2017

A generic memory design for a memoryless metaheuristic with the application of flowshop scheduling problem

Shayan Mohammadi

ABSTRACT

A GENERIC MEMORY DESIGN FOR A MEMORYLESS META-HEURISTIC WITH THE APPLICATION OF FLOWSHOP SCHEDULING PROBLEM

Shayan Mohammadi, M.S.
Department of Industrial and Systems Engineering
Northern Illinois University, 2017
Reinaldo Moraga and Gary Chen, Co-Directors

Although a strong construction phase in meta-heuristic algorithms is a critical factor to yield high-quality solutions in the local search, it has not been investigated thoroughly. The most effective mechanism to ensure the search in new areas is randomness, and a memory mechanism can help the algorithm tracking potential of good solutions during the search. This research focuses on depicting a general memory design in the construction phase of a memoryless meta-heuristic entitled Meta-RaPS (Meta-heuristic for Randomized Priority Search) in order to showing the effectiveness of spending more time in the construction phase. Permutation Flow Shop Scheduling Problem (PFSP) and famous Tillard's benchmark is represented as the application of memory mechanism in the construction phase of Meta-RaPS. The results highlight that implementing memory and learning mechanisms in the construction phase of Meta-RaPS improves its effectiveness. Computational results display the algorithm's competency even though the algorithm is just a construction meta-heuristic. The suggested technique strengthens the hypothesis that if the right procedure is executed in the construction phase of combinatorial optimization problems, local search can be eventually eliminated.

NORTHERN ILLINOIS UNIVERSITY
DEKALB, ILLINOIS

AUGUST 2017

A GENERIC MEMORY DESIGN FOR A MEMORYLESS META-HEURISTIC WITH THE APPLICATION OF FLOWSHOP SCHEDULING PROBLEM

By
SHAYAN MOHAMMADI

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
MASTER OF SCIENCE

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

Thesis Co-Directors:
Reinaldo Moraga
Gary Chen

ACKNOWLEDGEMENTS

I like to thank my advisor, Dr. Reinaldo Moraga, for his undeniable supports and counsels throughout the process of performing this research. At some points the progress was slow and the targets sounded unachievable; however, with Dr. Moraga's supports and encouragements I remained decisive to complete this research. Moreover, I would like to thank Dr. Gary Chen and Dr. Ziteng Wang for their support, insights, and suggestions to advance the quality of this research. Intuitions from this research will always remain with me. This research has shaped my mind about the realm of combinatorial optimization and artificial intelligence. I have gained incredible knowledge to utilizing applied mathematics in real world optimization problems. I also like to thank my family and friends who have contributed their expertise, time and resources during this research, I am thankful to all of them.

DEDICATION

I dedicate this research thesis to strongest supporters of my life, my parents–Abbas & Roya–who continuously supported me through all the steps in my life.

Table of Contents

List of Tables

List of Figures

1-INTRODUCTION

Pinedo (2008) states that scheduling is an integral part of the process of decision making where the intention of scheduling is to optimize the objective function(s) which is(are) subject to a group of constraints. The main constraints of scheduling problems are limited number of resources over the horizon of scheduling and due date of the jobs. Characteristics of resources may vary from industry to industry and attributes of each problem is unique. In the world of manufacturing the main resources are machines and operators. A detailed scheduling plan is necessary to satisfy the required efficiency and effectiveness of a system. For example, in real world, jobs may have to wait due to machines' unavailability or when a high priority job arises, preemptions may occur as a result of this priority. Constraints and machine environment of a scheduling problem are the main factors to define a problem and prepare a decent schedule.

The common notation in literature to address a problem is developed by Graham, Lawler, Lenstra and Kan (1979) which is a three parameters notation $\alpha/\beta/\gamma$. Pinedo (2008) describes the field of α as machine environment which consists of a single entry. Field of β discusses the characteristics of process and constraints, β can have no entries, single entry, or multiple entries. The last field γ, contains the objective function and usually has a single entry.

Machine environment comprises five major categories:

1. Single machine
2. Parallel machine
3. Job shop
4. Flowshop
5. Open shop

Sun, Zhang, Gao and Wang (2011) address the following notations for each machine environment, single machine is noted as 1, *Pm* is noted for identical parallel machine, *Qm* is noted for uniform parallel machine, *Jm* is noted for job shop, *Fm* is noted for flowshop, and *Om* is noted for open shop machining. Some of the examples for field β are permutation (*prmu*), sequence-dependent setup times (*SDST*), precedence constraints (*prec*), lot streaming (ls*m*), no-wait (*nwt*), preemption (*prmp*), limited buffers (*block*), breakdowns (*brkdwn*), machine eligibility (*Mj*), stochastic (*stch*), and reentrant (*retr*). Sun et al (2011) provides some examples for field γ such as total completion time (*C*), total flow time (*F*), total tardiness (*T*), maximum completion (C_{max}) and maximum lateness (L_{max}).

1-1 Background of the Problem

Pinedo (2008) mentions that in most facility and service industries each job has to follow a series of operations. Regularly, all operations have to be performed on every job in the same order which implies that, the jobs are following a same sequence or path. This condition is defined as flowshop machine environment. Flowshop machine environment assumes machines are in series. Flowshop machining can have some special features regarding to properties of a problem.

Flowshop machining varies from industry to industry, a more general type of flowshop machining is flexible flowshop. According to Pinedo (2008) a flexible flowshop or compound flowshop or hybrid flowshop has three characteristics. It has at least more than one machine in one of the stages, and a job has to be machined at each stage, but only on one of the machines on the stage(s) with more than one machine. Allaoui and Artiba (2006) mention that in recent years hybrid flowshop is a more common practice than traditional flowshop, specifically in electronic manufacturing industries.

According to Ruiz and Maroto (2006) the most common objective in flowshop machine environment is finding a sequence to process the jobs on machines in order to minimize the completion time of the last job on last the machine; this objective is called makespan or C_{max}. The processing times of jobs on machines are known in advance, fixed, and non-negative. In other words, the objective function is defined in a deterministic condition of process times.

Flowshops that do not allow sequence change between machines are called permutation flowshops. In permutation flowshops, all jobs have to be machines in a same sequence or permutation. According to Rahman, Sarker and Essam (2015) permutation flowshop is a way to process n jobs on m machines while each job has to be processed on each machine without exception in a same sequence.

