15	70	4

The Meta-RaPS ETP application is initially compared with its underlying priority rule, SAPT, in Table 10. This comparison shows how much Meta-RaPS can enhance the priority rule it uses. In this study Meta-RaPS is initially run to produce the same number of schedules as the greedy SAPT heuristic to establish fairness between comparisons. The number of iterations for the 10, 15, 20 and 25 job problems are 30, 45, 60 and 75, respectively.

```
Input Parameters ( I,%p,%r,%I) and  Load Test Problem
Best Solution=0, Best Nonimproved Solution=0, J={J1,J2,…,Jn}
While NofIterations ≤ I
        Escheduled={Ø}, Tscheduled={Ø},    APnonfeasible={Ø}
        DO
                p= Random(1,100)
                IF p≤%p
                        Select pair of jobs with smallest APij
                         Assign Ji as the last job in E schedule Escheduled
                         Assign Jj as the first job in T schedule Tscheduled
                         Update APnonfeasible as a job cannot follow Ji &
                                                 a job cannot precede Jj
                                                 and any possible combination for
                                                 jobs that are assigned
                ELSE
                        Select APkl randomly from


                        Assign Ji as the last open position job in E schedule Escheduled
                         Assign Jj as the first open position job in T schedule Tscheduled
                         Update APnonfeasible as a job cannot follow Ji &
                                                 a job cannot precede Jj
                                                 and any possible combination for
                                                 jobs that are assigned
                END_IF
        While all positions in Escheduled and Tscheduled are filled
Update current solution value
IF currentsolutionvalue< Best Nonimproved Solution
                Best Nonimproved Solution= currentsolutionvalue
END_IF
IF currentsolutionvalue< Best Nonimproved Solution[1+(%I/100)]
        do GPI
        FOR i=1 to n
                 FOR j=i+1 to n
                          Swap schedule of Ji and Jj if objective function improves
                 END FOR
        END FOR
END_IF
Update currentsolution with current schdule
IF currentsolutionvalue<Best Solution
        Best Solution = currentsolutionvalue
END_IF
END_WHILE
Report Best Solution
```

Figure 4: ETP Pseudocode for Meta-RaPS

In general Meta-RaPS applications, only a percentage of solutions from the construction stage are improved. However since one of the main objectives here is to compare the SAPT heuristic with Meta-RaPS under the same conditions, all the solutions from the constructions are improved by the GPI local search.

In Table 10 the Meta-RaPS results are compared with SAPT at same number of iterations. SAPT results are taken from Rabadi (1999). The values on the table are the minimum, average and maximum percentage deviation from the optimal solution for all three types of problems (low, medium, high settings for each 15 problems) and the standard deviation of the deviation from optimal. The results show that Meta-RaPS not only gives better average percentage deviation from optimal but also decreases the variation of solution around the average solution value which makes it more efficient and robust with respect to SAPT heuristic.

Table 10: Meta-RaPS & SAPT % Deviation from Optimal

		Meta-RaPS				SAPT			
Size	AP Range	Min % Dev.	Av. % Dev.	Max % Dev.	St. Dev % Dev	Min % Dev.	Av. % Dev.	Max % Dev.	St. Dev % Dev
10	low	0	0.9	5.39	1.7	0	2.55	11.66	3.6
	med	0	0.6	2.82	1.0	0	2.8	12.38	3.9
	high	0	1.54	7.98	2.4	0	2.8	17.52	5.1
15	low	0	2.23	3.85	1.7	0	3.59	8.06	2.2
	med	0	1.6	5.5	2.3	0	5.33	12.91	4.2
	high	0	2.04	6.76	2.1	0	8.22	18.67	5.5
20	low	0	1.78	4.41	1.5	0	3.13	5.3	1.4
	med	0	3.76	8.36	2.4	0	3.93	12.74	3.4
	high	0	2.43	6.88	1.7	0	5.95	13.7	3.9
25	low	0	1.92	3.06	0.5	1.01	3.21	6.56	1.3
	med	1.93	4.42	6.94	1.8	2.55	5.96	9.37	2.1
	high	2.87	5.1	8.49	1.9	1.39	6.83	11.42	2.7
Average		0.4	2.36	5.87	1.75	0.41	4.52	11.69	3.3

For each size and AP range combination, 15 test problems are experimented. The results in Table 10 are significant because it is rare for a meta-heuristic to have better quality solutions than the guided greedy heuristic rule when both meta-heuristic and the greedy heuristic are run for same number of iterations. When a greedy heuristic is randomized by a meta-heuristic like Meta-RaPS, the benefits of randomization come with further sampling of solutions. In a nutshell, although Meta-RaPS is expected to produce lower quality results than SAPT when same number of iterations are run, it is shown in Table 10 that Meta-RaPS is able to give better solution values. In SAPT heuristic, the solution variation, to form different schedules, is introduced either by building E or T schedule first or made by building n schedules starting with a different due date assigned job pair. SAPT searches only extreme case solutions as in the case of E or T schedules and also the variation of solutions is introduced by running the heuristic n times starting with a different starting job pair for n sized problem. Both of these strategies are not as effective as the randomization from Meta-RaPS and therefore Meta-RaPS outperforms SAPT.

Rabadi et al. (forthcoming) ran simulated annealing (SA) starting from the final SAPT solution to decrease the SA computation time and called this method SAPT-SA. Table 11 compares SA, SAPT-SA and Meta-RAPS for the 15 and 25 job problems. To save computation *%I* is set as 20, meaning 20% of the solutions from the construction stage will be improved. Meta-RaPS in Table 11 is run for 3,000 iterations.

Meta-RaPS is tested on 2.2 GHz Pentium IV PC and benchmarked methods used 1.7 GHz Pentium IV PC for testing and all methods are coded in C++. In Table 11 Meta-RaPS outperforms SA and SA-SAPT in terms of solution quality. Plain SA requires the most amount of time and is close to Meta-RaPS solution quality. Although SAPT-SA

speeds up the SA computation time, the solution quality is inferior to both Meta-RaPS and SA.

Table 11: Meta-RaPS vs. SA and SAPT-SA for Small Problems

	15job low		15job high		25job low		25job high	
	Avg. Dev	Time	Avg. Dev	Time	Avg. Dev	Time	Avg. Dev	Time
Meta-RaPS	0.04	0.33	0	0.33	0.61	3	1.07	2
SAPT - SA	1.95	0	1.87	0	1.11	1	2.53	1
SA	0.21	6.08	0.27	3.73	0.81	20.65	1.34	18.52

Meta-RaPS is also compared to the simulated annealing (Rabadi et al., forthcoming) for larger problems. Since no optimal solutions are available for these sizes of problems, Table 12 compares the solution based on relative percentage difference between the objective functions relative to SA which is labeled as % GAP. In Table 12, positive values indicate that the SA solution is better, meaning Meta-RaPS has a larger sum of earliness and tardiness, and negative values indicate that the Meta-RaPS has better objective function value. Since the SA does not have a clear number of iterations definition, Meta-RaPS was set to be run at the similar computation time at a similar PC configuration for comparison purposes. For 40 size problems Meta-RaPS is run for 5,000 iterations which yielded 22 seconds average per problem where SA took 24 seconds and for the 50 size problem Meta-RaPS is run for 3,000 iterations which took 19.5 seconds where SA took about 20.5 seconds. Each problem set is replicated for 30 times and the average value is reported on Table 12. As shown in Table 12, Meta-RaPS found better solution in 55 problems out of 60 problems. The Table 12 results are replicated 30 times and average of these 30 solutions is reported. When Meta-RaPS number of iterations is

increased (7000 iterations for 40 and 5000 iterations for 50 size problems) Meta-RaPS is able to outperform SA for all problems in large set.

Table 12: Meta-RaPS vs. SA for Larger Problems

Problem set	Avg. % GAP Relative to SA	Max % GAP Relative to SA	Min % GAP Relative to SA	No. of problems Meta-RaPS better
40 low	-0.68	0.07	-1.12	12 out of 15
40 high	-1.07	0.09	-2.54	13 out of 15
50 low	-0.62	-0.13	-1.34	15 out of 15
50 high	-1.45	-0.31	-2.98	15 out of 15

The results from this experimentation conclude that Meta-RaPS is able to introduce randomness and enhance the solution quality of the greedy SAPT. The comparison made with SA and SAPT-SA techniques show that even though the Meta-RaPS local search procedure is simple, it provides effective results for the Meta-RaPS construction stage solutions and it is able to find a good variation of promising solutions. The advantage of Meta-RaPS procedure is it is a simple and effective procedure with only two main parameters to be set. The simple nature of Meta-RaPS coupled with its ability to generate high quality solutions, makes Meta-RaPS a good meta-heuristic method for ETP.

CHAPTER 4: PARAMETER SETTING PROBLEM OF META-HEURISTICS

The parameter setting of meta-heuristics is an important topic because almost all the meta-heuristics have a number of parameters that need to be set. This chapter provides a review of parameter setting efforts of other researchers and motivates the development of the parameter setting procedures discussed in Chapter 5.

4.1 Effect of Parameter Settings and Parameter Setting Techniques

Almost all known meta-heuristics available have a number of parameters that need to be set. As an example meta-heuristics that have been used to solve the Vehicle Routing Problem (VRP) contain anywhere from 4 parameters to 25 parameters depending on the type of meta-heuristic (Coy, 2000). The performance of general purpose meta-heuristics performance like SA, GA, Tabu Search and also Meta-RaPS are dependent on the choice of these parameters.

In his heuristic parameter setting literature review, Coy(2000) states that there are many different procedures to find effective parameter settings. Also the complexity of parameter selection procedures are various, from simple trial-and-error procedures to more sophisticated sensitivity analysis and the use of other meta-models. While some researchers (Van Breedam, 1995) have tried to set all the technical parameters using trial-and–error experiments, there are other researchers who have developed systematic ways to tackle the parameter setting problem. The research efforts for the systematic parameter setting procedures are motivated by the investigation of the effects different parameter settings have on meta-heuristic applications.

Xu and Kelly (1996) tried to identify the relative contributions of five different components of their tabu search heuristic (network flow moves, swap moves, tabu short term memory, restart/recovery strategy and a simple tabu search procedure, TSTSP) for VRP by disabling each component one at a time and comparing the solutions of the five different strategies. They conclude that "… the Tabu Search memory and start/recovery strategy effectively help to locate extremely good solutions and TSTSP provides an effective enhancement over 3-opt…"

Van Breedam (1996) tried to determine the significant effects of parameters for GA and SA for the VRP using a technique called Automatic Interaction Detection Technique (AID) originally developed by Morgan and Sonquist(1963). ".. AID is a tree-based classification method that uses analysis of variance to summarize the relationship between predictor and response variables…". (Morgan, and Sonquist, 1963) AID uses binary splits of parameter setting combinations and at each split an analysis of variance is performed to see the significance of a certain parameter. The result of this research yields the conclusion that certain types of meta-heuristic parameters have "consistent significant effect for all problems." In other words some meta-heuristic (GA and SA) parameters can be set independent of the problem and some parameters are problem dependent.

Schaffer et al. (1989) study the effect of the control parameters affecting online performance of GA for function optimization. The findings of this research show that, for function approximation applications of GA, there are function-independent settings that result in significantly better performance. Like Schaffer et al. (1989), Maier and Whiting (1998) study the variation of parameter settings and their effects on performance for the SA. The study's findings are limited to two different problems, Harverly's Pooling Problems and the Benzene Alkylation Problem. Their results shows that "… the best

values for most of the parameters are largely problem independent…". The only parameter that makes difference in this study is λ, maximum length of a move attempt at the beginning of the algorithm. The study suggests the optimal value for λ is a function of the type of the problem. For the other parameter investigated they did not find a problem independent characteristic. Similar to Maier and Whiting (1998), Van Breedam (2002) study the parametric analysis of 10 different heuristics for VRP. The results of the Van Breedam's study conclude that there are three groups of parameters: problem independent parameters, parameters dependent on the problem characteristics and parameters that have limited or no significant effect on the solution value.

The research on the effects different parameter settings showed that although there are parameter settings for some heuristics that give good solution performance in general, most of the parameters required settings that are dependent on the size and the type of problem studied. The findings of the parameter setting effect studies show that there is a need for development of systematic ways to determine the appropriate parameters values. Moraga (2002) addresses the meta-heuristic parameter setting problem by associating it with the problem of simulation optimization. In both systems, a number of parameters are input to a stochastic system and a response is created. Simulation and Meta-RaPS can be thought of as a black box that takes in inputs (input parameters in Meta-RaPS and decision variables in simulation) and give out stochastic outputs due to variability caused by randomness. This similarity enables the techniques used for simulation optimization to be used in Meta-RaPS parameter selection. The parameter setting techniques considered by Moraga (2002) are: gradient based techniques (stochastic approximation and sample path methods), meta-models (response surface methodology and neural networks), statistical techniques (ranking and selection

techniques and multiple comparisons) and use of other meta-heuristic techniques. One meta-heuristic may set parameters of another meta-heuristic. For example; GA may set the parameters of another GA as done by Grefenstette (1986). Moraga's conclusion based on a comparison of different parameter setting methods is a GA search is less vulnerable to the selection of the initial parameter setting because GA starts with a reasonable sample size. Moraga found that RSM is extremely dependent on the initial design base chosen.

