Nonparametric Comparison of Two Dynamic Parameter Setting Methods in a Meta-Heuristic Approach

Seyhun HEPDOGAN, Ph.D.
Department of Industrial Engineering and Management Systems, University of Central Florida
Orlando, Florida 32816, USA

Reinaldo J. MORAGA, Ph.D.
Department of Industrial & Systems Engineering, Northern Illinois University
DeKalb, Illinois 60115, USA

Gail W. DePUY, Ph.D., P.E.
Department of Industrial Engineering, University of Louisville
Louisville, Kentucky 40292, USA

Gary E. WHITEHOUSE, Ph.D., P.E.
Department of Industrial Engineering and Management Systems, University of Central Florida
Orlando, Florida 32816, USA

ABSTRACT

Meta-heuristics are commonly used to solve combinatorial problems in practice. Many approaches provide very good quality solutions in a short amount of computational time; however most meta-heuristics use parameters to tune the performance of the meta-heuristic for particular problems and the selection of these parameters before solving the problem can require much time. This paper investigates the problem of setting parameters using a typical meta-heuristic called Meta-RaPS (Metaheuristic for Randomized Priority Search.). Meta-RaPS is a promising meta-heuristic optimization method that has been applied to different types of combinatorial optimization problems and achieved very good performance compared to other meta-heuristic techniques. To solve a combinatorial problem, Meta-RaPS uses two well-defined stages at each iteration: construction and local search. After a number of iterations, the best solution is reported. Meta-RaPS performance depends on the fine tuning of two main parameters, priority percentage and restriction percentage, which are used during the construction stage. This paper presents two different dynamic parameter setting methods for Meta-RaPS. These dynamic parameter setting approaches tune the parameters while a solution is being found. To compare these two approaches, nonparametric statistic approaches are utilized since the solutions are not normally distributed. Results from both these dynamic parameter setting methods are reported.

Keywords: Dynamic Parameter Setting, Meta-RaPS, 0-1 Multiple Knapsack Problem, Early Tardy Single Machine Scheduling Problem, Reactive Search.

1. INTRODUCTION

Almost all meta-heuristics have a number of parameters which need to be fine tuned. A general purpose meta-heuristic's performance, such as SA, GA, Tabu Search and Meta-RaPS, is dependent on the values chosen of these parameters. As an example, meta-heuristics that have been used to solve the Vehicle Routing Problem (VRP) contain several parameters; from 4 parameters to 25 parameters depending on the type of meta-heuristic [1]. Coy [1], in his heuristic parameter setting literature review, states that there are many different procedures to find effective parameter settings. Also the complexity of procedures are various, from simple trial-and-error procedures to more sophisticated sensitivity analysis and use of other meta-models.

Ideally the parameter selection method should be fast, efficient and should be capable of enhancing the performance of the heuristic method with respect to using a simpler parameter search or random parameters. The amount of human effort, expertise and experience required for the parameter setting procedure and the computational time used to set parameters should be minimal. Apart from these attributes, a parameter setting procedure can be dynamic or static which is also called online or offline and adaptive or not-adaptive. A dynamic, online, or adaptive 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 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 flexibly and ease of use of dynamic parameter settings provides an advantage over the non-dynamic parameter setting techniques.

For the experimentation of dynamic parameter setting procedures in this research, the Meta-RaPS meta-heuristic is applied to two combinatorial optimization problems: 0-1 Multiple Knapsack Problem and Early Tardy Problem. Although only Meta-RaPS is used in this paper for the

comparison of two dynamic parameter setting techniques, both of these techniques are applicable to other meta-heuristics as well.

2. META-RAPS

Meta-heuristic for Randomized Priority Search (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 [2]. Meta-RaPS has been successfully applied to many classical combinatorial optimization problems such as: 0-1 Multidimensional Knapsack Problem [2], Set Covering Problem [3], Traveling Salesman Problem [4] and Resource Constraint Project Scheduling Problem [5].

In general, Meta-RaPS includes two stages during one iteration: construction and improvement. In the construction stage a feasible solution is built by adding elements to a solution based on a priority rule. The construction stage ends after no possible elements can be added to solution. If the construction stage solution is found promising (i.e., the construction stage solution passes the improvement stage criteria) then it is improved, otherwise the construction stage solution is reported.

In the improvement phase, a local search is applied. The local search may use a similar priority rule as the construction stage priority rule or a completely different search procedure such as 2-opting [3]. After a number of iterations Meta-RaPS report the best solution found.