1-2 Problem Statement

The proposed problem in this research would be a flowshop scheduling problem with *n* jobs *j*=1 to n, and *m* machines $i = 1$ to m, and the objective function is C_{max} , the classification of problem base on the notation is:

$$F_m | prmu | C_{max}$$

Where:

$Cmax_{ij} = \max\{Cmax_{i,j-1}, Cmax_{i-1,j}\} + t_{ij}$

$Cmax = Cmax_{mn}$

$Cmax_{0,j} = Cmax_{i,0} = 0$

According to Johnson and Montgomery (1974), permutation flowshop scheduling with more than two machines is NP-hard. Therefore, most researches develop heuristics and approximate methods to find a high quality solution in a reasonable time.

Assumptions in permutation flowshop scheduling are:

- Process times t_{ij} where i is referring to machines and it is $i = 1, \dots, m$ and j is referring to jobs and it is $j = 1, \dots, n$, are known and deterministic.

- Each job j can be processed at most on one machine i at a time
- Each machine i can process only one job j at a time
- No preemption is allowed
- All jobs are independent and are available for processing at time 0
- The set-up times of the jobs on machines are negligible
- The machines are continuously available
- In-process inventory is allowed

The number of all possible permutations for PFSP is $n!$. Where n is the number of jobs on a given problem. The effort is to assign jobs on machines in a way that minimizes the completion time of the last job on the last machine.

Rabadi (2016) describes Meta-RaPS as a generic and high level algorithm which focuses on constructing and then improving a feasible solution by introducing randomness in a simple heuristic rule. Meta-RaPS was formally introduced by Moraga (2002). The main purpose of this research is to embed an adaptive memory based procedure in the construction phase of the Meta-RaPS with the application of PFSP. If high-quality solutions are generated in the construction phase it would be a great step toward diminishing the improvement phase.

1-3 Objective and Scope

The objective of this research is to enhance the performance of the construction phase of Meta-RaPS with memory and learning mechanisms. In order to get a high quality solution which eventually leads to eliminating the improvement phase when it is compared to the other heuristics and meta-heuristics.

Scope of the research is designing a generic memory technique for the construction phase of Meta-RaPS. PFSP is introduced as the application of this design. The priority sequencing rule is Dong, Huang and Chen (2008) priority rule and the proposed method is tested on Taillard's (1993) benchmark to investigate:

- Effectiveness of Meta-RaPS with adaptive memory against memoryless Meta-RaPS
- Effectiveness of Meta-RaPS with adaptive Memory against other heuristics and meta-heuristics

1-4 Deliverables

A generic memory design for the construction phase of Meta-RaPS has never been studied in the literature. Hence this research will assist the society of operations research by providing a completer version of Meta-RaPS.

2-LITERATURE REVIEW

Permutation flowshop scheduling problem is known to be NP-hard for more than two machines. The domain of feasibility has $n!$ unique solutions; therefore, only exhaustive search methods can lead to the guaranteed optimal permutation. Pinedo (2008) states that even for small size problems the exhaustive search method is expensive. Literature review section focuses on the main approaches to tackle the permutation flowshop problem, Meta-RaPS, and memory in meta-heuristics. Available techniques to deal with PSFP can be classified as exact algorithms, heuristics, and meta-heuristics.

2-1 Exact Algorithms

Computing all $n!$ permutations is the easiest and the most straightforward exact algorithm, since it explores all feasible solutions and then reports the best one(s) out of all. However, this strategy is almost impractical even for small size problems. This is due to computational complexity that the method is dealing with.

Johnson (1954) proposes an algorithm for flowshop problems with two machines and n jobs which provides the actual optimal solution. Johnson's research demonstrates that the same permutation of jobs can be applied on both machines to obtain the optimal solution. Johnson's algorithm is proven to find the optimal solution when the problem is dealing with two machines or three

machines under specific conditions. However, Johnson's rule is clearly ineffective when the number of machines is more than three.

In addition to exhaustive search and Johnson's rule, branch and bound techniques can provide the optimal solution when known upper bound or lower bound of the problem is used to restrict the search space. Ignall and Schrage (1965) developed the first and initial branch and bound algorithm for PFSP when the objective function is makespan.

2-2 Heuristics

According to Osman and Laporte (1996) in permutation flowshop problems, the most pragmatic procedures to obtain an optimal or near optimal solutions are heuristics. This due to the fact that these problems are NP-hard which results in expensive computational times for exact methods. Generally, heuristic algorithms are divided into two classes of construction and improvement. In constructive heuristics the effort is to construct a sequence based on some criteria; while improvement heuristics improve a constructed solution iteratively with a local search method.

2-2-1 Construction Heuristics

Palmer (1965) suggests a slop index ranking method to sequencing the jobs based on ascending order of indices, which means giving the higher priority to the jobs with lower slope (index). CDS by Campbell, Dudek, and Smith (1970) is a heuristic method which converts a n jobs m machines flowshop problem into a n jobs and two virtual machines problem to utilize the Johnson's rule on

the virtual machines. Gupta (1971) modifies Palmer's ranking algorithm according to a function, to sort the jobs and construct the schedule. Dannenbring (1977) introduces Rapid Access (RA) heuristic algorithm which is the combination of Palmer's sloping index with some modifications and CDS algorithm. Therefore, it has an edge over Johnson's rule which is applicable to two machines flowshops.

Nawaz, Enscore, and Ham (1983) introduce a construction heuristic for flowshop problems which is known as NEH heuristic. NEH has two stages:

1. Sorting the jobs in descending order of total process times of each job
2. Jobs are removed from the initial sorted sequence one by one and are then placed in a partial sequence. When a job is going to be added to a partial schedule, it will be inserted to all possible positions. Among all the positions, the job will be fixed in the cheapest one. The procedure is continued till all jobs are scheduled in the partial sequence and the sorted sequence is vacant.

According to Ruiz et al (2006) the computational complexity of the NEH is $O(n^3m)$. However, Tillard (1990) proposes a method that reduces complexity to $O(n^2m)$. Based on Turner and Booth (1987), Ruiz, et al (2006) and Viagas, Ruiz, Framinan (2016) reviews on PFSP, the NEH heuristic is regarded as the best construction heuristic for flowshop problems so far. Hence, developing different versions of the NEH has been investigated a lot. According to Vigas et al (2016) different versions of NEH are noted as $NEH(a|b|c)$ in the literature. Where a, b and c are:

a) Initial order in the first stage of NEH, some examples of this field are:
   LPT: describing value of sum of process time (original NEH)
   SPT: ascending value of summation of process time

Dev: ordering based on standard deviation of each job which is proposed by Li, Wang and Wu (2004)

AvgStd: sum of mean and standard deviation of each job which is proposed by Dong et al (2008)

b) Tie breaking mechanism when a same makespan is gained by different sequences
c) Reversibility of the problem which is studied by Ribas, Companys and Martorelli (2010)

2-2-2 Improvement Heurtistics

The essence of improvement algorithms is to modify an already constructed solution in order to enhance it. This modification can be categorized into two main classes: neighborhood search, which is switching a small portion of a constructed solution; and recombination mechanism, which is combining decent properties of different solutions. Based on Viagas et al (2016) the most studied neighborhood search techniques are simulated annealing (SA) and Tabu Search; and most studied recombination mechanisms are genetic algorithm (GA), artificial immune system (AIS) and artificial neural networks (ANN).