Similar to Moraga's (2002) research, other literature also suggests various statistical methods such as response surface methodology and GA as common techniques for optimization of the heuristic/meta-heuristic performances. Reeves and Steele (1994) use GA to optimize the performance of neural networks (NN) for sensor performance improvement application. The GA optimization of single layer NN structure yielded improvement of 10% over the previous findings for the same NN architectures. Gomes et al. (2001) and Delmaire et al. (1999) use reactive search (RS) to find the parameter settings for GRASP. Reactive methods set the parameters and use the set parameter values to come up with the final result simultaneously. Reactive methods eliminate the parameter setting phase for meta-heuristics. Batiti (1996) study the RS procedure for TS and also provide a bibliography on RS-based TS procedures. A reactive search Meta-RaPS is presented in Chapter 5.

Li and Kwan (2002) use simulated evolution (SE) for the Set Covering Problem. SE is an evolutionary algorithm like GA but they have very different mechanisms. Similar to GA, SE has a number of parameters that influence the overall performance greatly. Li and Kwan use Taguchi's orthogonal experimental design (OED) to tune seven parameters. Full experimentation at different value levels would have required 31,250 (2^1

x 5^6) possible evaluations, but with an OED design 50 experimental trials are used to find close to optimal settings. After the trials are made, analysis of variance (ANOVA) is used to analyze the results to determine how much variation each factor has contributed.

Grefenstette (1986) uses GA to optimize the performance of another GA that is used for a set of numerical optimization problems. Rosen and Harmonosky (2003) set the parameters of a simulation model by using a heuristic that uses techniques from response surface methodology and the SA. Coy (2000) addresses the Vehicle Routing Problem (VRP) and enhances the heuristic's performance by experimental design. The technique used for VRP is gradient descent, which is an iterative and cumbersome process but gives good results for simple search spaces (Van Breedam, 2002). The technique relies on taking sample points with a defined parameter settings experimental design then plotting the regression equation and moving away from the design center in the direction of increasing performance. Golden (1998) also addresses the VRP problem using by lagrangian relaxation (LR) heuristic and enhances it with a GA that is unique in the way that two layers of GA's are used for parameter setting. In the first stage a GA is used to determine good parameter settings for each of several problem instances. The second layer, which is another GA, tries to find robust parameter settings over the whole set of problem. The information from the first GA is systematically aggregated for finding good problem type independent parameter settings with another GA. The results of the study is compared to the experimental design type of approach and found comparable outcome.

The meta-heuristic parameter setting procedures found in the literature, as reviewed in this section, are mostly GA or a variation of RSM. The majority of the parameter search techniques in literature try to set parameters for a specific system or type of problem. It is expected that good parameter settings differ from problem to

problem and even settings may vary for the same type or size of another instance of a problem. For that reason, the parameter selection model has to have a procedure to suggest robust parameter settings that gives good performance for different types of problems.

Table 13: Summary of the meta-heuristic parameter setting methods & parameter effect studies in literature

		Parameter Setting Method Used				Parameter Effect Studies
		Statistical Methods			GA Based Methods	
		Anova	Gradient Descent & RSM	RS		
Parameter Tuned Meta-Heuristics	GA	Van Breedam (1996)			Grefenstette (1986)	Schaffer (1989)
	SA	Van Breedam (1996)				Mainer (1998)
	NN				Reeves & Steele (1994)	
	SE	Li & Kwan (2002)				
	Meta-RaPS		Moraga (2002)		Moraga (2002)	
	GRASP			Gomes (2001) Delmaire (1999)		
	TS			Batiti (1996)		Xu & Kelly (1996)
	Problem Specific Heuristics		Coy (2000)		Golden (2001)	Van Breedam (2002)

4.2 Problem Statement: Robust Parameter Settings in Meta-RaPS

One of the goals of this dissertation is to find an efficient approach to the parameter setting problem of Meta-RaPS. Ideally the parameter selection method should be fast, efficient and should be capable of improving the performance of the heuristic method with respect to using a simpler parameter search.

The amount of human effort, expertise (know-how) and experience required for the parameter setting procedure and the computational time used to set parameters is critical and better to be minimal in the parameter setting problem. If a complicated parameter setting procedure that requires extensive computation time and human effort is to be used for an application, the solution quality should outperform the solution quality that could be achieved by simpler trial-and-error parameter tests and/or random parameter settings.

Ideally the parameter setting procedure suggested for Meta-RaPS in this dissertation should be applicable to other meta-heuristics. Another issue to be considered in parameter setting is a variety of different parameter setting methods may be required for both the degree of parameter sensitivity of different meta-heuristics' solution performance and the degree of parameter sensitivity of a meta-heuristic for different applications may not be the same. In other words for some applications, the parameters may be set by some trial error samples but for others, the parameter search may need more complicated methods for complex interactions between different parameters may exist.

An advantage of Meta-RaPS over many other meta-heuristics is that it relies on only four parameters. For further simplification of the parameter setting these four

parameters can be effectively reduced down to only two parameters that need to be set. It is clear that high values of the parameter *I (number of iterations)* will enable Meta-RaPS to use more sampling and make it have a higher probability of finding better solutions at the expense of computation time. Similar to *I* parameter, high settings of the parameter *%I (Improvement percentage)* will only enhance the solution quality of Meta-RaPS again at the expense of computation time. The values of *I* and *%I* are strongly dependent on the problem type considered, problem size and the embedded heuristic rule (complexity of the priority rule and number of steps in the algorithm). The values of *I* and *%I* should be determined in terms of the time available for solution for specific application. The preceding discussion reduces the parameter set to %p (%priority) and %r (%restriction). %p and %r affect the construction stage of the Meta-RaPS by adjusting the randomness introduced into the priority rule. The parameter setting techniques developed in the next chapters will therefore be tested using the %p and %r parameters.

CHAPTER 5: PARAMETER SETTING PROCEDURES APPLICABLE TO Meta-RaPS

There are many parameter setting techniques available to meta-heuristics. A parameter setting procedure can be dynamic or static which is also called online or offline, and adaptive or non-adaptive. A dynamic parameter setting procedure merges the parameter setting and solution building phases for a meta-heuristic. Dynamic parameter setting methods sample different parameter setting levels and then they converge on the "best found" parameter setting level and ultimately report the best solution found by the meta-heuristic. Meta-heuristics using static, offline, or non-adaptive methods initially require a parameter setting phase in which the best parameter level is found and then the meta-heuristic is run again for the solution building phase using the best found parameter setting level. The flexibility and ease of use of dynamic parameter settings provides an advantage over the non-dynamic parameter setting techniques. Although it is application dependent, dynamic parameter setting techniques in literature provide more effective performance over non-dynamic methods (Agogino et al., 2000).

This section provides a benchmark of parameter setting techniques set for the proposed method presented in Chapter 6. Both dynamic and static techniques are used in this chapter; starting from simple fast procedures to more complicated procedures. In total, five techniques are discussed and at the end of this chapter the methods are compared against each other. A comprehensive comparison of parameter setting techniques such as this was not found in the literature. The development of a new parameter setting technique in Chapter 6 builds on the findings of this chapter.

5.1 Simple Parameter Setting Technique

From the discussion in Section 4.2, it is concluded that Meta-RaPS has mainly two parameters that need to be set. Parameter setting procedures do not guarantee that the optimal parameter settings will be found. In fact an optimal setting may not exist, but rather a range of good settings may be found. Therefore, a simple and straightforward parameter setting method will be adequate for some applications that are not extremely parameter sensitive.

The procedure of this method is as follows: for a specific problem, a number of *%priority* and *%restriction* combinations are searched by making replicated trials covering the parameter domain at a specified number of iterations. The number of iterations should be large enough so that the randomness effect is reduced and the true effect of a parameter setting can be identified. Then, the *%priority* and *%restriction* combination that gives the best result is selected as the best setting found.

Table 14 shows HP2 0-1 MKP test problem parameter settings. Each parameter combination is run for 1,000 iterations replicated 20 times. In Table 14 the average of 20 replications, in which the best solution value out of the 1,000 trials is selected, is shown. As seen in Table 14, the best combination that maximizes the objective function is 10% priority and 20% restriction combination. However it is possible that there may be more than one parameter combinations or a wide region of parameter domain that maximize the 0-1 MKP problem.

If robust parameters for a set of problems are needed then the parameter setting procedure needs to be able to choose the best parameter combination that give good performance in general rather than optimizing the performance of one specific problem.

To achieve this goal a mechanism to aggregate the solution quality of individual is necessary.

Table 14: HP2 Test Problem Results (Best value for 1000 trials each)

		% Restriction								
		10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
	10	3153.4	**3163.4**	3155.2	3154.4	3159.4	3155.2	3154.2	3122	3113
	20	3151	3159.4	3157.8	3160.6	3148	3153	3148.2	3130	3137.2
%Priority	**30**	3152.4	3157.4	3155.2	3157.6	3156	3157.6	3141.8	3140.4	3136.4
	40	3150.8	3153.2	3154	3153.6	3153.2	3150.8	3156.2	3142.2	3142.4
	50	3148.8	3152.4	3152.6	3153.2	3154.2	3148.4	3147.2	3155.4	3157.4
	60	3149.2	3151.8	3150.2	3150	3150.2	3151.6	3150.4	3153	3152.8
	70	3148.8	3152.8	3150	3150.8	3150.8	3151.6	3150.6	3153.2	3150
	80	3148	3148	3150.2	3149.8	3150.4	3148.8	3151.4	3150	3152.4
	90	3148	3148	3149	3149	3148	3150	3149	3149	3149.4

Moraga (2002) uses the following procedure to tune parameters for a set of problems:

1. Select a representative set of problems, preferably problem with known solution values, or lower/upper bounds known, from all of the problems that Meta-RaPS is going to be used on. It is important that the test sample chosen should have different sizes.
2. Select parameter domain and increment over which the parameters are to be varied. *%p* and *%r* range over 0 to 100, this range may be divided into different increment sizes of, 10, 20, 30 or any user specified increments.
3. For each problem in the sample problems, run Meta-RaPS over the entire parameter domain selected by increasing or decreasing the increments.

4. Find the best parameter setting by looking at an aggregate performance measure. The aggregate performance measure, for a specific setting, is best calculated as the solution deviation from optimal averaged over all test problems tested.

5.2 Analytic Parameter Setting Techniques

Apart from simpler procedures, some techniques approach the parameter setting problem in a more systematic way. These procedures are explained and compared in this section.

5.2.1 Response Surface Methodology

5.2.1.1 Description of Response Surface Methodology

Response surface methodology (RSM), or experimental design (ED) procedures, are the most frequently used parameter setting techniques because of their simplicity. RSM tries to find ways to collect as few data points as possible and get most information out of the data points using a statistical model. In RSM parameter setting application literature, Coy (2000) used gradient descent technique to find effective parameters for the Vehicle Routing Problem. His procedure which is applicable for any given type of combinatorial problem is as follows:

A subset of problems from the entire problem set is selected and high-quality parameter settings for each type of problem are found. The parameters found for different type or size of example problems are combined to yield the parameters that work well on

any type and/or example of problems being selected. For the parameter search with RSM, the starting level of each parameter, the range over which the parameter is varied and the increment used are needed. This is usually accomplished using a small pilot study which is done by taking a small number of problems and running some trial solutions to intuitively get a feel for the parameter domain.

Depending on the number of parameters two designs are considered: two-level full factorial designs, if the number of parameters is small, or partial two-level factorial design such as Taguchi design might be used to provide efficiency when there are more than a few parameters. For both the full factorial and partial factorial designs, it is recommended to test both the extreme minimum and maximum parameter settings (often coded as-1 and +1) as well as a mid-point setting (coded as 0) to test for curvature. After the experiment is conducted, linear regression is applied to the response surface and the path of steepest descent is calculated. Next in the direction of the steepest descent data points are taken by making small steps, along the path. The procedure is continued until the limit of the experimental region is reached or the best solution found has not changed for a specified number of steps. The linear regression may not always give optimal parameters, and this method does not provide exact optimization but a good approximation. To make the method more accurate at the expense of computation time, instead a quadratic model can be fit.

Because meta-heuristics give stochastic outputs, when evaluating design points it is important to take more than one trial run for each point. To find general good parameter settings independent of the problem, Coy (2000) recommends averaging the best parameters values found for different problems or different subsets of problems. This procedure could lead to some poor performance possibilities, because the problems being

studied may be too broad for one set of parameters and the class of problems may be divided into two or more subclasses. This division should be done in terms of the significant differences among the problems such as matrix density ratio, size of instance, computed generated of real world data problems, etc.

To summarize the main steps of this approach, Coy's (2000) procedure is as follows:

1. Select a subset of problems to analyze from the entire set of problems
2. Select the starting level of each parameter, the range over which each parameter is to be varied and the amount of increment to each parameter.
3. Find good parameter settings for each subsets, using design of experiments (DOE) and RSM optimization by gradient descent.
4. Average the parameter values found in Step 3 to find robust parameters for the entire class of problems.

The shortcoming of this approach is that even if optimal parameters have been set for each subset of problems, averaging those parameters will give equal importance to different types of problems. Golden(1998) proposed weighing the different examples in terms of their size by taking the natural logarithm of the size of examples and then normalizing this value for all the subsets to find the weights for each subset.

5.2.1.2 RSM Application to Meta-Raps

For the RSM the design center is chosen to be 40% priority and 40% restriction. The model for the HP2 0-1 MKP test problem is shown in Table 15. At α level of 0.05, apart from Priority*Restriction (the interaction term) all other terms are significant. The

model has and R^2 value of 90.3% and R^2 adjusted value of 83.3%, indicating a good fit. Table 16 shows the ANOVA table for the model. Figure 5 shows the response surface of the model, in this plot one can see that the parameter combinations for which the 0-1 MKP test problem is maximized. The response surface in Figure 5, is maximized at lower levels of %priority and %restriction.