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 the best solution from *I* iterations is reported. 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 the restriction percentage, %*r,* parameter. In 100-%p of the time the next element added is randomly chosen from the feasible elements whose priority values are within %r of the best priority value. The improvement percentage, *%i,* decides if the solution created at the construction stage is worthy to be improved. The solution from the construction stage is improved if the solution value of the construction stage is within *%i* of the best unimproved solution value found so far, i.e. the best to-date construction solution value [6]. The solution quality of Meta-RaPS depends on the number of iterations *I,* priority *%p,* restriction *%r* and improvement *%i* parameters. For Meta-RaPS parameter setting purposes, the higher values of *I* and *%i* parameters are always preferred and the value of these parameters depend on the availability of the computation time. However the *%p* and *%r* parameters should be tuned for the specific applications and problems. Since it has been demonstrated empirically that good construction solutions produced good improvement solutions, these %p and %r are the key parameters to produce good quality solutions [2].

For this research, Meta-RaPS is applied to two combinatorial optimization problems: 0-1 Multiple Knapsack Problem and the Early Tardy Problem.

0-1 Multi Dimensional Knapsack Problem (0-1 MKP)

The 0-1 Multidimensional Knapsack Problem (0-1 MKP) is one of the most studied combinatorial optimization problems. The objective of the problem is to maximize the profit or the total worth of the objects in the knapsack within its restrictions or capacity. The items to be selected are different in weight and profit. The knapsack has a set of *m* capacity constraints. The knapsack constraints can be thought of as size or weight constraints which are of less than or equal to some upper limit. For any particular 0-1 MKP instance, to fill the knapsack there are *n* different types of items available. Each item has one profit (c_i) value for the knapsack and one weight (a_{ij}) value for each constraint. The decision to include or exclude an item is formulized by binary decision variables. The formulation of 0-1 MKP is as follows:

$$Maximize \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$Subject\ to \quad \sum_{i=1}^{n} a_{ij} x_i \le b_j \qquad \forall j = 1,..,m \tag{2}$$

$$x_i \in \{0,1\} \qquad \forall i = 1,..,n$$

$$where \quad x_i = \begin{cases} 1 & if\ included \\ 0 & otherwise \end{cases} \tag{3}$$

There are many simple greedy heuristics available for 0-1 MKP in literature. The priority rule selected for this Meta-RaPS 0-1 MKP application ranks all items based on a profit/weight ratio. This ratio represents the desirability of an item to be included in the knapsack. This ratio, called the pseudo-utility ratio, is $\alpha_i = c_i/w_i$; where w_i is the penalty factor (or weight) for item i. The w_i is usually a function of a_{ij}. After the pseudo-utility ratios are calculated for each item, the items are ordered in decreasing ratio order and the ordered items are included in the solution one by one as long as they do not violate any constraints.

In this application, the weights are calculated using the normalization of weights idea introduced by Cho [7]. This weighting scheme uses a lognormal point function for the weight vector. This transformation gives more weight to the constraint with the least resource remaining. In this way the priority rule selects items that use the scarce constraints (resources) more effectively. The idea from Cho [7] is combined with a new weight ratio which has the following weight formula:

$$w_i = \sum_{j=1}^{m} \left(a_{ij} \cdot \sum_{j=1}^{m} \exp\left(\sigma \Phi^{-1}\left(CW_j / b_j\right)\right) \right) \tag{4}$$

In Eq. 4, Φ^{-1} is the inverse of standard normal cumulative density function and the σ parameter, which determines the shape of the lognormal point function, is set as 3 by trial and error. The term inside the Φ^{-1} function (CW_j / b_j) is normalized by dividing the amount of resources used by the initial capacity of the knapsack constraints. In other words, CW_j is the amount of j^{th} resource consumed by the items assigned so far. a_{ij} is the weight of item *i* for the constraint *j*.

For the 0-1 MKP Meta-RaPS implementation, during each iteration, the parameter %*p* is used to define the percentage of

time the next item to be included in the knapsack is selected based on the greedy rule, which is the pseudo-utility ratio. In the remaining 100-%*p* of the time the items that are %*r* close to the item with the highest pseudo-utility ratio value are included in a candidate list and next item to be added to knapsack is randomly selected from this list.