2-3 Meta-Heuristics

Osman et al (1996) describes meta-heuristics as systematic frameworks that can be used in combinatorial optimization problems to improve the quality of solutions. Tillard (1990) applied the first time version of Tabu search on flowshop scheduling, the other designs of Tabu Search for flowshop problems introduced by Nowicki and Smutnicki (1996), and Ben-Daya and Al-Fawzan (1998). Application of Genetic Algorithms on flowshop problems started by Reeves (1995); Chen,

Vempati, and Aljaber (1995); and Murata, Ishibuchi, and Tanaka (1996). Simulated Annealing functionality on flowshop problems was tested by Osman and Potts (1989) and Ogbu and Smith (1990). Moreover, utilization of hybrid methods in the flowshop problems was started with Zegordi, Itoh and Enkawa (1995) and Moccellin and Dos Santos (2000).

2-4 Meta-RaPS

Meta-RaPS stands for Metaheuristic for Randomized Priority Search. Moraga, DePuy and Whitehouse (2005) describe Meta-RaPS as a general strategy that produces feasible solutions by constructing and improving them through utilization of simple heuristic rules in a randomized manner. Authors mention that Meta-RaPS is the product of a research conducted on the application of a modified COMSOAL (Computer Method of Sequencing Operations for Assembly Lines) for solving combinatorial optimization problems.

Arin and Rabadi (2013) state that Meta-RaPS first generates a feasible solution by including randomness in the construction phase and then improves the generated feasible solution in the improvement phase. According to Arin et al. (2013), Meta-RaPS constructs a solution by adding feasible elements, variables, tasks or activities in a randomized fashion to a partially constructed solution based on a priority rule until a termination criterion is met. Meta-RaPS avoids most local optimums during the construction phase due to having the advantage of randomness.

Arin et al (2013) declares that Meta-RaPS execution needs four parameters: number of iterations (I), the priority percentage ($p\%$), restriction percentage ($r\%$), and the improvement

percentage ($i\%$). Meta- RaPS does not select the component, variable, task or activity with the best priority sequencing value in each iteration. On the other hand, the algorithm is constantly altering between choosing the best component, variable, task or activity or accepting a component, variable, task or activity with a good priority value, but not necessarily the best one. The parameter $p\%$ defines the percentage of the time that a component, variable, task or activity with the best priority sequencing value will be selected and added to the partial solution, and $100 - p\%$ of the times a component, variable, task or activity will be selected randomly from a candidate list (CL) which contains "good" components, variables, tasks or activities. The CL is constructed by including elements whose priority sequencing values (d_j) are within $r\%$ of the range of best priority value. Elements are added to a partial solution until a complete constructed solution is generated.

If the best feasible element has the lowest priority sequencing rule, an element belongs to CL if:

$d_j \leq \alpha + (\beta - \alpha) * r$, where $\alpha = \min(d_j)\ and\ \beta = \max(d_j)$

Otherwise, the best feasible element has the largest priority sequencing rule and an element belongs to CL if:

$d_j \geq \beta + (\alpha - \beta) * r$, where $\alpha = \min(d_j)\ and\ \beta = \max(d_j)$

Smaller values of $p\%$ and larger values of $r\%$ add more randomness to the algorithm. In addition, for a given value of $p\%$, greater values of $r\%$ add more randomness.

When construction of a feasible solution is completed it may go through the improvement phase. In the improvement phase, usually a neighborhood search algorithm is employed. The parameter $i\%$ determines whether an already constructed solution will be improved or not. Meta-RaPS keeps track of both the best $(b*)$ and the worst $(w*)$ constructed objective functions. A constructed solution will be considered for improvement phase if its objective function (Z) satisfies the following inequality constraints:

$\{Z \leq b^* + (w^* - b^*)i\%\}$ Minimization problems
$\{Z \geq b^* - (b* - w*)i\%\}$ Maximization problems

The idea is that the better constructed solutions can perform better in the improvement phase of Meta-RaPS. Arin et al (2013) states that the quality of generated solutions by Meta-RaPS heavily dependents on its parameters. Specifically the number of iterations and the improvement percentage. However, it is worth mentioning that increasing the value of these parameters will definitely boost the computational time of the algorithm.

Meta-RaPS has proven its effectiveness for some of the well-known N-P hard problems such as capacitated vehicle routing problem (CVRP) by Moraga, Whitehouse, DePuy, Neyveli and

Kuttuva (2001); bin packing problem by DePuy, Whitehouse and Moraga (2002); 0-1 multidimensional knapsack problem by Moraga et al (2005); traveling salesman problem (TSP) by DePuy, Moraga and Whitehouse (2005); unrelated parallel machine problem (PMSP) by Rabadi, Moraga and Al-Salem (2006); set covering problem (SCP) by Lan, DePuy and Whitehouse (2007); Early/Tardy Single Machine Scheduling Problem by Hepdogan, Moraga, DePuy and Whitehouse (2009); and Parallel Multiple-Area Spatial Scheduling Problem with Release Times by Garcia and Rabadi (2011).

2-5 Memory and Meta-Heuristics

Glover and Laguna (1997) introduce a classification method for metaheuristic algorithms that have an advantage over a memory mechanism. The classes are utilization of adaptive memory, neighborhood search memory, and the number of solutions transmitted from one iteration to the next ones. According to memory classification of Glover at al. (1997) a three field notation $a|b|c$ can help classify meta-heuristic algorithms. If a metaheuristic has an advantage over adaptive memory, field a will be noted as A, otherwise it would be regarded as memoryless and it will be noted as M. Based on the neighborhood search mechanism the second field, b, will either be N if there exists a systematic neighborhood search or S if a random sampling method is utilized. The field, c, would be noted as 1 if in each iteration the algorithm is dealing with one solution. On the other hand, the field, c would be noted as P if the algorithm is based on a population of solutions with size of P. Table1 summarizes the memory classification's description.