Table 15: Estimated Regression Coefficients and Significance of Terms

Term	Coefficient	SE Coef	T	P
Constant	3169.47	17.4794	181.326	0.000
Priority	-0.58	0.5453	-1.073	0.319
Restriction	-0.53	0.5453	-0.980	0.360
Priority^2	0.00	0.0056	0.021	0.984
Restriction^2	-0.1	0.0056	-2.307	0.054
Priority*Restriction	0.02	0.0074	3.089	0.018

Table 16: ANOVA for the Model

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Regression	5	2252.12	2252.12	450.42	12.98	0.002
Linear	2	1732.42	53.26	26.43	0.77	0.500
Square	2	188.45	188.45	94.23	2.71	0.134
Interaction	1	331.24	331.24	331.24	9.54	0.018
Residual Error	7	242.99	242.99	34.71		
Lack-of-Fit	3	196.16	196.16	65.39	5.28	0.065
Pure Error	4	46.83	46.83	11.71		
Total	12	2495.12				

After the model is fitted, the normality assumption is checked by looking at the plot of the residuals, and the model fit parameters. The next step is to find the stationary point using the eigenvalue analysis. The stationary point is the plane tangent to the surface, that is parallel to the XY plane (in 3D). Another definition of the stationary point is where the derivative of the function equals zero. The stationary point is important

because depending on the eigenvalues the response surface has the point of maximum response if all eigenvalues are negative, or the point of minimum if all eigenvalues are positive or the sadde point (point of inflection) if the eigenvalues are mixed in sign. The stationary point is found as 5.55% priority and 2.9% restriction and it is a saddle point because the eigenvalues are mixed in sign.

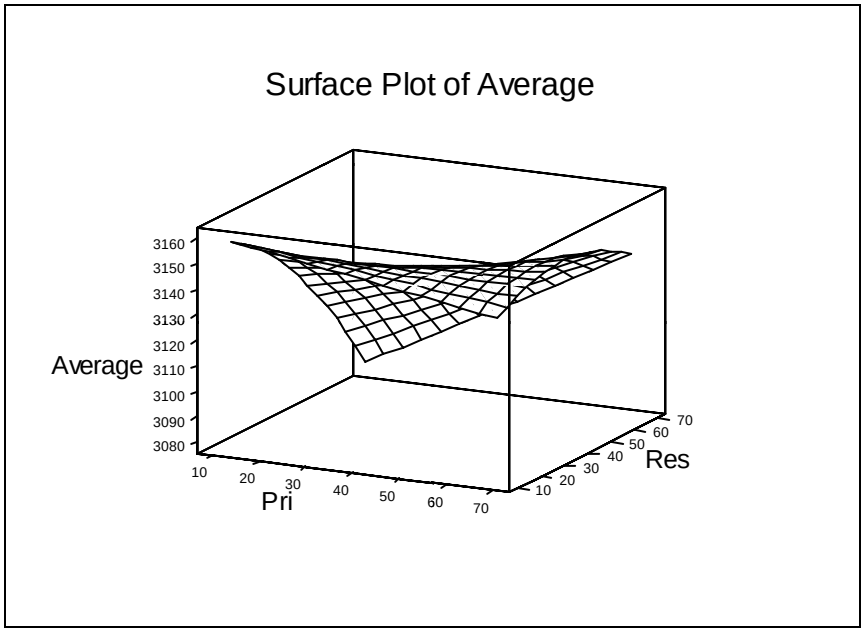

Figure 5: Surface Plot of Knapsack Problem Solution Value

Based on the surface and the contour plots, the direction of gradient ascent is found and another model with a different base is fitted in the region where the 0-1 MKP problem is maximized.

The secondary model has the base at the 10% priority and 10% restriction point. An identical procedure is carried out on this model to determine the validity of the model. The new model has R^2 = 92.8%, R^2 adjusted = 87.7% values and except for the square terms, all other terms are significant. After the eigenvalue analysis, the stationary point

(maxima) is found as 9.5% priority and 5.5% restriction values which is the final parameter setting the model proposes.

5.2.2 Genetic Algorithms

In addition to being a meta-heuristic to solve combinatorial problems, genetic algorithms (GA), can also be used to set parameters for other meta-heuristics. (Grefenstette, 1986)

5.2.2.1 Description of Genetic Algorithms

Description of the GA procedure has been given in section 2.2.1. Genetic algorithms as a parameter search method has many advantages over statistical parameter setting techniques. The parameters of the meta-heuristics may heavily influence both computation time and the quality of the solution. One of the reasons that setting robust parameters is difficult is that there may be complex interactions among different parameters (Golden et al., 1998). Grefenstette (1986) points out that if the response surface is fairly simple, conventional nonlinear optimization or control theory techniques may be suitable, however, for many applications the response surface may be difficult to search, e.g., a high-dimensional, multimodal, discontinuous, or noisy function of parameters.

One main advantage of GA over RSM type of methods is GA can be executed with much less information about the parameter space and the type of problem. Another difference between these methods is the RSM methods require the user to specify a design center which requires the user to have prior information about the solution quality.

Specification of the design center can prevent the exploration of the full parameter space. On the other hand although the RSM procedure is cumbersome, it is a more straightforward procedure which can be applied in the same way to any type of problem. GA parameter search has to be modified for different applications. The analyst has to apply GA in an efficient way for a satisfying performance (Golden et al., 1998).

5.2.2.2 Genetic Algorithms application to Meta-RaPS

GA parameter setting is applied to all large set 0-1 MKP problems. Real-coded GA is used for the parameters of Meta-RaPS, *%priority* and *%restriction,* which are continuous over the [0,100] interval. Blend crossover and random mutation is used as described by Deb (2001), for the reproduction, binary tournament selection with an elitist strategy is used in which the individuals in the top 10% of the population's best performance is transferred to the next generation.

For this application the following GA parameter values are used; population size 30, crossover rate 0.9, mutation Rate: 0.5. These parameters are taken from literature (Deb, 2001) and they are verified/tuned by experimentation. For this application, the starting population is randomly selected. If a better population with higher fitness values is selected, it is expected that average fitness value will convergence at an earlier number generations.

Figure 6 shows the convergence of GA algorithm. After about 30^{th} generation the average solution value, for 30 individuals, of the HP2 0-1 MKP Test Problem becomes stable.

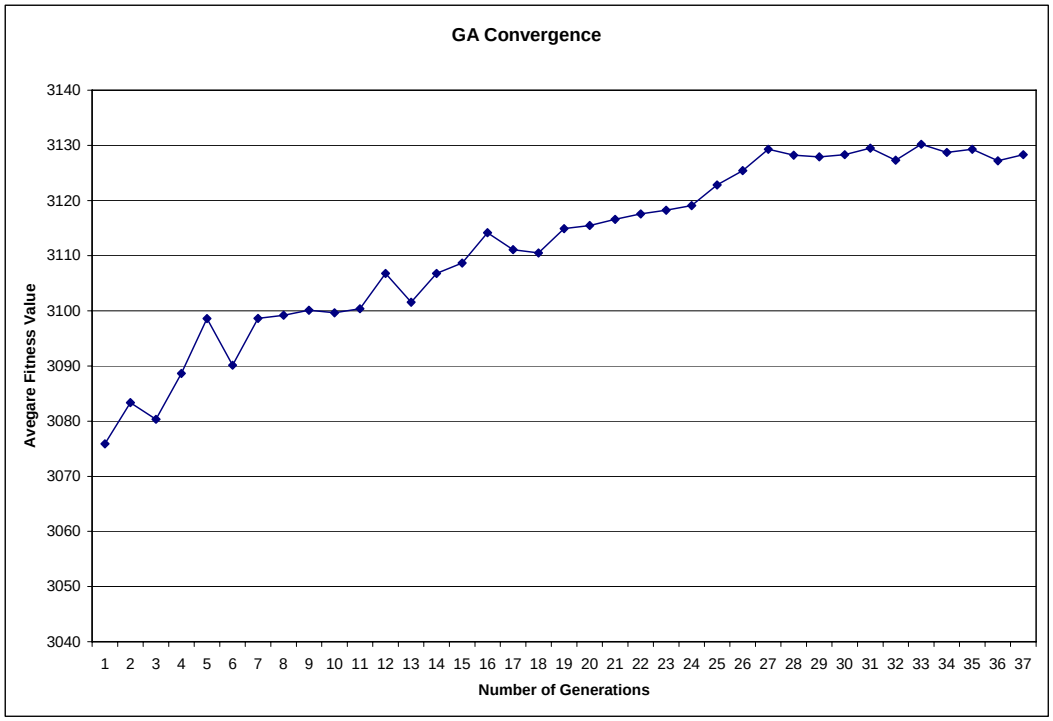

Figure 6: GA Convergence

Table 17 shows the %priority and %restriction values and the corresponding 0-1 MKP solution values (fitness values) of all the 30 individuals in the 30th generation. The best solution value comes from the 26th individual that has a %priority of 13.3 and %restriction 6.6.

Table 17: The %Priority and the %Restriction values of the 30th Generation

Individual Number	%Priority	%Restriction	Solution Value
1	19.5	6.7	3150.6
2	19.5	81.1	3097.6
3	19.5	48.5	3118.8
4	22.4	89.7	2991.6
5	22.4	74.9	3069.2
6	26.0	86.0	2952.8
7	26.0	51.8	3116.0
8	39.2	63.7	3085.2
9	29.0	37.1	3137.6
10	30.2	17.4	3156.0
11	34.9	45.1	3092.6
12	21.7	40.3	3114.0
13	48.8	89.0	3120.2
14	41.0	68.2	3117.0
15	27.9	2.5	3150.6
16	55.8	8.9	3057.2
17	26.0	86.0	3025.0
18	15.3	15.9	3037.0
19	69.1	73.5	3078.2
20	64.9	98.6	3100.0
21	17.0	48.5	3093.4
22	13.4	50.9	3104.0
23	69.9	81.1	3026.6
24	6.1	10.0	3126.6
25	44.9	64.4	3077.8
26	**13.3**	**6.6**	**3159.8**
27	13.4	93.8	3031.8
28	40.8	8.3	3113.0
29	67.7	78.2	3028.4
30	67.7	46.5	3049.8

5.2.3 Reactive Search

Reactive parameter setting methods have a major advantage over other methods in that they eliminate a separate parameter setting phase and incorporate parameter setting and solution building.

5.2.3.1 Description of Reactive Search

The reactive search (RS) method uses the feedback from the meta-heuristic to set the parameters. RS aims to eliminate the parameter setting problems of meta-heuristics and make them robust. RS incorporates a history-based adaptive procedure in meta-heuristic search for online determination of the parameters. History-based learning gradually sets parameter values to better performing parameter combinations. After a given number of iterations or predefined time, the parameters having higher probabilities are determined to be better parameters in terms of solution performance. The online setting of parameters eliminates the need for a parameter setting procedure and sets the parameters as the meta-heuristic is run. The RS procedure has been applied to GRASP meta-heuristics (Gomes, 2001) and TS (Rayward-Smith, 1996; Delmaire, 1999).

In the RS procedure, one parameter setting combination is randomly selected from a candidate set of parameter setting combinations at each iteration and the meta-heuristic is run with the selected parameter combination. At the end of a predefined fixed number of iterations the probability of selecting a particular parameter setting is calculated based on the performance (the solution value) of that parameter setting with respect to the best parameter setting performance found so far. The RS procedure stops when there is no improvement (change in the values of probabilities) for a number of iterations.

For a given parameter setting Delmaire (1999) measures the effectiveness in two dimensions:

1. Quality of solution by determining the average deviation from the best solution known so far.

2. Variability of solution which is the ability to generate many different solutions. This is measured by the proportion of different solutions obtained at a setting by total number of iterations for which the setting is used.

   Both dimensions are aggregated as the utility of the parameter setting.

Gomes (2001) reports that the RS procedure gives better results than the GRASP heuristic alone because it is able to determine appropriate values of parameter(s). Delmaire (1999) reports that for some test problems of Single Source Capacited Plant Location Problem, RS incorporated GRASP reduced the average deviation from the value of best solution obtained with GRASP by at least 50%. For a specific set of test problems, the RS applied GRASP average mean deviation never exceeds 0.5% while pure GRASP is always above 1% deviation.

The RS procedure applied to Meta-RaPS is as follows:

1. Candidate parameter selection: 9 levels of *%p* and 9 levels of *%r* parameters,both starting from 10 to 90 with 10 increments. 81 *%p* and *%r* combinations in total.

   $\mathbf{C}=\{\mathbf{c_1}, \ldots, \mathbf{c_{81}}\}$ (6)

2. Each parameter setting combination in set C is set to have equal probability of being selected

3. Meta-RaPS is run for 200 iterations. At each iteration parameters are randomly selected from set **C** based on their probabilities (p_i) and the best

solution value for all the parameter combinations that are run are stored in array

$$\bar{A} = \{a_1, \ldots, a_n\} \quad (8)$$

4. After every 200 iterations q_i, values are updated according to Eq. 9.

5. The probabilities are updated at the last step by Eq. 10.

6. Procedure is terminated at 1000 iterations. It may also be terminated when there is no change in p_i values for predefined number of iterations.

5.2.3.2 Application to Meta-RaPS

The reactive search is applied to 0-1 MKP. The problem used for the application is HP2. Table 18 shows the probabilities after 5000 trials for the HP2 test problem. Higher probability values for a particular parameter combination indicates better solution performance. The *%priority* and *%restriction* parameters are searched with the increments of 10 within 10% to 90% domain for both parameters.