For the Meta-RaPS improvement stage, the exchange neighborhood search is used. The exchange neighborhood improvement tries exchanging all possible combinations of items in the solution with the items that are not in solution. The improvement stage in Meta-RaPS 0-1 MKP application has two different exchange improvement procedures; two-way and one-way exchange improvement. Initially any possible two items which are already included in the solution are exchanged with any two items that are not in the solution. If the solution value improves the exchange is accepted, otherwise the exchange is rejected. After two-way exchange, a one-way exchange improvement procedure is conducted.

Early/Tardy Single Machine Scheduling Problem

The second Meta-RaPS application is for the Early/Tardy Single Machine Scheduling Problem with Common Due Date and Sequence-Dependent Setup Times (ETP). The objective of this machine scheduling problem is to minimize the sum of earliness and tardiness of the jobs which are assigned to a single machine at a common due date [8]. The objective function of ETP is given in Eq. 5.

$$\min Z = \sum_{j=1}^{n} \left(E_j + T_j \right) = \sum_{j=1}^{n} \left| d - C_j \right| \tag{5}$$

The conditions and assumptions of the problem are as follows: all jobs are available to be assigned at time zero, each job has a processing time p_j, a common due date d, and a sequence-dependent setup time s_{ji} which depends on the predecessor job. The machine is able to work on one job at a time without preemption. Eq. 5 C_j calculates the completion time of a job.

Unlike 0-1 MKP, ETP does not have many simple priority rules available. For the Meta-RaPS ETP application, Shortest Adjusted Processing Time (SAPT) heuristic [9] is used as the underlying priority rule for Meta-RaPS. SAPT is a simple, effective multi-start heuristic. Similar to Meta-RaPS meta-heuristic, the SAPT heuristics constructs a number of schedules and reports the best solution. The SAPT heuristic is based on the solution of ETP without setup times [9].

The optimal schedule for the Early/Tardy problem without setup times follows Longest Processing Time (LPT) order for the early schedule and the Shortest Processing Time (SPT) order for the tardy schedule [9]. This type of a schedule is called V-shaped schedule.

For the sequence dependent setup time case, the V-shaped scheduling does not guarantee optimality because the sequence dependence may force a job to be scheduled that does not follow LPT rule before the due date or SPT rule after the due date for the optimal job schedule. 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 [9] based his heuristic on this concept and developed SAPT which builds V-shaped like solutions.

The SAPT builds three different schedules, E, T and ET. In the case of the E schedule, initially jobs before the due date are scheduled (early schedule) and then the remaining jobs are scheduled after the due date (tardy schedule). In the T schedule, initially the tardy schedule is built and then the early schedule is constructed. The ET schedule starts building solutions from the due date. The early or tardy jobs are added to the solution one by one depending on which job increases the objective function the least. In other words ET schedule simultaneously builds both Early and Tardy schedule directions at the same time.

In Meta-RaPS ETP application, Instead of using three different types of schedule forming strategies (E, T and ET schedules), only the ET scheduling is implemented. Rabadi [1] reports that in the SAPT procedure one third of the best solutions come from each of the E, T and ET schedules. It is expected that the randomness introduced by Meta-RaPS will be able to construct the E and T schedule solutions as well as the ET schedule solutions. Furthermore, the use of a single priority rule can demonstrate the search capability of Meta-RaPS without complicating the solution procedure.

In the Meta-RaPS ETP application, *%p* of the time the greedy SAPT ET rule is used. In the remaining *100-%p* of the time the job to be assigned is randomly selected from the list of candidate jobs which are *%r* close to the job with the smallest Adjusted Processing (AP) time, which is the sum of processing time and setup time for a job. Incorporating randomness in SAPT in this manner does not further necessitate running the heuristic *n* different times starting with a job combination that has a different AP. Instead Meta-RaPS is run for a large number of iterations and the randomness produces a variety of different starting job combinations.

Generalized Pairwise Interchange (GPI) improvement is used for the local search as part of the SAPT heuristic to enhance the solution quality further. GPI goes through all the possible two-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 $n(n\text{-}1)/2$ different neighborhoods for a problem size of n jobs [8]. The 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. The interested reader is referred to [10] for more details regarding this Meta-RaPS ETP application.

Testing on 0-1 MKP and ETP

The Meta-RaPS 0-1 MKP and the Meta-RaPS ETP applications are coded in C++ and tested on a P4 2.2 GHz PC.