Table 1. Memory in Metaheuristics Classification

a	Using Adaptive Memory	A	Adaptive Memory
		M	Memoryless
b	Type of Neighborhood Search	N	Systematic Search
		S	Random Search
c	Number of Solutions at Each Iteration	1	Single Solution
		P	Population Solution

Arin et al (2013) states that considering Tabu Search (TS) is the best way to define the foundation of memory and learning in meta-heuristics. TS is based on four major facets: recency, frequency, quality, and influence. Recency tracks the changes in attributes of solutions in the search process which are changed recently. Tabu is an attributes in the recent visited solutions and a tabu move is a move that leads to a tabu attribute.

The best way to summarize recency in one word is short term memory. Therefore, there should be a facet that deals with long term memory. Frequency takes the responsibility of long term memory in TS. Frequency includes two major aspects: transition frequency, which measures how frequent attributes are changing; and residence frequency, which measures how frequent attributes are considered in generated solutions. Glover et al (1997) illustrates this feature by an example in scheduling: transition frequency can be considered as the number of times that job *j* has been moved to an earlier position in the sequence, and residence frequency can be considered as the

summation of tardiness for a given job (j) when it is located in position $[k]$. Quality deals with the shared features of good solutions or the paths that guide to a good solution. Mechanisms such as applying penalty for poor moves that may lead to poor solutions are considered are instances of quality. The last mentioned facet is influence. Influence is the effects of the decisions which are made in the process of generating a solution. Quality facet can be regarded as a type of influence facet.

2-5-1 Adaptive Memory Programming

Taillard, Gamberdella, Gendreau and Potvin (2001) summarize principles of an effective adaptive memory programming (AMP) in meta-heuristics. This research compares and integrates common features between different approaches in metaheuristics that have an advantage over AMP. According to Taillard et al (2001), the foundation of an effective AMP is based on some or all of the following features:

1. Memorizing a set of solutions or special data structures that aggregates the particularities of the generated solutions throughout the search
2. Constructing a provisional solution with memorized data;
3. Using a logical search algorithm or a more sophisticated metaheuristic to improve the solutions
4. Using new solutions' information to update memorized data structure.

Taillard (1998) declares that the performances of AMP approaches are much better than heuristic algorithms such as simulated annealing and genetic algorithms, and this is due to utilization of

memory in AMP algorithms. It can be seen in Tillard's work (1998) that utilization of AMP can find a better solution in a shorter time compared to Tabu Search, Simulated Annealing, and Variable Neighborhood Search. Tillard provides stronger evidences to support his argument in a further research entitled "Adaptive Memory Programming: A unified view of metaheuristics", Taillard et al (2001). This research compares different heuristics including different AMP methods on NP-hard problems such as quadratic assignment problem (QAP).

2-5-2 Meta-RaPS and Memory

Based on Glover et al (1997) metaheuristics classification, Meta-RaPS is a memoryless meta-heuristic. Therefore, it can benefit from a systematic procedure that utilizes memory in its structure. Meta-RaPS with memory concept was investigated by Lan et al (2007) for the first time on a set covering problems (SCP). Two adaptive memory structures were utilized in Lan et al (2007) research. First, utilization of elements' fitness in a priority rule, which is the idea of quality in TS or how frequent an element from a solution collaborates in a set of elite solutions. Second, partial construction which it is an illustration of a work done by Glover and Laguna (2000) on scatter search and path relinking which explore new solutions by tracking high-quality solutions.

Zegarra-Ballón (2009) investigates incorporation of memory in Meta-RaPS on a problem of unrelated parallel machine scheduling by integration of element fitness research by Lan et al. (2007), and recency from TS. Zegarra-Ballón's (2009) algorithm tracks the changes that are made in the recent past based on recency and changes the priority rule with element fitness. Computational results of these two researches show that Meta-RaPS with memory outperforms

the original Meta-RaPS. Current researches show that not only Meta-RaPS is a strong meta-heuristic framework for solving combinatorial optimization problems, but also the integration of memory in its structure can improve its performance.

Objective of this research is to design a generic adaptive memory structure for the construction phase of Meta-RaPS with the application of permutation flowshop scheduling problems and compare the outcomes with original Meta-RaPS and other competitive meta-heuristics in the literature. The ideal purpose of this research is to assess the idea of generating high-quality solutions that eliminate the necessity of the improvement phase.

Although there exist Meta-RaPS designs with memory in the literature, but a generic memory design is not available. Hence, designing a memory framework that helps Meta-RaPS to be considered as an Adaptive memory based meta-heuristic (according to Glover et al (1997) classification in Table 1) is favorable.

2-6 Summary

Literature review of flowshop scheduling problems reveals that the quality of Meta-RaPS as a meta-heuristic to solve flowsshop problems has not been investigated yet, so it can be seen as a gap to perform a research on. Most available meta-heuristics spend the minimum time on construction while Meta-RaPS attempts to balance the workload between the construction phase and the improvement phase.

Meta-RaPS with memory was studied by Lan et al (2007) to solve set covering problems and by Zegarra-Ballón (2009) to solve unrelated parallel machine scheduling problems. However, the design in neither of these researches is generic. Therefore, a generic memory structure for Meta-RaPS is another gap in this area.

In summary, the purpose of this research is to design a memory framework for Meta-RaPS to improve the effectiveness of this algorithm. Moreover, since the performance of Meta-RaPS in flowshop scheduling is not investigated, PFSP is selected as the application of this design.

3-METHODOLOGY

It is stated in the literature review that Meta-RaPS is a promising meta-heuristic that not only uses construction and improvement heuristics to generate high quality solutions, but it also attempts to balance the workload among these two phases. Rabadi (2016) states that Meta-RaPS constructs feasible solutions through utilization of randomness in a systematic procedure to avoid getting stuck in most local optimums and then uses a local search technique to enhance the solutions with the hope of reaching to a global optimum. Meta-RaPS execution needs four parameters: the priority percentage ($\%p$), the restriction percentage ($\%r$), the improvement percentage ($\%i$), and the number of iterations (I).

Number of iterations (I) controls execution of the process. p and r determine how the construction heuristic selects the next job and adds it to the partial solution. In addition, the improvement percentage determines whether a constructed solution will be passed through the improvement phase or not.

3-1 Meta-RaPS Construction Phase

NEH is the best construction heuristic (Turner et al, 1987; Ruiz et al, 2006; and Viagas et al, 2016) for PFSP with the objective of makespan.

Original NEH is as follows:

1- Order the n jobs in decreasing value of summation of process times on machines
2- Take the first two jobs and calculate both possible permutations' makespan, and then select the permutation with the lower makespan
3- For job $j^{th}, j = 3, \ldots, n$, insert the job into all possible places in the partial sequence and among j possible positions put the job in the position that minimizes makespan

It is obvious that NEH construction heuristic has two phases as follows:

1- Generate an initial sequence
2- Construct a solution

Speaking colloquially, first phase of NEH is assigning a priority sequencing value to the jobs and sorting them by descending value of this priority sequencing rule, and second phase of NEH is constructing a partial schedule gradually until a complete solution is obtained.