As the results of the parameter setting techniques suggests, there is more than one parameter combinations that give good results. In Table 18, the 10% priority and 10% restriction combination gives the best solution performance as it is the parameter setting with the largest probability. It should be noted that while using RS to generate the values

in Table 18 to determine the best parameter setting, Meta-RaPS is simultaneously arriving at the final solution value for the MKP problem HP2.

Table 18: Reactive Search Results

%Priority	%Restriction	Probability	Times used
10	**10**	**0.01816**	**16**
10	20	0.01736	12
10	30	0.01575	13
10	40	0.01308	14
10	50	0.01029	19
10	60	0.01146	9
10	70	0.00731	11
10	80	0.00438	5
10	90	0.00057	6
20	10	0.01620	18
20	20	0.01728	16
20	30	0.01469	14
20	40	0.01225	9
20	50	0.01054	9
20	60	0.01096	13
20	70	0.00593	12
20	80	0.00365	5
20	90	0.00213	5
30	10	0.01555	15
30	20	0.01589	12
30	30	0.01479	14
30	40	0.01291	12
30	50	0.01047	12
30	60	0.01004	13
30	70	0.00620	4
30	80	0.00458	6
30	90	0.00413	8
40	10	0.01618	10
40	20	0.01744	9
40	30	0.01542	14
40	40	0.01166	19
40	50	0.01266	15
40	60	0.01012	10
40	70	0.00748	10
40	80	0.00547	17
40	90	0.00502	13
50	10	0.01501	16
50	20	0.01613	13
50	30	0.01538	16
50	40	0.01417	13
50	50	0.01175	12
50	60	0.01162	12
50	70	0.00784	11
50	80	0.00762	11
50	90	0.00707	5
60	10	0.01412	16
60	20	0.01669	19
60	30	0.01581	13
60	40	0.01599	15
60	50	0.01307	9
60	60	0.01152	13
60	70	0.01132	12
60	80	0.01015	12
60	90	0.00886	17
70	10	0.01324	6
70	20	0.01513	12
70	30	0.01559	17
70	40	0.01510	17
70	50	0.01564	13
70	60	0.01423	13
70	70	0.01303	15
70	80	0.01186	13
70	90	0.01208	13
80	10	0.01298	6
80	20	0.01497	15
80	30	0.01574	19
80	40	0.01598	15
80	50	0.01655	11
80	60	0.01457	13
80	70	0.01302	16
80	80	0.01553	12
80	90	0.01493	12
90	10	0.01231	8
90	20	0.01314	15
90	30	0.01399	7
90	40	0.01403	12
90	50	0.01606	12
90	60	0.01481	15
90	70	0.01488	18
90	80	0.01369	13
90	90	0.01477	10

5.2.4 Ranking and Selection Techniques

The ranking and selection (R&S) method is a very common statistical parameter selection method for different settings. This methods is based on systematically eliminating inferior parameter settings.

5.2.4.1 Description of Ranking and Selection Technique

Moraga (2002) proposed solving the parameter setting problem of Meta-RaPS by ranking and selection procedures for simulation. Ranking and Selection methods are techniques in simulation to find the best of k treatments or the subset of size m containing the best of k treatments or m best of k treatments (Moraga, 2002).

Since the goal of the parameter setting problem is to find the best possible parameter settings the R&S technique of selecting the best of k treatments can be applied. In this case, the goal is to select the setting with the maximum response, because 0-1 MKP is a maximization problem, but the procedure can also be applied to set the minimum response without any modifications. Due to randomness preserved in the Meta-RaPS procedure, R&S technique requires a large number of replications to overcome the problem of finding an inferior parameter setting.

To make sure that one setting is better than another, the R&S technique uses two parameters which are to be determined by the analyst. If A is the correct parameter combination and B is the inferior parameter combination which yields close solutions to A, then $P(CS) \geq P^*$, the probability of making the correct selection($P(CS)$) is more than P^*. P^* the parameter determined by analyst should be greater than $1/k$ where k is the

number of systems compred.. Also $A\text{-}B \geq d^*$, the indifference amount $d^*>0$ should be specified by the analyst which is the minimal difference of A and B so that A will be regarded as a better solution than B (Law and Kelton, 2000).

The procedure for selecting best of k parameter combinations is as follows:

1. In the first stage sampling, make $n_0 \geq 2$ replications for each k settings and compute the mean and standard deviation.
2. In the second stage, the total required sample size for setting i, N_i is calculated using the formula:

   where is the smallest integer that is greater than or equal to real number x, and h_1 (k,P^*,n_0) is the Rinott's constant obtained from tables (Mendenhall and Sincich, 1994).
3. Run $N_i\text{-}n_0$ more replications of treatment i and the average for this stage is computed.
4. Compute the weights to needed to combine first and second stage results using the following formula:

5. Compute the sample means as follows:

6. Select the smallest as the best of k treatments (Moraga, 2002)

5.2.4.2 Application to Meta-RaPS

As done for the other methods, the R&S procedure is applied to *%priority* and *%restriction* values between [10,90] with 10 increments except 95% percent last data point is used for both parameters which is used as an additional point to have a higher P* value. For each of ten *%priority* settings, the best of ten *%restriction* value will be selected. For example, for p=10, the best out of 10 settings will be found: (10,10), (10,20), (10,30),…, (10,90), (10,95). Then from the best 10 %priority settings, whose best %restriction values are already set, the best setting for the entire parameter range will be selected (Moraga, 2002).

The parameters selected for the procedure are as follows P^*=0.90, n_0=20, k=10 and constant h_1=3.182. The R&S procedure demonstrated using 0-1 MKP test problem HP2 would proceed as follows.

Stage 1: Selection of the best *%restriction* values for each of the *%priority* values. In Table 19, which is the first screening, the best ten combinations (given a *%priority* value the *%restriction* with largest *Xt* value is selected), are selected for the second and final screening. The ten best value are the highest ten *Xt* values from the Table 19.

Table 19 Shows the best *%restriction* setting for each *%priority*.

Table 19: Selection of best %restriction value for a given %priority (Stage 1)

%Priority	*%Restriction*	X	Std.	Ni	Ni-no	Y(ni-no)	Wi1	Wi2	Xt
10	10	2977.72	99.88	1010	910	2956.93	0.100	0.9	2959
10	**20**	**2972.25**	**87.57**	**777**	**677**	**2968.01**	**0.136**	**0.863**	**2968.6**
10	30	2941.48	109.2	1208	1108	2932.95	0.087	0.913	2933.7
10	40	2890.36	127.5	1647	1547	2875.95	0.064	0.935	2876.9
10	50	2837.06	158.9	2558	2458	2846.98	0.040	0.96	2846.6
10	60	2859.36	164.2	2730	2630	2838.42	0.039	0.96	2839.3
10	70	2780.02	169.9	2923	2823	2767.3	0.036	0.963	2767.8
10	80	2723.86	198.3	3980	3880	2700.62	0.026	0.974	2701.2
10	90	2650.97	198.2	3979	3879	2658.28	0.026	0.973	2658.1
10	95	2598.42	232	5452	5352	2648.14	0.019	0.981	2647.2
20	10	2950.18	98.82	989	889	2950.27	0.105	0.895	2950.3
20	**20**	**2970.72**	**104.6**	**1108**	**1008**	**2966.51**	**0.093**	**0.906**	**2966.9**
20	30	2921.2	118.7	1426	1326	2930.41	0.074	0.926	2929.7
20	40	2874.53	116.1	1366	1266	2873.55	0.078	0.922	2873.6
20	50	2841.78	149.5	2264	2164	2838.86	0.045	0.955	2839
20	60	2849.74	157.6	2516	2416	2819.46	0.04	0.958	2820.7
20	70	2753.6	169.7	2918	2818	2757.2	0.036	0.963	2757.1
20	80	2709.95	192.6	3757	3657	2707.51	0.02	0.973	2707.6
20	90	2680.82	203.6	4198	4098	2680.84	0.025	0.974	2680.8
20	95	2656.2	197.9	3966	3866	2678.04	0.026	0.973	2677.5
30	10	2937.77	94.02	896	796	2945.29	0.121	0.878	2944.4
30	**20**	**2944.23**	**100.2**	**1016**	**916**	**2970.33**	**0.101**	**0.899**	**2967.7**
30	30	2923.13	119.3	1441	1341	2936.2	0.070	0.929	2935.3
30	40	2887.18	129.8	1706	1606	2878.11	0.059	0.94	2878.7
30	50	2840.44	157.6	2517	2417	2850.47	0.043	0.957	2850
30	60	2832.25	165.7	2782	2682	2811.7	0.03	0.962	2812.5
30	70	2758.72	170.7	2950	2850	2762.76	0.037	0.963	2762.6
30	80	2727.67	186.4	3518	3418	2732.53	0.030	0.969	2732.4
30	90	2719.11	204.8	4247	4147	2711.72	0.024	0.975	2711.9
30	95	2706.84	208.9	4420	4320	2698.84	0.024	0.975	2699
40	10	2949.84	103.7	1090	990	2934.45	0.097	0.902	2936
40	**20**	**2973.84**	**105.9**	**1137**	**1037**	**2964.04**	**0.094**	**0.906**	**2965**
40	30	2935.12	103.1	1076	976	2935.41	0.096	0.903	2935.4
40	40	2863.31	135.5	1861	1761	2887.82	0.058	0.942	2886.4
40	50	2882.41	137.3	1910	1810	2858.75	0.055	0.944	2860.1
40	60	2833.73	174.5	3084	2984	2857.02	0.035	0.965	2856.2
40	70	2783.24	157.3	2506	2406	2784.85	0.042	0.957	2784.8
40	80	2744.68	186.9	3536	3436	2756	0.029	0.97	2755.7
40	90	2736.04	198.6	3993	3893	2738.25	0.027	0.973	2738.2
40	95	2727.2	210.1	4470	4370	2736.47	0.023	0.976	2736.3

Continued

%Priority	*%Restriction*	X	Std.	Ni	Ni-no	Y(ni-no)	Wi1	Wi2	Xt
50	10	2927.35	105	1118	1018	2926.69	0.096	0.903	2926.8
50	**20**	**2948.72**	**100**	**1013**	**913**	**2965.16**	**0.104**	**0.895**	**2963.4**
50	30	2934.52	114.8	1336	1236	2942.86	0.080	0.919	2942.2
50	40	2911.27	138.3	1936	1836	2903.74	0.054	0.945	2904.2
50	50	2864.93	171.1	2965	2865	2870.37	0.036	0.964	2870.2
50	60	2862.53	160.5	2608	2508	2842.56	0.041	0.959	2843.4
50	70	2790.15	191.6	3716	3616	2807.31	0.02	0.972	2806.8
50	80	2785.95	195.2	3857	3757	2791.15	0.028	0.972	2791
50	90	2775.38	191.1	3698	3598	2778.4	0.028	0.971	2778.3
50	95	2800.17	203.3	4184	4084	2769.25	0.025	0.975	2770
60	10	2910.39	100	1013	913	2908.67	0.105	0.895	2908.9
60	**20**	**2959.57**	**108.9**	**1202**	**1102**	**2955.64**	**0.089**	**0.91**	**2956**
60	30	2942.68	118.8	1429	1329	2949.73	0.074	0.926	2949.2
60	40	2946.07	116.4	1371	1271	2924.64	0.07	0.924	2926.3
60	50	2890.17	153.2	2378	2278	2891.64	0.045	0.955	2891.6
60	60	2860.61	168.3	2868	2768	2869.87	0.03	0.964	2869.5
60	70	2856.62	160	2592	2492	2843.08	0.04	0.959	2843.6
60	80	2834.24	175.4	3114	3014	2838.82	0.034	0.966	2838.7
60	90	2809.59	193.4	3789	3689	2824.37	0.026	0.973	2824
60	95	2800.63	188	3580	3480	2825.93	0.029	0.971	2825.2
70	10	2893.45	96.13	936	836	2901.39	0.113	0.887	2900.5
70	**20**	**2929.58**	**112**	**1272**	**1172**	**2950.58**	**0.085**	**0.915**	**2948.8**
70	30	2938.49	101.3	1039	939	2948.73	0.097	0.903	2947.7
70	40	2929.07	138.4	1939	1839	2934.52	0.053	0.947	2934.2
70	50	2939.41	130	1712	1612	2919.64	0.063	0.936	2920.9
70	60	2912.39	130.4	1722	1622	2900.33	0.062	0.937	2901.1
70	70	2889.39	142.6	2059	1959	2884.79	0.050	0.949	2885
70	80	2867.04	167	2823	2723	2880.02	0.037	0.963	2879.5
70	90	2871.24	157.9	2524	2424	2871.75	0.043	0.957	2871.7
70	95	2855.38	169.3	2904	2804	2865.24	0.037	0.963	2864.9
80	10	2888.44	97.43	962	862	2875.16	0.113	0.887	2876.7
80	20	2926.6	114.2	1320	1220	2931.08	0.077	0.923	2930.7
80	30	2941.41	121.9	1506	1406	2940.9	0.07	0.928	2940.9
80	40	2946	116	1364	1264	2938.01	0.078	0.922	2938.6
80	**50**	**2956.76**	**117.5**	**1397**	**1297**	**2940.72**	**0.072**	**0.927**	**2941.9**
80	60	2918.99	134.2	1823	1723	2926.46	0.057	0.943	2926
80	70	2889.23	133.9	1817	1717	2923.73	0.060	0.94	2921.7
80	80	2937.25	124	1558	1458	2920.49	0.06	0.934	2921.6
80	90	2925.9	123.5	1545	1445	2913.32	0.06	0.934	2914.2
80	95	2902.49	152.7	2363	2263	2910.36	0.045	0.954	2910