For 0-1 MKP the application is tested on 270 large-sized test problems from OR-Library [11] which were generated by Chu and Beasley [12]. This problem suite has 9 different problem sets of different sizes. The number of constraints (m) in these problems is set to 5, 10 and 30 and the number of items (n) is set to 100, 250 and 500. For each *n-m* combination, 30 problems were generated by Chu and Beasley [12] for a total of 270 problems. Meta-RaPS 0-1 MKP is run for 1000 iterations for these test problems.

The ETP test problems are generated by Rabadi [13]. These problems have 15 and 25 jobs and 3 different settings (low, medium, and high) of adjusted processing times, which is the

sum of the processing time and the setup time for each job. For each setting and number of job combination, 15 problems are generated by Rabadi [13]. Meta-RaPS 0-1 MKP is run for 3000 iterations for these test problems. Both Meta-RaPS applications are able to give competitive results compared to the literature.

3. BEST SOLUTION DISTRIBUTION FOR 0-1 MKP AND ETP

Before discussing the dynamic parameter setting techniques in section 4, an approach to compare different parameter setting levels of a meta-heuristic is introduced. Almost all the meta-heuristics have a number of parameters to be set [1] and therefore should benefit from the parameter setting techniques developed in this research.

Coy [1] provides a good survey of parameter setting methods. In his review, some of the procedures such as response surface methodology and ranking and selection, assume that the best solutions are normally distributed. This assumption is tested for the ETP and 0-1 MKP problem. Each ETP and 0-1 MKP problem was run for 3000and 1000 iterations respectively and the best solution was recorded. This process was repeated 100 times for each problem. Histograms of the 100 best solutions for all ETP and 0-1 MKP problems were then studied. Sample histograms for an ETP and a 0-1 MKP problem are shown in Figures 1 and 2 respectively. Review of all the best solution histograms for both the Meta-RaPS ETP and 0-1 MKP applications show that the best solution distribution does not follow any particular distribution. In most cases, both Kolmogorov-Smirnov and Chi-Square tests reject the normal distribution fit. In other words, a normal distribution is rarely able to represent the best solution distribution for the two applications studied. The best solution 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 the maximization problems it is better for the meta-heuristics solution distribution to be skewed to the right since this indicates the meta-heuristic is producing the majority of solutions near the optimal (maximum). Similarly, for minimization problems, a distribution skewed to the left is preferred.

Figure 1 shows the best solution distribution for ETP problem 25-7-high. Note this distribution is skewed to the left indicating the meta-heuristic is producing many solutions near the optimal, minimum solution. The other ETP test set problems produced similarly shaped distributions. In general, the best fitting distributions were a beta type distribution.

Figure 2 shows the best solution distribution of the first MKP problem with 5 constraints and 100 items. Almost all the problems in the 0-1 MKP test set show a similar shaped distribution as Figure 2 which is skewed to right. A normal distribution rarely represents the best solution distribution for the OR-Library [11] problems. In the vast majority of cases tested, normality is rejected based on statistical tests.