3-1-1 Meta-RaPS in the Initial Phase of NEH

Dong et al (2008) improves the NEH by changing the priority rule in the phase one. The priority rule is based on the following hypothesis: the larger the deviation for processing times for a given job on machines, the higher its priority should be. Therefore, Dong et al (2008) use a priority rule of $AVG_j + STD_j$ in phase one and sort the jobs in a decreasing order of $AVG_j + STD_j$ while AVG_j is:

$$AVG_j = \frac{1}{m}\sum_{i=1}^{m} p_{ij}$$

And standard deviation of process times for a given job j is:

$$STD_j = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(p_{ij} - AVG_j)}$$

Dong et al (2008) perform a one-side paired *t*-test on their priority rule and original NEH on Tillard's benchmark (1993). They state when the significance level is 0.05 $AvgDev = AVG_j + STD_j$ performs better that NEH. Hence the utilized priority sequencing rule in this research is Dong et al (2008) priority rule and it is noted as $AvgDev$.

Li et al (2004) and Dong et al (2008) show that initial sequence of NEH has a great impact on the quality of the second of phase of NEH; therefore in this section utilization of Meta-RaPS in the first phase of NEH with Dong priority rule is introduced.

In sequence construction phase, Meta-RaPS uses $AvgDev$ as the priority sequencing rule. While adding a job to the sequence, the parameter p determines that p percent of the time the job with the best priority sequencing value ($max(AvgDev)$) should be added to the sequence, and $100 - p\%$ of the time a job from the CL. The jobs in CL are the ones that are not sequenced and their $AvgDev$ values are within r percent of the range of best $AvgDev$.

When a sequence is constructed the second phase of NEH is performed. In other words, a job is always positioned in the position that minimizes makespan. Ties are broken arbitrary. The pseudo code for this algorithms is as Figure 1.

1- Set parameter I, p, r, Sch^*=inf,
2- For Iter= 1 to I
3- Set Seq=Ø, Sch= Ø, $j^a = \{1, \dots, n\}, and\ K = 1$
4- Generate a random number rnd ∈ [0,1]
5- If rnd <p
6- j^* is the job that satisfies $d^*_j = \max(AvgDev_j)$
7- Else
Take j^* randomly from the set of all $j \in j^a$ that satisfy
$d_j \geq \max(AvgDev_j) + [\min(AvgDev_j) - \max(AvgDev_j)] * r$
8- End
9- Assign job j^* to Seq in position k
10- Set k=k+1
11- Delete j^* from j^a
12- If j^a= Ø go to line 14, else go to line 4
13- End
14- Perform NEH second phase insertion method on Seq to construct Sch
15- If $Cmax(Sch) < Cmax(Sch^*)$
16- Update Sch^*
17- End
18- End
19- Report Sch^* and $Cmax(Sch^*)$

Figure 1. Pseudo Code for Meta-RaPS In the First Phase of NEH

3-1-2 Meta-RaPS in the Second Phase of NEH

This section introduces another design of Meta-RaPS, which is implementing Meta-RaPS in the second phase of NEH or job insertion in the partial schedule and selecting the position that minimizes makespan of partial schedule.

Initial sequence is constructed with Dong's priority rule and in the second phase, Meta-RaPS uses C_{max} partial schedule as the priority rule. While adding a job to a partial schedule the parameter p determines that p percent of the time the job is going to be inserted in the position that minimizes the C_{max}. The remaining times $(1-p)$, the job is going to be located in a position randomly from those positions that their C_{max} values are within r percent of the range of best (minimum) C_{max}. The pseudo code is as Figure 2.

3-1-3 Meta-RaPS in Both Phases of NEH

This design is the integration of both Meta-RaPS in the first and Meta-RaPS in the second phase of NEH. First the algorithm constructs a sequence with Dong's priority rule in Meta-RaPS framework and when a feasible sequence is generated the algorithm uses this sequence as the input of Meta-RaPS in the second phase of NEH. Trial and error process confirms that the parameters p and r should be tuned separately for Meta-RaPS in the first phase and Meta-RaPS in the second phase. Meta-RaPS in the first phase is noted by MR1 and the Meta-RaPS in the second phase is noted by MR2. The process is illustrated in Figure 3.

```
1-  Sequence the jobs in descending order of AvgDev
2-  Set parameters p, r, I, Sche* = Ø, Cmax(Sche*)=inf
3-  For Iter=1 to I
4-      PartialSchedule= Ø
5-          For j=1 to n
6-          Take job j from sequence
7-              If j==1
8-              Locate j in PartialSchedule
9-              End
10-         For k=1 to j
11-         Insert job j in position k of PartialSchedule and calculate Cmax(k)
12-         End
13-        Generate a random number (rnd)
14-            If rnd<p
15-            Make position of job j permanent in min(Cmax(k))

16-            Else choose position of job j arbitrary from positions that satisfy
               Cmax(k) > min(Cmax(k))+[max(Cmax(k))-min(Cmax(k))]*r
17-            End
18-        End
19-        Sche= PartialSchedule
20-           If Cmax(Sche)<Cmax(Sche*)
21-           Update Cmax(Sche*)
22-        End
23- End
24- Report Cmax(Sche*) and Sche*
```

Figure 2. Pseudo Code for Meta-RaPS in the Second Phase of NEH

```
1-  Set parameter r1,p1, r2, p2, I and Sch*=inf
2-  For Iter=1 to I
3-          Apply MR1 line 3 till 13
4-          Apply MR2 line 5 till 19
5-          If Cmax(Sche)<Cmax(Sche*)
6-              Update Cmax(Sche*)
7-          End
8-  End
9-  Report Cmax(Sche*) and Sche*
```

Figure 3. Pseudo Code for Meta-RaPS in Both Phases of NEH

3-2 Adaptive Memory Based Meta-RaPS

This research investigates the effectiveness of embedding memory in the construction phase of a memoryless meta-heuristics called Meta-RaPS which is a randomized algorithm that tries to share the workload between the construction phase and the improvement phase with the hope of eliminating the improvement phase. The focus of this section is to design a generic memory mechanism in the construction phase of Meta-RaPS with the application of flowshop scheduling problems. It is mentioned in the literature review section that memory is successfully implemented in Meta-RaPS by Lan et al (2007) for Set Covering Problem and by Zegarra-Ballón (2009) for Unrelated Parallel Machine Scheduling Problem by element fitness which is a technique derived from Tabu Search and is generally embedded on improvement heuristics rather than construction heuristics.