Continued

%Priority	*%Restriction*	X	Std.	Ni	Ni-no	Y(ni-no)	Wi1	Wi2	Xt
90	10	2875.61	102	1053	953	2853.66	0.101	0.899	2855.9
90	20	2891.51	104.8	1113	1013	2896.27	0.097	0.902	2895.8
90	30	2907.73	119.3	1442	1342	2910.98	0.074	0.926	2910.7
90	40	2908.55	120.4	1468	1368	2928.41	0.074	0.926	2926.9
90	50	2947.41	131	1738	1638	2930.89	0.063	0.937	2931.9
90	60	2923.6	117	1386	1286	2920.85	0.07	0.924	2921.1
90	70	2924.93	126.7	1625	1525	2930.07	0.066	0.933	2929.7
90	**80**	**2942.17**	**120.7**	**1475**	**1375**	**2934.33**	**0.071**	**0.929**	**2934.9**
90	90	2922.7	125.8	1604	1504	2928.79	0.06	0.932	2928.4
90	95	2926.2	123.5	1544	1444	2935.65	0.070	0.93	2935
95	10	2846.42	70.95	510	410	2837.31	0.206	0.794	2839.2
95	20	2873.76	101.1	1036	936	2869.75	0.100	0.899	2870.2
95	30	2876.39	97.58	965	865	2876.85	0.113	0.887	2876.8
95	40	2908.75	118	1410	1310	2893.58	0.076	0.923	2894.7
95	50	2889.04	105.3	1124	1024	2895.64	0.094	0.905	2895
95	60	2897.84	108.8	1200	1100	2904.6	0.088	0.911	2904
95	70	2896.75	110.2	1230	1130	2906.77	0.088	0.912	2905.9
95	**80**	**2907.72**	**120.7**	**1476**	**1376**	**2908.24**	**0.073**	**0.926**	**2908.2**
95	90	2908.91	125.4	1592	1492	2907.61	0.066	0.934	2907.7
95	95	2911.47	122.6	1521	1421	2906.01	0.068	0.931	2906.4

Stage 2: The same procedure in Stage 1 is applied to all the selected %priority values with the best %restriction values. The experiment is run again for the parameters of the second stage. Results are shown at table Table 20. The best parameter combination has the the largest Xt value which is the 20*% priority* and 20*% restriction*.

Table 20: Selection of the best parameter setting (Stage 2)

%Priority	*%Restriction*	X	Std.	Ni	Ni-no	Y(ni-no)	Wi1	Wi2	Xt
10	20	2973.0	102	1057	957	2965.4	0.1014	0.898	2966.2
20	**20**	**2984.5**	**85**	**744**	**644**	**2968.4**	**0.1359**	**0.864**	**2970.7**
30	20	2963.1	105	1134	1034	2964.8	0.0922	0.907	2964.7
40	20	2969.0	95	916	816	2966.9	0.1176	0.882	2967.2
50	20	2960.9	106	1153	1053	2960.4	0.0925	0.907	2960.5
60	20	2960.8	104	1117	1017	2952.0	0.0971	0.902	2952.9
70	20	2944.1	116	1386	1286	2941.8	0.0782	0.921	2942
80	50	2940.6	121	1487	1387	2936.3	0.0699	0.930	2936.6
90	80	2931.4	121	1504	1404	2930.5	0.0701	0.929	2930.6
95	80	2903.0	128	1678	1578	2903.8	0.0651	0.934	2903.8

5.2.5 Summary of Techniques for 0-1 MKP and ETP

The four analytical parameter setting techniques discussed in this chapter are also applied to ETP. Table 21 shows the results of all the parameter setting techniques applied to one sample 0-1 MKP and ETP (5^{th} problem of 25 Job high setting set) problems.

Table 21: Final Parameter Setting Suggestions for 0-1 MKP and ETP

Problem	RSM	GA	RS	R&S
0-1 MKP	9% Priority 5% Restriction	13% Priority 6% Restriction	10% Priority 10% Restriction	20% Priority 20% Restriction
ETP	62% Priority 33% Restriction	45% Priority 34% Restriction	50% Priority 40% Restriction	40% Priority 40% Restriction

The different techniques usually result in consistent parameter settings but there may be occasions where alternative parameter settings could be found by different parameter setting techniques. The results in Table 21 show that the best parameter found by different techniques are within a small region of the parameter domain (e.g., in Table 21 the parameter settings of different techniques for the ETP problem are all close to 45% priority and 35% restriction).

The drawbacks of the reactive search (RS) and ranking and selection (R&S) procedures are that they are both dependent on the initial set of candidate parameters which are defined by the user. The principal concern with the response surface method (RSM) procedure is that it does not guarantee optimality. This leaves with genetic algorithms (GA) as the only technique that can globally optimize the parameters. Both R&S and RSM require extensive human effort each time the procedure is repeated. Once the code is completed both GA and RS can be repeated without any human effort.

In summary, aside from issues of initial choice of parameters and being able to find global best parameters, GA appears to be the best in terms of solution quality, and the most robust setting technique. The major strength of RS is its ability to simultaneously set parameters while generating the final result. However, this can be achieved with GA keeping the best solution from all the individuals. Any GA procedure using elitism, which carries the fittest individual to next generation without any genetic operations, is able to keep the best found solution till the end of the procedure.

For its flexibility in application, ease of use, repeatability, global optimization performance, reactiveness, and freedom from user defined candidate parameter settings, the GA appears to be the best parameter setting technique of those considered. A more in depth comparison of different parameter setting techniques is continued in Chapter 6, where the techniques introduced in this chapter are compared to the technique developed in this research effort.

5.3 Setting Robust Parameters For a Set of Problems

Section 5.2 discussed methods to find the best setting or preferred settings for *%priority* and *%restriction* for one problem. However most of the time the Meta-RaPS application (or the application of another meta-heuristic) is used for a number of problems and due to time restrictions it is not always possible to set parameters for each problem and run each problem with its best setting.

To overcome this problem, researchers have studied procedures that find parameters that yield good solutions for a set of problems. If the number of problems in the set is big, it is recommended that instead of using all the problems for parameter

setting, a small subset of problems should be selected. The sample subset problems are selected so that the sample is representative of the problem set in terms of the size and the structure of the problem. The structure of the problem depends on the problem's specifications. For some problem types, the complexity of the problems may also be various. The matrices in the problems that represent distance, flow, capacity may be binary values or integers even though problems may have the same size. Also the density of the matrix (ratio of non-zero element of the matrix to the number of entries in the matrix) may differ in similar problem sizes and this may require a different set of parameter values.

After a representative sample is selected, Coy (2000) proposes setting the best parameters for each of these problems and then averaging the parameter values for the final parameter settings. Golden et al. (1998) combines the parameter settings for each problem by linearly weighting them. Averaging parameter values give equal weight for each test problem. Golden et al. (1998) set the weights equal to the natural logarithm of the size of the test problem for example, number of nodes in the VRP problem. The GA that Golden et al. (1998) implemented searches through the best settings available using the function of tour length for VRP which combines different problems by using the weights. Moraga (2002) used the average of all the %deviation from the best known or optimal solution of the problems in the subset as the aggregate performance measure to find out which parameter settings average %deviation performs best.

Although the parameter setting techniques discussed in this chapter are effective procedures, none of the techniques is initially designed specifically for the parameter setting context. Also some of these techniques (i.e. R&S for ranking of simulation models, both RSM and GA as a general optimization algorithm) are not specifically

tailored to include features to fit the parameter setting context. They are used for parameter setting purpose without any modification to their procedure. Chapter 6 initially discusses how parameter setting methods compare and select parameter settings, and then tries to design a new parameter setting technique that will outperform current parameter setting techniques found in the literature.

CHAPTER 6: PARAMETER SETTING WITH NON-PARAMETERIC BASED GENETIC ALGORITMS

A new parameter setting technique, non-parametric based genetic algorithms (NPGA), is introduced in this chapter as well as the motivation for its development. A comparison of NPGA with the other known techniques discussed in Chapter 5 is also presented.

6.1 Analysis of the best solution behavior

Some of the parameter setting methods (RSM and R&S) used in Chapter 5 assume that the distribution of the best solution from a given number of iterations is normal. It is the nature of any combinatorial optimization problem that the distribution of the best solution values form a number of iterations of a randomized meta-heuristic, taken from a number of iterations will be a discrete distribution. The feasible region of a combinatorial optimization problem will have discontinuous points in the solution space. Additionally the best solution distribution will be bounded by the optimal solution value.

The trials made with both Meta-RaPS 0-1 MKP and ETP applications show that the best solution distribution does not follow any particular distribution. Both the Kolmogorov-Smirnov and chi-square goodness of fit tests reject the fit of a normal distribution. The distribution is dependent on the problem used, type of application and type of algorithm used. In fact it is desired for a well designed meta-heuristic to give non-symmetrical solution distribution. For a maximization problem it is better for the meta-heuristic's solution distribution to be skewed to the left near the optimal solution value. For the minimization problems a solution distribution skewed to the right is preferred.

Figures 7 and 8 show examples of the best solution distributions for the ETP and 0-1 MKP problems. In Figure 7, the distribution for the 7th 25 job high setting problem's best solution (for 100 iterations) is shown. Since ETP is a minimization, the right-skew is expected. While Figure 7 is the distribution for one specific problem, the same general shape distributions are present for all the ETP data sets.

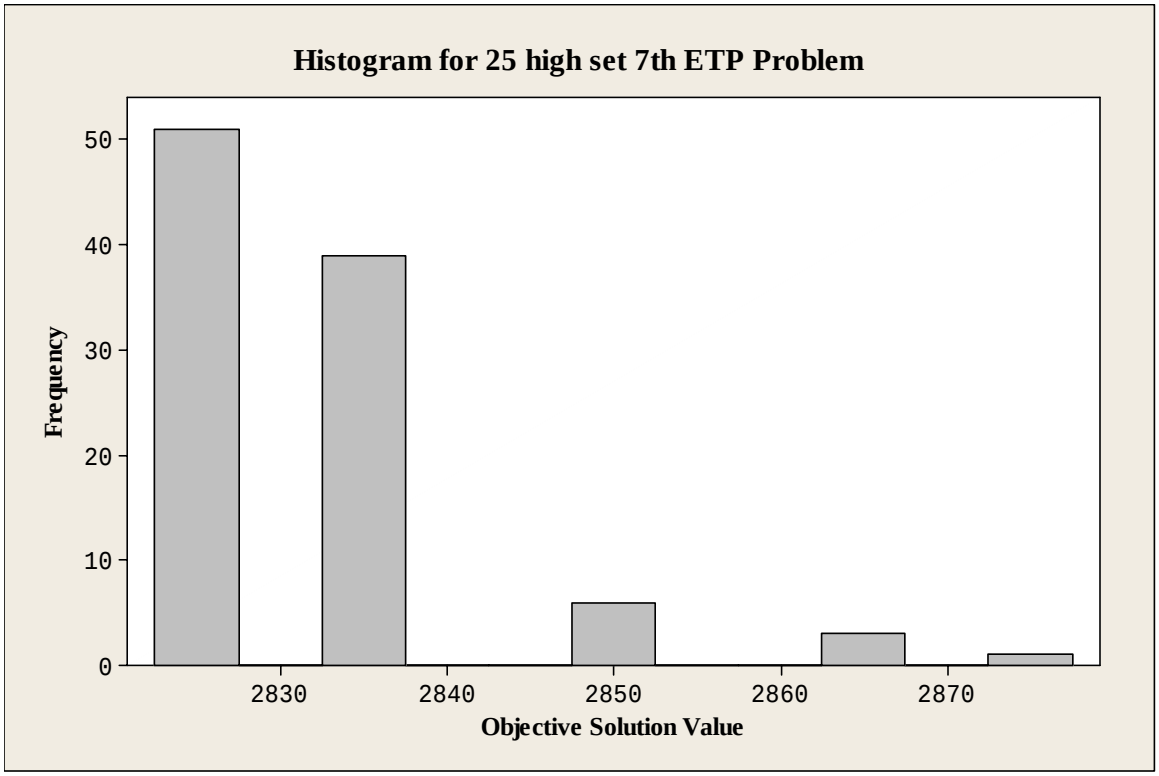

Figure 7: Meta-RaPS ETP 25-7-high Problem Solution Distribution

Table 22 shows that only 15.4% of the ETP Problems solutions are normally distributed and even if the solution distribution is normal, 53% of the time normal distribution is not the best parametric distribution that represents the data. The experimentation showed that most of the best fits come from a beta distribution, followed by triangular and Erlang distributions.

Table 22: Summary of Normal Distribution Fit to ETP Solution Distribution

Normal Distribution is:	Number of Problems	Percentage of Problems in all ETP Problems
Rejected	208	86.6%
Failed to Reject and not found as best fit	17	8.2%
Failed to Reject and found as best fit	15	7.2%
Total	240	100%

Figure 8 shows the distribution of best solutions (from 100 iterations) for the first 0-1 MKP problem of 5 constraint, 100 item test problem set. As summarized in Table 23, almost all the problems in 0-1 MKP set show a similar shaped distribution as Figure 8 which is skewed to the left. Normal distribution rarely represents the best solution distribution from a large number of iterations for the OR-Library (Chu & Beasley 1999) problems. In most cases (93.7% of the problems), normality is rejected based on statistical tests (i.e. chi-square and Kolmogorov-Smirnov).