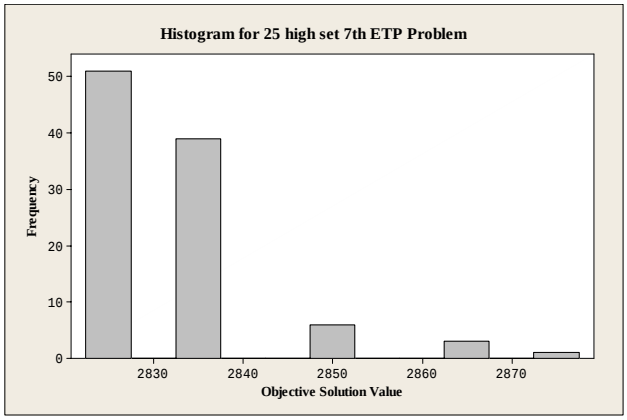

Figure 1: Meta-RaPS ETP 25-7-high Problem Best Solution Distribution

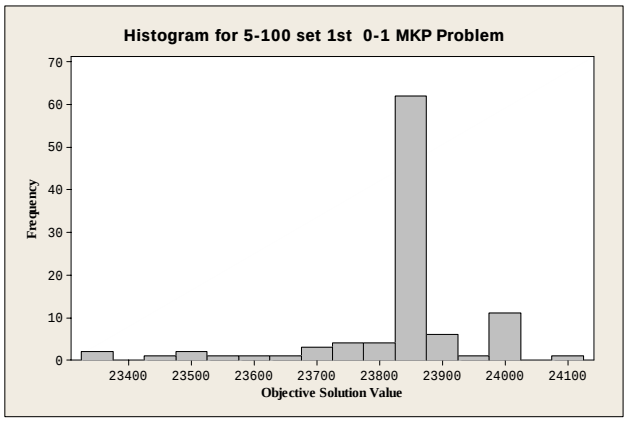

Figure 2: Meta-RaPS 0-1 MKP 5-100-1 Problem Best Solution Distribution

Because the best solutions do not seem to be normally distributed and because the best solutions distribution is dependent on the problem, the use of 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 nonparametric method. Although normality assuming techniques may work and be able to suggest good parameter settings for some cases, in general they are not the proper techniques to represent, or model, the solution distribution of a parameter setting of a meta-heuristic.

4. DYNAMIC PARAMETER SETTING TECNIQUES

This research investigates two different dynamic parameter setting techniques: the Reactive Search (RS) and the Genetic Algorithms (GA).

Reactive Search

The Reactive Search (RS) method uses feedback from the meta-heuristic to set the parameters. RS incorporates a history-based adaptive procedure in the meta-heuristic search for dynamic determination of the parameter levels. History-based learning gradually investigates a variety of parameter settings and determines a probability of selecting each parameter setting for future iterations. Eventually the parameters which lead to the best solutions have a higher probability of being selected by

having higher probabilities. The dynamic setting of parameters eliminates the need for a parameter setting procedure and determines the parameter settings as the meta-heuristic is run. RS procedure has been applied to GRASP [14], which is a meta-heuristic similar to Meta-RaPS.

In the RS procedure, a parameter setting is randomly selected in each iteration from a candidate set of parameters and the meta-heuristic is run with the selected parameter combination. At the end of some predefined fixed number of iterations the probability of selecting a particular parameter setting is calculated based on the performance of that parameter setting with respect to the best performing 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.

Reactive Search had been used as the parameter selection technique within the GRASP heuristic and was found to improve solution performance [14]. Gomes [14] reports that RS procedure gives better results than the GRASP heuristic alone because it is able to determine appropriate values of the parameter(s). Delmaire [15] reports that for some test problems of the Single Source Capacitated Plant Location Problem, RS incorporated GRASP reduced the average deviation from optimal of the 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 Reactive Search seems to provides a good parameter selection procedure since it is a distribution free method and because of its performance on the GRASP heuristic which is a similar meta-heuristic to Meta-RaPS.

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

1. Candidate parameter selection: 9 levels of both *%p* and %r parameters, both starting from 10 to 90, with 10 increments. Therefore 81 %p and %r combinations in total. $C=\{c_1, \ldots, c_{81}\}$
2. Each parameter setting combination in set C is set to have equal probability of being selected
$$p_i = 1/81 \quad 1 \le i \le 81 \tag{6}$$
3. Meta-RaPS is run for 1000 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 $\bar{A}= \{a_1, \ldots, a_n\}$
4. After every 1000 iterations q_i, values are updated according to Eq. 7.
$$q_i = \frac{f(s^*)}{a_i} \quad 1 \le i \le 81 \tag{7}$$
*where s * is best solution found so far*
5. The probabilities are updated at the last step by Eq. 8.
$$p_i = \frac{q_i}{\sum_{j=1}^{n} q_i} \quad 1 \le i \le 81 \tag{8}$$
6. The procedure is terminated at 5000 iterations. It may also be terminated when there is no change in p_i values for a predefined number of iterations.

Non-Parametric Based Genetic Algorithms (NPGA)

Genetic algorithms (GA) are approaches to solve combinatorial optimization problems based on natural selection and evolution mechanisms. GA can also be used as a parameter setting procedure [16, 17]. Grefenstette [16] points out that for the parameter search 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. For these types of applications GA is able to perform well. GA was originally developed by John Holland at University of Michigan to mimic natural selection and evolution. The main theme of the GA is robustness. Nature laws and evolution are thought to be the key element of robustness where survival of the fittest is of primary priority.

Different from most other optimization methods, GA uses a population of solutions to evaluate the performance of a system based on a fitness function value and screens the population for fit individuals that are more likely to survive. The underlying idea is that the mating of fit individuals will result in a yield of better fit offspring. The local optimum is avoided and search diversity is maintained by "genetic operations" which are random selection procedures, crossover and mutation [18].