3-2-1 Memory Design and its Integration in Meta-RaPS Construction Phase

Multi-start algorithms usually do not benefit from generated knowledge of previous solutions in their construction phase. However, a multi-start algorithm that benefits from this knowledge in its construction phase can be designed by applying a well-established precept into it.

Consider list S that consists of e elite solutions with respect to an objective function. S has e null solutions at the beginning of the search with the costs of infinity (for minimization problems). Let $C(N)$ be the cost of a new feasible solution N, and $C(Worst(S))$ be the maximum cost in list S, if $C(N) < C(Worst(S))$, N is a candidate to be added in S and it will replace $Worst(S)$.

In each iteration, before Meta-RaPS generates a new solution, a frequency matrix is implemented in a probabilistic procedure as a memory mechanism. Probability matrix tracks the occurrence of each variable in the elite list. Consider Table 2 as the set of elite solutions for a problem with 5 jobs ($n = 5$) and 3 machines ($m = 3$) where $e = 4$.

Table 2. List S of elite solutions

	List S				
	Position				
S	*1*	*2*	*3*	*4*	*5*
1	4	1	5	2	3
2	3	2	5	1	4
3	2	4	3	5	1
4	2	1	5	3	4

Elements of matrix represent the jobs and their positions in each elite solution. As an example in S1: job4 is in position1, job1 is in position2, job5 is in position3, job2 is in position4 and job3 is in position5. The frequency matrix which is a square matrix $(n * n)$ tracks the occurrence of each job in each position and it can be derived from S (Table 2), Table 3 shows the frequency matrix as follows:

Table 3. Frequency Matrix

		Position				
		1	*2*	*3*	*4*	*5*
Job	1	0	2	0	1	1
	2	2	1	0	1	0
	3	1	0	1	1	1
	4	1	1	0	0	2
	5	0	0	3	1	0

The probability $p_{jk} = \frac{O_{jk}}{\sum_j O_{jk}}$ where O_{jk} is the number of times job j is happening in position k.

The probability matrix of this example is as Table 4 where the summation of each row and each column of the table should be equal to one.

Table 4. Probability Matrix

		Position				
		1	*2*	*3*	*4*	*5*
Job	1	0	0.5	0	0.25	0.25
	2	0.5	0.25	0	0.25	0
	3	0.25	0	0.25	0.25	0.25
	4	0.25	0.25	0	0	0.5
	5	0	0	0.75	0.25	0

Now consider parameter m as mimicking percentage parameter. This parameter controls how often the probability matrix is going to be used. Elite solutions have n variables (in this case position); therefore for k values one through n a random number *(rnd)* would be generated, if $rnd(k) < m$ a job j would be copied in the partial solution with respect to the probability of jobs on the associated column with position $[k]$ in the probability matrix, then the job should be removed and the probability matrix should be updated after this removal. When utilization of memory is done the solution is incomplete, so Meta-RaPS construction phase (without used jobs) would be applied on this currently incomplete solution to generate a complete feasible solution.

Suppose $m = 0.4$ and $rnd(1) = 0.3$, since $rnd(1) < m$, so position [1] should be filled with the probability matrix. Chance of jobs to be selected for position [1] are: $\{p_{11} = 0\ ,\ p_{21} = 50,\ p_{31} = 25,\ p_{41} = 25$ and $p_{51} = 0\}$. Assume job 3 is selected for position [1], now job 3 will be located in the position [1] as Figure 4.

Position				
1	*2*	*3*	*4*	*5*
3(job)				

Figure 4. Partial Constructed Solution

When job 3 is removed from frequency matrix the probability matrix should be updated as Table 5. This is to avoid using this job in another position. The process continues for position $k = 2\ to\ n$ (to consider all positons for partial construction). The new partial constructed solution is currently incomplete, so Meta-RaPS construction phase would be applied on this currently incomplete solution to generate a complete feasible solution.

Table 5. Updated Probability Matrix after Removing Job 3

		Position				
		1	*2*	*3*	*4*	*5*
Job	1	0	0.50	0	0.33	0.33
	2	0.67	0.25	0	0.33	0
	3	0	0	0	0	0
	4	0.33	0.25	0	0	0.67
	5	0	0	1.00	0.33	0

To keep the diversity of S high in the first iterations the value of m should be adjusted adaptively, but up to a threshold(δ). The threshold ensures that the percentage of partial construction will not

pass a certain level. This is to keep the essence of randomness in Meta-RaPS. Value of m can be adjusted with a function of $f(iter)$. A simple theoretical function that has a linear behavior is used in this research as follows, behavior of this function is depicted in Figure 5

$$f(iter) = \log(1 + e^{iter})$$

$$m = \begin{cases} f(iter) & if\ 0 < f(iter) < \delta \\ \delta & if\ f(iter) > \delta \end{cases}$$

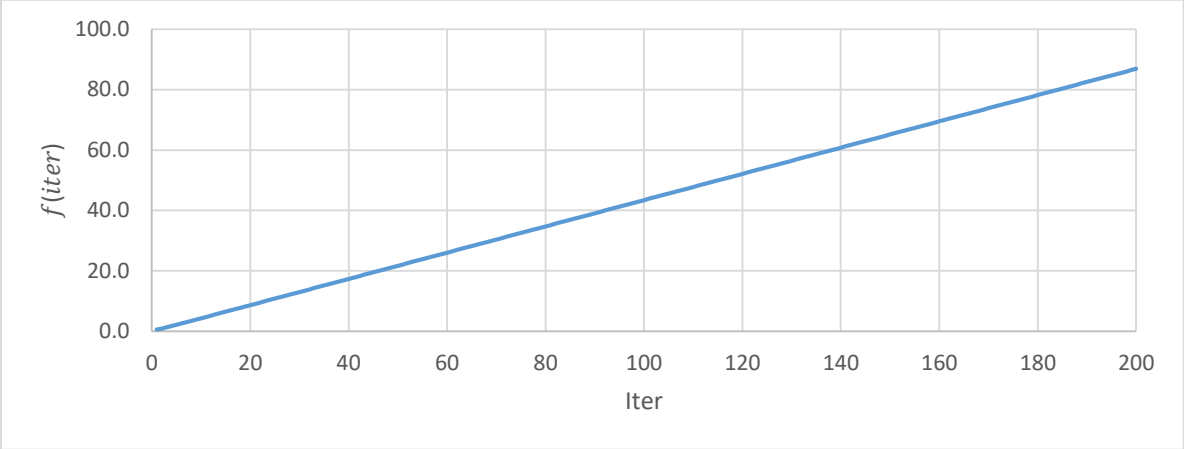

Figure 5. Behavior of Theoretical Function with Number of Iterations

3-2-2 Adding a Learning Mechanism

Elite list (S) tracks high quality solutions and it is used in a probabilistic fashion to construct a partial solution from good solutions. However, the memory mechanism does not bias the search and priority sequencing rule toward high quality solutions; therefore adding a mechanism that adjusts the priority rule toward high quality solutions is favorable. Essentials of priority rule adjustment are designed according to the principle of Tabu Search by Glover et al (1997). Meta-RaPS with memory and learning structure is denoted as MMR.