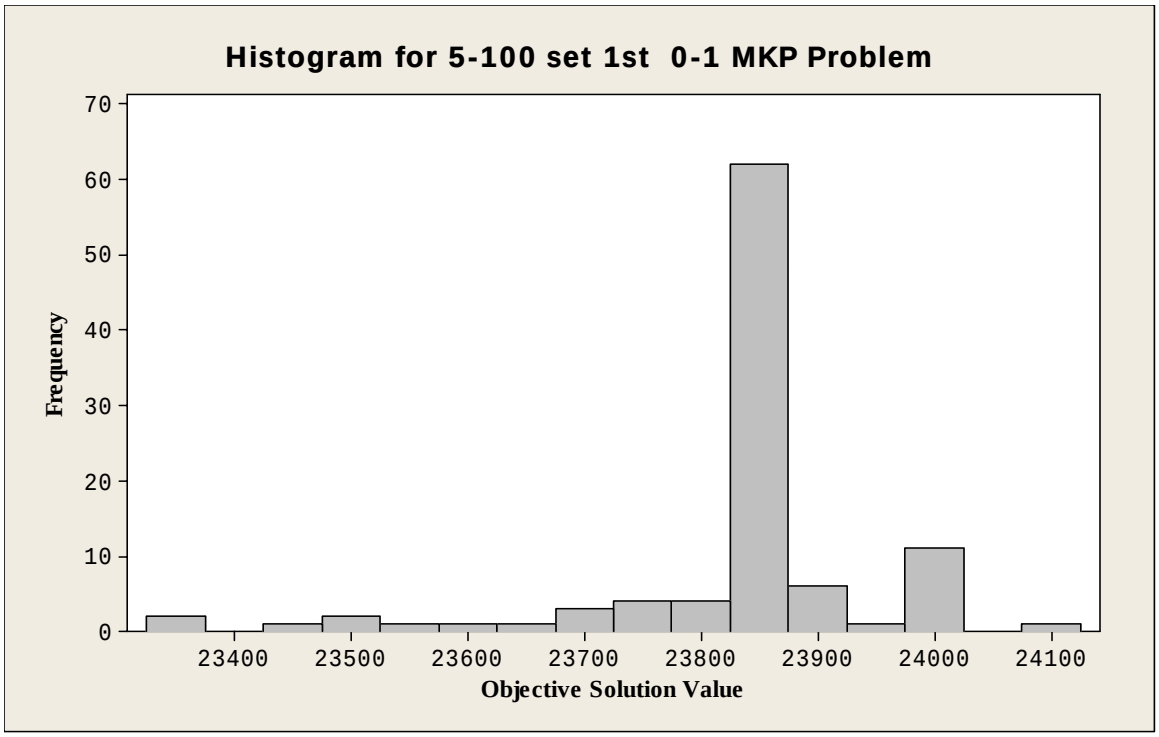

Figure 8: Meta-RaPS 0-1 MKP 5-100-1 Problem Solution Distribution

Table 23: Summary of Normal Distribution Fit to 0-1 MKP Solution Distribution

Normal Distribution is:	Number of Problems	Percentage of Problems in all ETP Problems
Rejected	253	93.7%
Failed to Reject and not found as best fit	7	2.6%
Failed to Reject and found as best fit	10	3.7%
Total	270	100%

Because the solution distributions are not normally distributed, as discussed above, parameter setting techniques that rely on normality assumption to fail or to give inconsistent results. Because the best solution distribution is not normal and dependent on the application and/or problem, the use of techniques involving parametric techniques may not be appropriate for parameter setting purposes. Therefore a robust parameter setting method should make a comparison between parameter setting performances with a distribution-free or non-parametric method. Although normality assuming techniques may work and be able to suggest good parameter settings for some cases, in general they are not the right techniques to represent, or model, the solution distribution of a parameter setting of a meta-heuristic.

6.2 Non-Parametric Tests

When the performance of different parameter values is compared, there can be two comparison cases: two different parameter's solutions can be compared, or a group of parameter's solution values can be compared.

The parametric techniques for these kind of comparisons are t-test for comparing two parameter setting samples and a one-way ANOVA for comparing three or more parameter setting samples. Both parametric techniques assume that the populations come from normal frequency distributions. The non-parametric equivalents of these techniques

are Mann-Whitney (also called Wilcoxon rank sum test) for two samples and Kruskal-Wallis Test for more than two samples. The following description of the Mann-Whitney and Kruskal-Wallis, non-parametric tests are taken from Hollander (1999).

The steps of the Mann-Whitney Test is:

1. Rank two combined samples in ascending order.
2. If there is a tie between two or more the observations then average rank is assigned.
3. Sum of the ranks for first sample (the larger sample if sample sizes are different) is summed and this value is called T_1.
4. Calculate the test statistics

5. Reject H_0: equality of population medians if

In Equation 15, m refers to the sizes of the 2 populations, N is the combined sample size which is $n+m$ and R_i stands for the rank of observations. The z value can be used if both n and m are greater than or equal to 20. Otherwise a t value with N-1 degrees of freedom at specified alpha level should be used.

The steps of the Kruskal-Wallis Test are:

1. Rank combined samples in ascending order.
2. If there is a tie between two or more the observations then average rank is assigned.

3. The sum of ranks for all sample groups are calculated. (R_i)
4. Calculate the test statistic

5.

6. Reject H_0: mean ranks of k groups do not differ if

In Steps 4 and 5, n_i stands for the i^{th} sample size for k samples, N stands for the total number of observations which is the sum of all n_i's from 1 to k, R_{ij} is the rank of individual observations and R_i is the sum of ranks of the observations within a group.

When comparing a number of different parameter values, the Kruskal-Wallis test may arrive at the conclusion that all the sampled populations are not identical. However Kruskal-Wallis does not answer the question of which populations are different than the others. In order to answer this question multiple comparisons can be performed. However there exists a problem in keeping the stated significance level (α), used in Kruskal-Wallis, while making C independent comparisons. Because the number of multiple comparisons made (C) effects the overall probability of making only correct decisions at 1-α, when the null hypothesis of no difference among populations is true, in order to adjust this problem Dunn's (Snell, 1983) procedure is used.

The Dunn's comparison formula uses the z-table and each pairwise comparison is performed according to following formula:

For samples i and j of like size; k is the number of samples (different parameter settings), is the mean of the ranks of a sample and N is the total number of observations. In Equation 16 if the difference of mean of ranks are larger than the right-hand-side of the inequality, then test is significant at α level.

It should be noted here that instead of Kruskal Wallis test, Mood's median test could also have been used. Mood's median test is more robust than is the Kruskal-Wallis test against outliers, but is less powerful for data from many distributions.

6.3 Non-Parametric Based Genetic Algorithm

The Non-Parametric Based Genetic Algorithm (NPGA) uses similar genetic operation and coding as the GA used in section 5.2.2. The difference between the NPGA developed here and GA developed in section 5.2.2 is during the selection of parents for reproduction which is done by tournament selection. In NPGA, when two different individuals representing two parameter settings are compared to be parents in tournament selection, the winner is selected by looking at the distribution of fitness values, which is the solution value of the combinatorial optimization problem, instead of comparing one fitness value from each individual as done in the GA described in section 5.2.2. NPGA compares parameter settings with each other to determine if they are statistically better than one another by using the non-parametric methods described in section 6.2. The comparison Mann-Whitney test is used to compare two individuals (parameter settings) from 10 best solutions selected out of 100 iterations. After the tournament selection, the

NPGA continues to the regular GA's genetic operations as described in section 5.2.2. NPGA also uses blend crossover and random mutation for these two procedures since they are common and effective procedures for the real-coded (chromosomes represented with real values as opposed to binary coding procedure which uses binary strings)) GAs. Although the GA does not assume normality within its procedure, during parent selection, it compares two different parameter settings and does not use a non-parametric comparison. Therefore NPGA enhances and modifies the GA procedure to use non-parametric comparisons. The flowchart of the NPGA procedure is given in Figure 9.

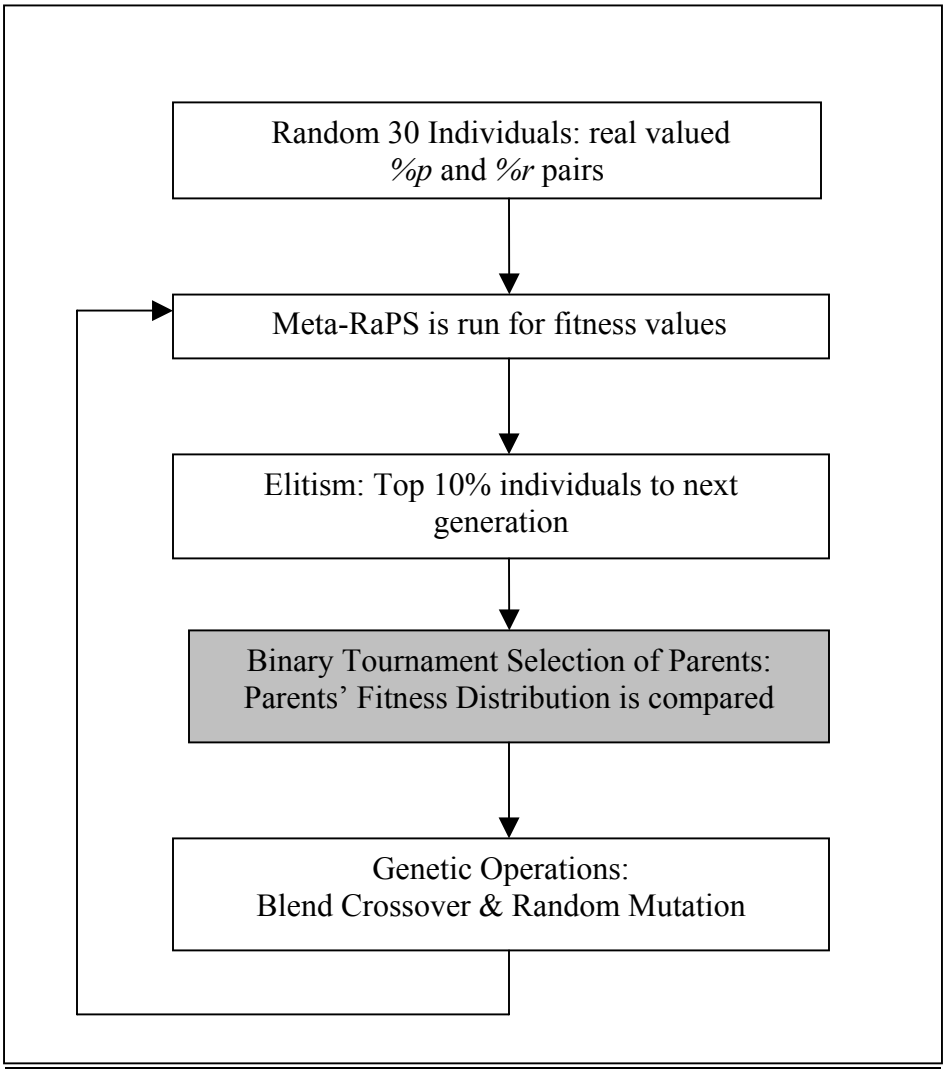

Figure 9: Flowchart of NPGA

For the experimentation in this chapter, the NPGA uses 30 individuals for the population with 90% crossover and 50% mutation rates. These parameters are taken from the literature on GA (Deb, 2002) and verified by experimentation. Based on experimentation, the performance of the NPGA parameter tuned Meta-RaPS is robust to the small changes in the NPGA parameters' values used. In other words, NPGA yields consistent results even if NPGA parameters used are varied in a narrow range from the literature.

It should be noted that the importance of this new non-parametric GA method (NPGA) comes from both the performance of the parameter setting methods. As well as it is sound, the NPGA is the correct procedure to use since the distribution of the best solution values is rarely normally distributed.

NPGA can be used for any type of meta-heuristic method. It is not specific to Meta-RaPS. The method can be extended to more than two parameters and the parameter ranges do not have to be percentage values. The flexibility of GA enables different types of parameters to be set by NPGA for any meta-heuristic procedure that requires parameter tuning.

6.4 NPGA Results

Similar to the parameter setting methods mentioned in Chapter 5, NPGA is tested for both 0-1 MKP and ETP. Tables 24 and 25 compare the results of different parameter techniques for 0-1 MKP and ETP problems respectively. The comparison between different methods is done using Kruskal-Wallis (K-W) and multiple comparison tests as described in Section 6.2. If K-W test is found significant, then multiple pairwise

comparisons are performed using Dunn's method. For these comparisons a significance level of 0.05 is selected.

The numbers shown in Tables 24 and 25 are the counts of problems for a parameter setting technique that gives statistically better distribution of solution values (solution performance) than others. For each problem set all 30 problems are tested. In some cases Kruskal-Wallis (K-W) test may not be significant, meaning that all the parameter setting methods had similar performance. The far right column in Table 24 shows the number of problems from the set of 30 in which the K-W test is significant (i.e. when at least one parameter setting method's solution performance is better). When the K-W test is significant, there are cases when the solution performance is optimized by more than one parameter setting techniques. In other words, a subset of parameter setting methods provide the best performance and their performance is indifferent when compared within each other. In Tables 24 and 25 compare all five parameter setting techniques discussed in Chapter 5 (including trail-and-error, labeled as TE) to NPGA.

Table 24 shows for 54 problems of the 270 MKP problems compared, at least one parameter setting method yielded a significantly different solution value than other parameter setting techniques. For these 54 problems, the Meta-RaPS results using the NPGA parameter setting technique are statistically better or similar to the other parameter setting techniques in most (50) of these problems.