The type of GA proposed in this research as a dynamic parameter setting procedure is called Non-Parametric Based Genetic Algorithm (NPGA). NPGA uses similar genetic operations as real value coded GAs. The difference between NPGA and the regular GA is in the selection of parents for reproduction, which is usually done using tournament selection. In the case of NPGA, when two different individuals, representing two parameter settings, are compared to be parents in tournament selection, the winner is selected using a statistical comparison of the distribution of fitness values which is the best solution value distribution of the combinatorial optimization problem. The reason for this type of comparison is as follows: for a given parameter setting level, Meta-RaPS may give different answers when it is run multiple times because of randomness, and therefore the quality of a parameter setting combination can only be represented by a distribution of values. Furthermore, to be able to conclude that one parameter setting level gives higher or lower values than another setting level, the comparison has to be made by comparing the distribution of solution values taken by these parameters due to randomness incorporated within Meta-RaPS. NPGA compares parameter settings with each other to determine if they are statistically better than each other by using non-parametric methods. Again, a non-parametric method is used is because the best solution distribution for a parameter setting level may not be normally distributed as was discussed in section 4. The Mann-Whitney nonparametric 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 the regular GA's genetic operations. NPGA also uses blend cross over and random mutation [19]. These two procedures are shown to be common and effective procedures for the Real-coded GA [19].

For the experimentations, the NPGA used in this study uses 30 individuals for the population with 90% crossover and 50% mutation rates. These parameters are taken from literature [19]

and they are verified by experimentation. The trials made using NPGA parameters showed that performance of the NPGA parameter tuned Meta-RaPS is robust to small changes in the NPGA parameters' values. In other words, NPGA yields consistent results even if the NPGA parameters used are varied in a narrow range from the literature.

It should be noted that the importance of this new GA method (NPGA) comes from both the quality of results (i.e. the performance of the parameter setting methods) as well as its theoretical soundness in that the distribution of the best solution values is rarely normal distributed.

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

5. RESULTS

The two dynamic parameter setting techniques: NPGA and RS are tested for both the 0-1 MKP and ETP problem sets. Tables 1 and 2 compare the results of different parameter techniques for the 0-1 MKP and ETP problems respectively. For the experimentation, both the 0-1 MKP and ETP problems are run for 20 replications of 1000 iterations for each parameter setting considered. The best solution is recorded for each replication and in this way a distribution of best solutions can be calculated for each parameter setting level. The comparison between different parameter methods is done using the Mann-Whitney test. While doing this comparison, a significance level of α=0.05 is selected for the tests.

Tables 1 and 2 indicate which parameter setting technique gives statistically better solution values based on the Mann-Whitney test. If both methods suggested the statistically same parameter levels then the Best method is indicated as "Both." and no method is seen in the Inferior column. If the methods are not statistically equivalent, then the method that gives statistically better solution values is placed in Best column and the method giving worse solution values is placed in Inferior column.

Table 1: Parameter Setting Methods for 0-1 MKP

m	N	Problem #	Best	Inferior
5	100	9	NPGA	RS
5	100	14	Both	-
5	250	12	Both	-
5	250	6	Both	-
5	500	8	NPGA	RS
10	100	9	Both	-
10	100	25	Both	-
10	250	1	Both	-
10	500	5	NPGA	RS
30	100	17	Both	-
30	250	9	NPGA	RS
30	500	10	Both	-

The results of the Table 1 suggest the NPGA parameter set Meta-RaPS results are statistically better or similar to the RS technique.

Table 2: Parameter Setting Methods for 0-1 ETP

Size	Level	Problem #	Best	Inferior
15	High	1	NPGA	RS
15	High	7	Both	-
15	Med	8	Both	-
15	Med	9	NPGA	RS
15	Low	5	Both	-
15	Low	8	Both	-
25	High	4	NPGA	RS
25	High	12	NPGA	RS
25	Med	1	NPGA	RS
25	Med	8	Both	-
25	Low	5	Both	-
25	Low	4	Both	-

Similarly, Table 2 also suggests the NPGA is consistently able to set better or similar parameters than RS. For both combinatorial optimization problems tested, NPGA never resulted in worse performing parameters than RS

6. CONCLUSIONS

In this paper two dynamic parameter setting techniques, Nonparametric Genetic Algorithms (NPGA) and Reactive Search (RS) are compared. The parameter setting techniques are used to set the parameters of Meta-RaPS meta-heuristic and for experimentation purposes two combinatorial optimization problems are used: 0-1 MKP and ETP.