The assumption of adjustment is the following hypothesis, the more frequent a job is appearing in the first positions of the elite list, the higher its priority should be. Glover et al (2000) describes the principles of priority rule adjustment as follows: consider attribute sets as $A(x)$, each attribute is counted (weighted) in accordance with the number of times it appears in elements of $A(x)$. Speaking colloquially, the weight assigned to $A(x)$ is the weighted average of attributes of A to create a linear weighting combination. By considering the hypothesis of adjustment and the Glover principle of adjustment, frequency matrix can be used to count how often a job is appearing in a given position in the elite list. Moreover, a slope index adjusts the weight of each position (attribute). Let $k = 1$ to n be the notation that shows the position in schedule. By a linear weighting scheme we would have:

$$W_{(k)} = \frac{n + 1 - k}{n}$$

Now the impact of each position (attribute) k on the priority rule for each job is:

$$AvgDev_{j,k} = \frac{AvgDev_j}{1 - l * W_k * p_{jk}}$$

A learning parameter l is introduced to control the influence of adjustment on the priority rule (setting l=0 forces the model to utilize original priority rule). Dividing the priority rule by the weighting scheme that gives higher weight to the first positions makes sure that if a job is observed more in first positions its priority rule should be increased more.

Finally the new priority rule can be calculated by taking the average of all positions' priority rules:

$$AvgDev_j = \frac{1}{k}\sum_{k=1}^{n} AvgDev_{j,k}$$

3-2-3 Keeping Diversity of Elit List

A critical factor to avoid being trapped in local optimums is to keep the elite list divers. Designing a criterion to evaluate the goodness of replacing a high quality solution with one of the current solutions in S is essential.

Level of diversity can be evaluated by comparing the candidate solution (S') with available elite solutions in S . In other words, measuring number of observed differences in a candidate solution $[S']$ and elite solutions$[S_i]$. If similarity level of the candidate solution with elite solution i is less than a parameter d (diversity parameter), the candidate solution is diverse enough and it will be added to S if its objective function is better than $Worst(S)$. A candidate solution may have a better objective function than $Best(S)$. If this condition happens, the candidate solution will be added to elite list even if it is not diverse enough, because it satisfies an aspiration criterion which is having the best objective function. In summary a solution will be added to elite list if:

$$\begin{cases} [S'] - [S_i] < d.n \\ C(S') < C(Best(S)) \end{cases}$$

3-3 Adding a Tie Breaming Strategy to MR2

Preliminary results showed that implementation of Meta-RaPS in the second phase of NEH is facing considerable amount of ties in candidate list, and according to the essence of Meta-RaPS which is selecting randomly from the candidate list, ties are broken arbitrarily. It is obvious that the cheapest insertion does not lead to a high quality solution (cheapest insertion is the method of original NEH). In current state of the algorithm ties are broken arbitrarily but a more scientific way to break the ties may improve the effectiveness of the algorithm. According to Dong et al (2008) there is no global tie breaking algorithm that breaks the ties optimally in all problems but tie breaking strategies can improve the overall performance of an algorithm. Vigas and Framinan (2014) reviewed all available tie breaking rules for PFSP and all the proposed tie breaking algorithms are applied on the cheapest position. Vigas et al (2014) Computational results confirm that minimization of total idle time is best tie breaking mechanism, however it may not be effective in all problems. The tie breaking strategy in this research is the minimization of total idle time for the candidate list instead of cheapest position.

Idle time does not have a universal meaning in literature (Vigas et al, 2014). Available definitions for idle time are:

- Considering front delays and back delays
- Without consideration of both front delays and back delays.
- Considering front delays and excluding back delays.

Last definition of idle time which is considering front delays and excluding back delays is utilized in the tie breaking strategy of this research. Let It_i be the idle time of machine i and n be number of currently scheduled jobs, the idle time of machine i can be calculated by:

$$It_i = Cmax_{in} - \sum_{j=1}^{n} t_{ij}$$

And the total idle time is:

$$It = \sum_{i=1}^{m} It_i$$

This tie breaking strategy means if a position from candidate list is selected and there exist(s) another/other position(s) with the same makespan the position with lower idle time is prior to other position(s). If there is another tie, ties are broken arbitrary.

4-COMPUTATIONAL RESULTS

Three different procedures to implement Meta-RaPS in NEH are discussed, in the initial phase (sequencing stage), in the second phase (insertion stage), and a Meta-RaPS with memory and learning structure. This chapter starts with parameter tuning and goes through different designs to test the effectiveness of implementing Meta-RaPS in PFSP. Methods will be tested on Tillard's benchmark (1993), Tillard's benchmark has 12 problem sets. Starting with 20 jobs on 5 machines and going to 500 jobs on 20 machines and each problem set contains 10 instances.

4-1 Parameter Tuning in Meta-RaPS

Meta-RaPS is controlled by four parameter: priority percentage (p), restriction percentage (r), improvement percentage (i) and number of iterations (I). Since the focus of this research is just on the construction phase of Meta-RaPS the improvement percentage is considered zero.

According to Coy, Golden, Runger, and Wasil (2001) there are several strategies to effectively tune heuristics' and meta-heuristics' parameters. The complexity of process varies from trial and error to a more complex sensitivity analysis; therefore, it is a difficult task to find a universal effective value for parameters. In this paper, for each parameter a pre-determined set of values is considered.

Moraga (2002) defines a systematic approach to tune the parameter as following:

Step 1: Select a subset of problems to analyze from all the instances of problems.

Step 2: Select the domain that parameters will vary upon.

Step 3: For each problem in the subset, run Meta-RaPS with changing a parameter over its domain while other parameters are unchanged. Report the best outcome for the first parameter, continue tuning such that all parameters are reported. For example first manipulate parameter r while p and i are constant (set to zero). Report the $r\%$ associated with the best objective function. Continue tuning process with next parameter, p, while r is constant and i is zero.

Step 4: Use the reported parameters in Step 3 and apply them to problem sets.