Table 24: Comparison of Parameter Setting Methods for 0-1 MKP

Problem Set		Technique is in the Best Techniques List						K-W Test significant
m	n	TE	RS	RSM	R&S	GA	NPGA	
5	100	3	4	5	3	6	6	6
5	250	2	5	6	2	7	8	10
5	500	2	6	4	2	3	7	9
10	100	2	5	5	2	5	7	7
10	250	2	1	4	5	5	5	5
10	500	0	2	2	3	3	4	4
30	100	2	2	3	0	1	3	3
30	250	0	0	1	0	1	2	2
30	500	2	5	8	3	8	8	8
Total Count		15	30	38	20	39	50	54

Similar to the 0-1 MKP results shown in Table 24, Table 25 suggests that NPGA is consistently able to set the best parameters for ETP as well. For 41 problems of the 240 ETP problems compared, at least one parameter setting method yielded a significantly different solution value than other parameter setting techniques. It is observed in Table 25 the 10 and 15 Job ETP problems are not parameter sensitive and their parameters can be set with simple parameter setting methods such as running small experiments. However for the larger size problems, the problems get harder for Meta-RaPS to solve and the parameter tuning for Meta-RaPS gains more importance.

Table 25: Comparison of Parameter Setting Methods for ETP

Problem Set		Technique is in the Best Techniques List						K-W Test
n	Level	TE	RS	RSM	R&S	GA	NPGA	significant
10	Low	0	0	0	0	0	0	0
10	Medium	0	0	0	0	0	0	0
10	High	0	1	0	1	1	1	1
15	Low	0	0	0	0	0	0	0
15	Medium	0	0	1	0	1	0	0
15	High	2	1	2	2	1	2	2
20	Low	2	3	3	3	3	3	3
20	Medium	0	1	1	1	1	2	2
20	High	2	1	2	2	2	3	3
25	Low	2	2	4	1	3	4	5
25	Medium	2	2	2	2	4	4	4
25	High	1	1	2	2	1	2	3
40	Low	3	2	4	2	3	4	5
40	High	1	3	3	1	3	3	3
50	Low	2	1	3	5	3	4	4
50	High	3	4	5	4	5	6	6
Total Count		20	22	32	26	31	38	41

CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH

This dissertation offers a comprehensive comparative study of several established meta-heuristic parameter setting techniques as well as developing a new parameter setting method. Although parameter setting is required for most meta-heuristics, it has not been addressed extensively in literature.

Although the proposed parameter setting technique, NPGA, is tested on Meta-RaPS, it is applicable to any meta-heuristic that has parameters. The experimentation from several different combinatorial problems reveals that the distribution of a best solution of a meta-heuristic taken from a large number of iterations rarely follows a mound-shaped and/or normal distribution. The distribution of best solution, from a given number of iterations, for 87% of the 240 ETP test problems and 94% of the 270 0-1 MKP test problems are found not to be normally distributed. Therefore when using a parameter setting technique, that compare different parameter setting levels, a distribution-free method should be used, hence the development of NPGA.

The proposed method, NPGA, integrates GA with non-parametric tests and is able to set efficient parameters. NPGA is compared to five existing parameter setting methods for both ETP and 0-1 MKP problems. In the vast majority of the problems, NPGA method consistently proposed the best parameters found by any of the parameter setting studied. For ETP, NPGA provided best 38 parameter settings out of the 41 problems in which there is a statistical performance difference between the parameter setting techniques and for the 0-1 MKP, NPGA provided best 50 parameter settings out of the 54 problems in which there is a statistical performance difference between the parameter setting techniques. In some cases the problems are not parameter sensitive and all of the

parameter setting techniques considered are able to devise similar parameter settings Apart from these problems, for most of the problems the NPGA is able to suggest parameter settings whose solution performance is statistically better than the performance of parameter settings found by other parameter setting techniques. From all the parameter setting techniques investigated, NPGA set parameters give the best solution performance for Meta-RAPS.

In addition to the parameter setting work done within this dissertation, advances were also made on solution algorithms for the two combinatorial optimization problems used to demonstrate the parameter setting techniques; 0-1 Multidimensional Knapsack (0-1 MKP) and the Early/Tardy Single Machine Scheduling Problem with a Common Due Date and Sequence Dependent Setup Times(ETP).

The first combinatorial optimization problem studied in this dissertation is 0-1 MKP which is one of the most studied problems in literature. The Meta-RaPS application of 0-1 MKP uses a new heuristic based on the idea of Cho (2004). The Meta-RaPS 0-1 MKP application yields better solution performance than the other pre-existing Meta-RaPS 0-1 MKP(Moraga, 2005). The application in this research also gives comparable solution performance against other meta-heuristics in the literature except GA (Chu and Beasley, 1998) and ADP (Bertsimas, 2002). The Meta-RaPS gives 0.75% average percent deviation, from best known solutions for the 270 0-1 MKP test problems, where other Meta-RaPS, GA, ADP and second GA application for 0-1 MKP by Haul and Voss (1997), yields 0.77%, 0.53%, 0.74% and 0.93% respectively.

The second combinatorial optimization problem studied in this research effort is ETP which is a special case of single machine scheduling problem. Meta-RaPS utilizes SAPT (Rabadi, 1999) heuristic and enhances its solution quality by introducing

randomness. The ETP Meta-RaPS is compared against the simulated annealing (SA) meta-heuristic by Rabadi (2004) and is able to give better solution performance on comparable CPU time. Out of 60 larger size (40 and 50 job size problems) ETP problems compared, Meta-RaPS provided better solution performance than SA for 55 of the problems. For the smaller set, up to 25 job problem size, in all four different problem sets, each of which have 15 problems generated, the Meta-RaPS solution performance outperformed SA and SA-SAPT algorithms.

While conducting this research several areas of future work were identified. During the parameter setting techniques comparison, it is identified that most problems are either not parameter sensitive or their parameters does not need any sophisticated parameters setting methods, meaning their parameters can be set by simple trial error methods. Future research can be directed to analysis of the parameter sensitivity of the problems by investigating their structure (e.g., correlation structure of a problem, size of problem). Apart from the problem challenged in this dissertation another gap found in literature of heuristics and / or meta-heuristics is how to combine the different parameter settings for different problems and come up with parameters that yield good performance independent of the problem. Another future research topic comes from a practical application point of view, given a limited amount of time, the discussion of how much of this time should be spend on parameter selection and how much time should be spared to run the Meta-RaPS with the setting found, should be made. Another potential area of parameter setting research is whether to use a single combination of parameter settings, or multiple combinations. Given a number of iterations, is the best strategy to solve a problem to use all the iterations with the best setting found from a parameter setting technique or to partition the number of iteration to n best parameter setting combinations.

In addition to parameter setting related research topics, additional modification to Meta-RaPS itself should be considered. Because of its simplicity, Meta-RaPS is open to modifications in its procedure and is able to use some of the strategies that other meta-heuristics use. Currently the parameters *%restriction* and *%priority* are kept constant during each iteration. Similar to simulated annealing (see section 2.2.2), Meta-RaPS could modify the degree of randomness within an iteration. The underlying idea used by SA is that during the initial stages of the iteration, the algorithm is allowed to explore the solution space and do a random search. However, when the iteration is close to the end stages of constructing a solution, the greedy rule may dominate and the algorithm may limit randomness. Meta-RaPS procedure can make use of this idea SA uses and guide the search to have a varied level of randomness during different parts of the search. Introduction of updating parameter settings during an iteration is valuable for Meta-RaPS because this makes Meta-RaPS a dynamic procedure. In other words dynamic parameter driven Meta-RaPS will not require any pre-parameter setting phase and thus Meta-RaPS procedure will be a holistic procedure that modifies, sets and guides its own parameters.

LIST OF REFERENCES

Agogino A., Stanley K., Miikkulainen R., (2000), Online Interactive Neuro-evolution, Neural Processing Letters, vol. 11, no 1., pp. 29-38.

Batiti R., (1996), Reactive Search: Toward Self-Tuning Heuristics, In Rayward-Smith V.J., Osman I.H., C.R, Reeves C.R., Smith G.D.,(eds), Modern Heuristic Search Methods, (1996), John Wiles & Sons.

Baykasoglu A., (2004), A Meta-heuristic Algorithm to Solve Quadratic Assignment Formulations of Cell Formation Problems without Presetting Number of Cells, Journal of Intelligent Manufacturing, vol. 15, no. 6, December, pp. 753-759.

Beasley, J.E., 1990, OR-Library: Distributing Test Problems by Electronic Mail, Journal of the Operational Research Society, vol. 41, pp. 392-404.

Benyahia I., Potvin J. Y., (1995), Generalization and Refinement of Route Construction Heuristics using Genetic Algorithms, , IEEE International Conference on Evolutionary Computation, vol.: 1, 29 Nov.-1 Dec., pp. 39-43.

Bertsimas, and Demir, R., (2002), An Approximate Dynamic Programming Approach to Multidimensional Knapsack Problem, Management Science, vol. 48, no.40, pp. 550-565.

Black P. E., (1998), Dictionary of Algorithms and Data Structures, National Institute of Standarts and Technology, http://www.nist.gov/dads/HTML/greedyalgo.html.

Brown J.T., (1995), Priority Rule Search Technique for Resource Constrained Project Scheduling, Ph.D. dissertation, Orlando, FL: University of Central Florida.

Brucker P., Knust S., Schoo A., Thiele O., (1998), A Branch and Bound Algorithm for Resource Constrained Project Scheduling Problem, European Journal of Operational Research, vol. 107, pp. 272-288.

Chatterjee S., Carrera C., Lynch L., (1996), Genetic Algorithms and Traveling Salesman Problem, European Journal of Operations Research, vol. 93, pp. 490-510.

Chen, Z-L., (1997), Scheduling with Batch Setup Times and Earliness-Tardiness Penalties, European Journal of Operational Research, vol. 96, pp. 518-537.

Cho Y. K., Moore J.T., Hill R.R., (2004), Empirical Analysis for the Multidimensional Knapsack Heuristics Based on the Various Correlation Structures, Proceedings of the 9th Annual International Conference on Industrial Engineering – Theory, Applications and Practice, The University of Auckland, November 27-30.

Cho, Y. K., J. T. Moore, and R. R. Hill, (2004), Insights Gained via an Empirical Analysis of Multidimensional Knapsack Problems, Proceedings of the 2004 Industrial Engineering Research Conference, Houston, TX, May 16-19.

Chu P.C., Beasley J.E., (1998), A Genetic Algorithm for the Multidimensional Knapsack Problem, Journal of Heuristics, vol. 4, pp. 63-86.

Cohen D. M., Dalal S. R., Fredman M. L., Patton G. C., (1997), The AETG System: An Approach to Testing Based on Combinatorial Design, IEEE Transaction on Software Engineering, vol. 23, no. 7, pp. 437-444.

Coleman, B. J., (1992), A Simple Model for Optimizing the Single Machine Early/Tardy Problem with Sequence-Dependent Setups, Production and Operations Management, vol. 1, pp. 225-228.

Connoly D. T., (1990), An Improved Annealing Scheme for the QAP, European Journal of Operational Research, vol.46, pp.93-100.

Coy S. P., Golden B. L., Runger G. C., Wasil E. A., (2000), Using Experimental Design to Find Effective Parameter Settings for Heuristics, Journal of Heuristics, vol. 7, pp. 77-97.

Daniel W.W., (1990), Applied Nonparametric Statistics, PWS-KENT Publishing Company, Second Edition, Boston.

Deb K., (2001), Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, West Sussex, England.

Deb K., Pratap A., Agarval S., Meyarivan T.A., (2002), Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computing, vol. 6, no. 2 pp. 182-197.

DePuy G.W., Whitehouse G.E., (2001), A Simple and Effective Heuristic for the Resource Constrained Project Scheduling Problem, International Journal of Production Research, vol. 39, no. 14, Sep 20, pp. 3275-3287.

DePuy G.W., R.J. Moraga, Whitehouse G.E., (2005),Meta-RaPS: A Simple And Effective Approach For Solving The Traveling Salesman Problem, Transportation Research Part E: Logistics and Transportation Review, vol. 41, no. 2, pp. 115 130.

Dammeyer F., Voss S., (1993), Dynamic Tabu List Management Using the Reverse Elimination Method, Annals of Operations Research, vol 41, pp. 31-46

Delmaire H., Diaz J. A., Fernandez E., (1999), Reactive Grasp and Tabu Search Based Heuristics for the Single Resource Capacitated Plant Location Problem, INFOR, vol. 37, pp. 194-225

Feo T.A., Resende M.G.C., Smith S.H., (1994), A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set, Operations Research, vol. 42, pp. 860-878.

Feo T. A., Resende M.G.C., (1995), Greedy Adaptive Search Procedures, Journal of Global Optimization, vol. 6, pp. 109-133.

Feo T. A., Sarathy K., McGahan J., (1996), A Grasp for Single Machine Scheduling with Sequence Dependent Setup Costs and Linear Delay Penalties, Computers & Operations Research, vol. 23, no. 9, pp. 881-895.

Festa P., Resende M.G.C.,(2001), GRASP: An Annotated Bibliography. Appear in "Essays and Surveys on Meta-heuristics", Hansen P., Ribeiro C.C. eds., Kluwer Academic Publishers

Gilmore, P.C. and Gomory, R.E., 1966, The Theory and Computation of Knapsack Functions, Operations Research, vol. 14, pp. 1045-1074.

Glover F., (1991), Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem, ORSA Journal on Computing.