This research effort is based on the premise that the best solution value distribution for a combinatorial optimization problem will, most likely, not follow a normal distribution. Therefore any statistical comparison of parameter setting levels should be made using nonparametric techniques.

Comparing the two dynamic parameter setting techniques considered, NPGA performed better than RS. The parameter setting levels set by NPGA yielded better solution values for both ETP and 0-1MKP problem sets. The dynamic parameter setting techniques have an advantage over static parameter setting techniques in that they make the procedure more robust. They eliminate parameter setting phase and merge this phase with the meta-heuristic's solution building phase.

Future research can be directed to the application of NPGA to different meta-heuristics and investigation of the best solution behavior of different meta-heuristics and applications.

7. REFERENCES

[1] Coy S. P., Golden B. L., Runger G. C., Wasil E. A., "Using Experimental Design to Find Effective Parameter Settings for Heuristics", **Journal of Heuristics**, Vol. 7, 2001, pp. 77-97.

[2] Moraga, R. J., DePuy, G.W. and Whitehouse, G.E., "Meta-RaPS approach for the 0-1 Multidimensional Knapsack Problem", **Computers and Industrial Engineering**, Vol. 48, 2005, pp. 83-96.

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

[4] DePuy, G.W., R.J. Moraga, and G.E. Whitehouse, 2005, "Meta-RaPS: A Simple And Effective Approach For Solving The Traveling Salesman Problem," **Transportation Research Part E: Logistics and Transportation Review**, 41(2), 115-130.

[5] Whitehouse, G.E., DePuy, G.W., Moraga, R.J. Meta-RaPS "Approach for Solving the Resource Allocation Problem", **Proceedings of the 2002 World Automation Congress**, June 9-13, Orlando, Florida.

[6] Moraga R. J., Meta-RaPS An Effective Solution Approach for Combinatorial Problems. **Ph.D. Dissertation. Orlando, FL, University of Central Florida**, 2002.

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

[8] Rabadi G., Mollaghasemi M., Anagnostopoulos G.C., "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, 2004, pp. 1727-1751.

[9] Rabadi, G ., Anagnostopoulos, G., and Mollaghasemi, M., "A Heuristic Algorithm For The Just-In-Time Single Machine Scheduling Problem With Setups: A Comparison With A Simulated Annealing", **International Journal of Advanced Manufacturing Technology,** Vol 32, 2007, pp. 326-335.

[10] Hepdogan, S., R. Moraga, G.W. DePuy, and G.E. Whitehouse. "A Meta-RaPS Solution for the Early/Tardy Single Machine Scheduling Problem," to appear in **International Journal of Production Research**, accepted 2007.

[11] Beasley, J.E., "OR-Library: Distributing Test Problems by Electronic Mail", **Journal of the Operational Research Society**, Vol. 41, 1990, pp. 392-404.

[12] Chu, P.C. and Beasley, J.E., "A Genetic Algorithm for the Multidimensional Knapsack Problem", **Journal of Heuristics**, Vol. 4, 1998, pp. 63-86.

[13] Rabadi G., "Minimizing the Total Earliness and Common Due Date and Sequence-Dependent Setup Times", **Ph.D. Dissertation. Orlando, FL, University of Central Florida**, 1999.

[14] Gomes F. C., Pardalos P., Oliveira C. S., Resende M. G.C.,(2001), "Reactive GRASP 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**, 2001, pp. 60-67.

[15] Delmaire H., Diaz J. A., Fernandez E., Reactive Grasp and Tabu Search Based Heuristics for the Single Resource Capacitated Plant Location Problem, **INFOR**., Vol. 37, 1999, pp. 194-225.

[16] Grefenstette, J. J., (1986), Optimization of Control Parameters for Genetic Algorithms, **IEEE Transactions on Systems, Man and Cybernetics**, Vol. SMC-16, No. 1, Jan-Feb, 1986, pp. 122-128.

[17] Golden B., Pepper J., Vossen T., Using genetic algorithms for setting parameter values in heuristic search, **Intelligent Engineering Systems Through Artificial Neural Networks**, Vol. 8, 1998, pp. 239-245.

[18] Goldberg D. E., **Genetic Algorithms in Search Optimization & Machine Learning**, 2002, Addison Wesley.

[19] Deb K., **Multi-Objective Optimization Using Evolutionary Algorithms**, John Wiley & Sons, 2001, West Sussex, England.