Glover F., Kochenberger G.A., (1995), Critical Events Tabu Search For Multidimensional Knapsack Problems, in: I.H. Osman, J.P. Kelly(Eds.), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, Dordrecht, 1995, pp. 407-428.

Glover F., Laguna M., (1997), Tabu Search, Kluwer Academic Publishers.

Golden B., Pepper J., Vossen T., (1998), Using genetic algorithms for setting parameter values in heuristic search, Intelligent Engineering Systems Through Artificial Neural Networks, vol. 8, pp. 239-245.

Goldberg D.E. and Smith R.E. (1987) Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy. ln J. J. Grefenstette(editor), Proceedings of the 2nd International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 59-68.

Goldberg D. E., (2002), Genetic Algorithms in Search Optimization & Machine Learning, Addison Wesley.

Gomes F. C., Pardalos P., Oliveira C. S., Resende M. G. C.,(2001), Reactive GRAPS with Path Relinking for Channel Assignment in Mobile Phone Networks, Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications,pp. 60-67.

Grabowski J., Wodecki M., (2004), A Very Fast Tabu Search Algorithm for the Permutation Flow Shop Problem with Makespan Criterion, Computers and Operations Research, vol. 31, no. 11, September, pp. 1891-1909.

Grefenstette, J. J., (1986), Optimization of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-16 n 1 Jan-Feb 1986. pp. 122-128.

Hansen, P., (1986), The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming, Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

Haul C., Voss S., (1997), Using Surrogate Constraints in Genetic Algorithms for Solving Multidimensional Knapsack Problems. In D.L. Woodruff, Advances in computational and stochastic optimization, logic programming and heuristic search. Interfaces in computer science and operation research (pp. 235-251) Dordecht: Kluwer Academic Publishers.

Hino C.M., Ronconi D.P., Mendes A.B., (2005), Minimizing Earliness and Tardiness Penalties in a Single-Machine Problem with a Common Due Date, European Journal of Operational Research, vol. 160, no. 1, Jan 1, pp. 190-201.

Hollander M., Wolfe D., (1999), Nonparametric Statistical Methods, John Wiley and Sons, NY.

Kellerer H., Pferschy U., Pisinger D. (2004), Knapsack Problems, Springer Publications.

Khattab M., Choobineh F., (1991), A New Approach for Project Scheduling with a Limited Resource, International Journal of Production Research, vol. 29, pp. 185 198.

Koksalan M., Ahmet K.B., (2003), Using Genetic Algorithms for Single-Machine Bicriteria Scheduling Problems, European Journal of Operational Research, vol. 145, no, 3, Mar 16, pp. 543-556.

Kolisch R., Drexl A., (1996), Adaptive Search for Solving Hard Project Scheduling Problems, Naval Research Logistics, vol. 43, pp.23-40

Kolisch R., Specher A., (1996), PSPLIB- A Project Scheduling Problem Library, European Journal of Operational Research, vol. 96, pp. 205-216

Kuik R.,Salomon M. (1990), Multi-Level Lot-Sizing Problem: Evaluation of a Simulated Annealing Heuristic. EJOR, vol. 45, pp.25-37

Laguna M., Barnes J.W. and Glover F. (1991) Tabu Search Methods for Single Machine Scheduling Problem, Journal of Intelligent Manufacturing, vol. 2, pp. 63-74.

Lan G., DePuy, G.W., Whitehouse G.E.,(forthcoming), An Effective and Simple Heuristic for the Set Covering Problem, European Journal of Operational Research, Available online 6 December 2005.

Law A.M., Kelton W.D.(2000), Simulation Modeling and Analysis, McGraw Hill Publications.

Lee K., (2005), Simulated annealing by grand canonical ensemble and the TSPs, WSEAS Transactions on Computers, vol. 4, no. 8, August, pp. 890-897.

Li J., Kwan R. S. K., (2002), A Fuzzy Evolutionary Approach with Taguchi Parameter Setting for the Set Covering Problem, CEC '02, Proceedings of the 2002 Congress on Evolutionary Computation, Vol.: 2 ,12-17 May 2002 pp. 1203–1208.

Liepins G. E. and Potter W. D. (1991) A Genetic Algorithm Approach to Multiple-Fault Diagnosis. In Davis L.(editor), Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991, New York,pp. 350-372.

Liu W., Ewins D. J., (1994), Neural Networks: A Method for Joint Dynamic Parameter Identification, Neural Networks for Signal Processing IV. Proceedings of the 1994 IEEE Workshop (Cat. No.94TH0688-2), pp. 633-640.

Maier R. W., Whiting W. B., (1998), The Variation of Parameter Settings and Their Effects on Performance for the Simulated Annealing Algorithm, Computers & Chemical Engineering, vol. 23, pp. 47-62.

Misevicius A., (2005), A Tabu Search Algorithm for the Quadratic Assignment Problem, Computational Optimization and Applications, vol. 30, no. 1, January, pp. 95-111.

Modares A., Somhom S., Enkawa T., (1999), A Self-Organising Neural Network Approach For Multiple Traveling Salesman And Vehicle Routing Problems, International Transactions in Operations Research, vol. 6, pp. 591-606.

Moraga, R.J., Whitehouse G.E., DePuy G.W., Neyveli B., and Kuttuva S., (2001). Meta Raps: An Efficient And Practical Approach For Solving The Traveling Salesman Problem, The 5th International Conference on Engineering Design and Automation, August 5-8, Las Vegas, NV, pp. 191-195.

Moraga R. J., (2002), Meta-RaPS An Effective Solution Approach for Combinatorial Problems. Ph.D. thesis. Orlando, FL: University of Central Florida.

Moraga R. J., Depuy G. W., Whitehouse G. E.,(2002), Using Meta-RaPS Approach to Solve Combinatorial Problems, Proceedings of the 2002 Industrial Engineering Research Conference, Orlando, Florida.

Moraga R. J., Depuy G. W., Whitehouse G. E.,(2003), A Meta-Heuristic Approach for the 0-1 Multidimensional Knapsack Problem. Ln: Y. Dessouky, Proceedings of the 31st International Conference on Computers and Industrial Engineering (pp.173-178). San Francisco, CA.

Moraga, R.J., G.W. DePuy, and G.E. Whitehouse, (2004), A Meta-Heuristic and Oscillation Improvement Strategy for the 0-1Multidimensional Knapsack Problem, Proceedings of the 2004 Industrial Engineering Research Conference, May 15-19, 2004, Houston, Texas.

Moraga, R. J., DePuy, G.W. and Whitehouse, G.E., (2005), Meta-RaPS Approach for the 0-1 Multidimensional Knapsack Problem, Computers and Industrial Engineering, vol.48, pp. 83-96.

Morgan J.N., Sonquist J.A., (1963), Problems in the Analysis of Survey Data, and a Proposal, Journal of the American Statistical Association, vol.58, No.302,pp. 415-434.

Papadimitriou C.H., Steiglitz K.,(1982), Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Inc.

Pham D.T., Karaboga D., (2000), Intelligent Optimization Techniques, Springer-Verlag London Limited.

Pirkul, H., (1987), A Heuristic Solution Procedure for the Multiconstraint Zero-one Knapsack Problem, Naval Research Logistics, 34, pp. 161-172.

Poopalasingam S., Reeves C. R. , Steele N. C.,(1994), Application of Neural Networks for Sensor Performance Improvement, IV. Proceedings of the Neural Networks for Signal Processing 1994 IEEE Workshop ,6-8 Sept. 1994, pp.633-640.

Prais M., Ribeiro C.C., (2000), Reactive GRASP: An Application to Matrix Decomposition Problem in TDMA Traffic Assignment, INFORMS Journal on Computing, vol. 12, pp. 164-176.

Rayward-Smith V.J., Osman I.H., C.R, Reeves C.R., Smith G.D., (1996), Modern Heuristic Search Methods, John Wiles & Sons.

Rabadi G.(1999), Minimizing the Total Earliness and Tardiness for a Single Machine Scheduling Problem with a Common Due Date and Sequence-Dependent Setup Times, University of Central Florida Phd. Dissertation.

Rabadi G., Mollaghasemi M., Anagnostopoulos G.C.,(2004), A Branch-and-Bound Algorithm for the Early/Tardy Machine Scheduling Problem with a Common Due-Date and Sequence- Dependent Setup Time, Computers & Operations Research, vol. 31, pp. 1727-1751.

Rabadi, G ., Anagnostopoulos, G., and Mollaghasemi, M., (forthcoming), A Heuristic Algorithm For The Just-In-Time Single Machine Scheduling Problem With Setups: A Comparison With A Simulated Annealing, to appear in The International Journal of Advanced Manufacturing Technology.

Reeves C.R. and Steele N.C., (1991), A Genetic Algorithm Approach to Designing Neural Network Architecture. Proc. 8th International Conference on Systems Engineering.

Reeves C., (1993), Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons.

Reeves C. R., Wright C. C., (1995), Genetic Algorithms and Statistical Methods: A Comparison, Genetic Algorithms in Engineering Systems: Innovations and Applications, 12-14 Sep., no. 414, pp.137-140.

Reeves C. R., Eremeev A. V., (2004), Statistical Analysis of Local Search Landscapes, Journal of the Operational Research Soceity, vol. 55, pp. 687-693.

Rosen S. L., Harmonosky C. M., (2005), An Improved Simulated Annealing Simulation Optimization Method for Discrete Parameter Stochastic Systems, Computers & Operations Research, vol. 32, no. 2, pp. 343-358.

Schaffer J.D., Caruana R. A., Eshelman L. J., Das R., (1989), A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, 3rd International Conference on Genetic Algorithms.

Schaffer D. J., Elshelman L. J., (1996), Combinatorial Optimization by Genetic Algorithms: The Value of the Genotype/Phenotype Distinction, In Rayward Smith, I., et al. (Eds), Modern Search Methods, John Wiley & Sons.

Senyu, S. and Toyoda, Y., (1968), An Approach to Linear Programming with 0-1 Variables, Management Science, 15, pp. 196-207.

Shih, W., (1979), A Branch and Bound Method for the Multiconstraint Zero-one Knapsack Problem, Journal of Operation Research Society, vol. 30, pp. 369-378.

Skorin- Kapov J., (1990), Tabu Search Applied to the Quadratic Assignment Problem, ORSA, Journal on Computing, vol.2, pp. 33-45.

Skorobohatyj G., (2002). MP-TESTDATA: Integer/ Mixed-Integer Programming Problems. http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/index.html

Snell M.C., (1983). Recent Literature on Testing for Intergroup Concordance, Applied Statistics, vol. 32, pp. 134-140.

Sohrabi B., Bassiri M.H., (2004), Experiments to determine the simulated annealing parameters case study in VRP, International Journal of Engineering, Transactions B: Applications, vol. 17, no. 1, April, pp. 71-80.

Song Y., Zhang Z., Zheng L., (2005), Ant colony optimization for the single machine early or tardy problem with distinct ready times, Journal of Tsinghua University, vol. 45, no. 11, November, pp. 1577-1580.

Tang Q., (2004), Simulated annealing in lot sizing problems, International Journal of Production Economics, vol. 88, no. 2, pp. 173-181.

Taillard E. (1990) Some Efficient Heuristic Methods for the Flowshop Sequencing Problem, European Journal of Operational Research, vol. 47, pp. 65-74.

Toth, P. and Martello, S., (1990), Knapsack Problems: Algorithms and Computer Implementations, Wiley-Interscience Series in Discrete Mathematics and Optimization, England.

Toyoda Y., (1975), "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-one Programming Problems", Management Science, vol. 21. no. 12, pp. 1417-1427.

Van Breedam A., (1995), Improvement Heuristics for the Vehicle Routing Problem Based on Simulated Annealing , European Journal of Operational Research, vol.86, pp.480-490.

Van Breedam A.(1996), An Analysis of the Effect of Local Improvement Operators in Genetic Algorithms and Simulated Annealing for the Vehicle Routing Problem, RUCA Working Paper 96/14, Faculty of Applied Economics, University of Antwerp, Antwerp, Belguim.

Van Breedam A., (2002), A Parametric Analysis of Heuristics for the Vehicle Routing Problem with Side-Constraints, European Journal of Operational Research, vol. 137, pp. 348-370.

Vasko F.J., Knolle P.J., Spiegel D.S., (2005), An Empirical Study of Hybrid Genetic Algorithms for the Set Covering problem, of the Operational Research Society, vol. 56, no. 10, October, pp. 1213-1223

Voudouris C., Tsang E., (1999), Guided Local Search And Its Application To The Traveling Salesman Problem, European Journal of Operations Research, vol. 113, pp. 469-499

Wang Y., Han L., Li Y., Zhao S., (2006), A New Encoding Based Genetic Algorithm for the Traveling Salesman Problem, Engineering Optimization, vol. 38, no. 1, January, pp. 1-13.

Weingartner, H.M., 1967, Mathematical Programming and the Analysis of Capital Budgeting Problems, Markham publishing, Chicago.

Whitley D., Starkweather T., Shanner D.,(1991) The Traveling Salesman and Sequence Scheduling; Quality Solutions Using Genetic Edge Recombination. In Davis L.(editor), Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991, New York,pp. 350-372.

Yang N., Tian W., Jin Z., (2006), Crossover Tabu Search for Traveling Salesman Problem, Journal of System Simulation, vol. 18, no. 4, April, pp. 897-899.

Yuan L., Liu F., Zhao B., (2005), Improved GA and its Application to Knapsack Problem, Systems Engineering and Electronics, vol. 27, n 4, April, 2005, pp. 718-719.