Table 6 summarizes selected domain for parameters p and r for MR1 and MR2. Parameters with subscript 1 are associated to Meta-RaPS in the first phase of NEH (MR1) and parameters with subscript 2 are associated with Meta-RaPS in the second phase of NEH (MR2). A random problem set with 3 small size problems (20 and 50 jobs) and 3 large size problems (100 and 200 jobs) is selected from Tillard's benchmark. The outcome of the tuning process and values are demonstrated in Table 6. The number of iterations (I) is set to 200.

Table 6. Parameters Tuning Summary

Parameter	Set	Best value
p_1	$\{10\%, 20\%, 30\%\}$	10%
r_1	$\{\ 50\%, 60\%, 70\%, \}$	70%
p_2	$\{60\%, 70\%, 80\%, 90\%$	80%
r_2	$\{5\%, 10\%, 15\%, 20\%\}$	10%

4-2 Meta-RaPS in the First Phase of NEH Computational Results

Initial phase of NEH is sequencing stage. In the original NEH jobs are sequenced in decreasing values of summation of process times. Meta-RaPS adds randomness into a given priority sequencing rule to create more diverse solutions. Integration of NEH sequencing phase in Meta-RaPS construction phase is illustrated in section 3-1-1. Methods are tested on Tillard's benchmark with 5 independent runs and the best solution is reported from all runs. The algorithm is coded in Matlab 2011 with an Intel CORE-i5 CPU @ 2.5GHz and 4GB installed memory.

Performance of the algorithm on each instance is measured by the deviation of makespane (C_{max}) from the best known solution (BKS) in OR-library as of April 2017.

$$Dev = \frac{Cmax_{sol} - BKS}{BKS} * 100$$

NEH deviation of makespans and MR1 deviation of makespans from BKSs and the average computational time of MR1 for each Tillard's instance is shown in Table 7, Table 8, Table 9, and Table 10. If a BKS solution is found it is highlighted in bold.

Table 7. Computational Results on 20-job Taillard's Benchmark (MR1)

Problem	*n*m*	BKS	NEH	NEH Deviation (%)	Meta-RaPS	Meta-RaPS Deviation (%)	Average time (s)
TA001	20*5	1278	1286	0.63	**1278**	0.00	1
TA002		1359	1365	0.44	1365	0.44	
TA003		1081	1159	7.22	1104	2.13	
TA004		1293	1325	2.47	1306	1.01	
TA005		1235	1305	5.67	1250	1.21	
TA006		1195	1228	2.76	**1195**	0.00	
TA007		1239	1278	3.15	1241	0.16	
TA008		1206	1223	1.41	1207	0.08	
TA009		1230	1291	4.96	1238	0.65	
TA010		1108	1151	3.88	1126	1.62	
TA011	20*10	1582	1680	6.19	1604	1.39	2
TA012		1659	1729	4.22	1691	1.93	
TA013		1496	1557	4.08	1514	1.20	
TA014		1377	1439	4.50	1398	1.53	
TA015		1419	1502	5.85	1442	1.62	
TA016		1397	1453	4.01	1421	1.72	
TA017		1484	1562	5.26	1494	0.67	
TA018		1538	1609	4.62	1557	1.24	
TA019		1593	1647	3.39	1620	1.69	
TA020		1591	1653	3.90	1617	1.63	
TA021	20*20	2297	2410	4.92	2322	1.09	5
TA022		2099	2150	2.43	2106	0.33	
TA023		2326	2411	3.65	2347	0.90	
TA024		2223	2262	1.75	2227	0.18	
TA025		2291	2397	4.63	2323	1.40	
TA026		2226	2349	5.53	2263	1.66	
TA027		2273	2362	3.92	2312	1.72	
TA028		2200	2249	2.23	2220	0.91	
TA029		2237	2320	3.71	2260	1.03	
TA030		2178	2277	4.55	2210	1.47	

Table 8. Computational Results on 50-job Taillard's Benchmark (MR1)

Problem	*n*m*	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA031	50*5	2724	2733	0.33	**2724**	0.00	21
TA032		2834	2843	0.32	2843	0.32	
TA033		2621	2640	0.72	2622	0.04	
TA034		2751	2782	1.13	2761	0.36	
TA035		2863	2868	0.17	2864	0.03	
TA036		2829	2850	0.74	**2829**	0.00	
TA037		2725	2758	1.21	**2725**	0.00	
TA038		2683	2721	1.42	**2683**	0.00	
TA039		2552	2576	0.94	2554	0.08	
TA040		2782	2790	0.29	**2782**	0.00	
TA041	50*10	2991	3135	4.81	3086	3.18	28
TA042		2867	3032	5.76	2953	3.00	
TA043		2839	2986	5.18	2950	3.91	
TA044		3063	3198	4.41	3107	1.44	
TA045		2976	3160	6.18	3075	3.33	
TA046		3006	3178	5.72	3104	3.26	
TA047		3093	3277	5.95	3178	2.75	
TA048		3037	3123	2.83	3088	1.68	
TA049		2897	3002	3.62	2964	2.31	
TA050		3065	3257	6.26	3169	3.39	
TA051	50*20	3850	4082	6.03	4007	4.08	35
TA052		3704	3921	5.86	3864	4.32	
TA053		3640	3927	7.88	3796	4.29	
TA054		3720	3969	6.69	3857	3.68	
TA055		3610	3835	6.23	3879	7.45	
TA056		3681	3914	6.33	3816	3.67	
TA057		3704	3952	6.70	3874	4.59	
TA058		3691	3938	6.69	3898	5.61	
TA059		3743	3952	5.58	3878	3.61	
TA060		3756	4079	8.60	3874	3.14	

Table 9. Computational Results on 100-job Taillard's Benchmark (MR1)

Problem	*n*m*	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA061	100*5	5493	5519	0.47	5493	0.00	29
TA062		5268	5348	1.52	5280	0.23	
TA063		5175	5219	0.85	5179	0.08	
TA064		5014	5023	0.18	5021	0.14	
TA065		5250	5266	0.30	5250	0.00	
TA066		5135	5139	0.08	5135	0.00	
TA067		5246	5259	0.25	5249	0.06	
TA068		5094	5120	0.51	5097	0.06	
TA069		5448	5489	0.75	5449	0.02	
TA070		5322	5341	0.36	5328	0.11	
TA071	100*10	5770	5846	1.32	5807	0.64	43
TA072		5349	5453	1.94	5394	0.84	
TA073		5676	5824	2.61	5713	0.65	
TA074		5781	5929	2.56	5895	1.97	
TA075		5467	5679	3.88	5562	1.74	
TA076		5303	5375	1.36	5335	0.60	
TA077		5595	5704	1.95	5648	0.95	
TA078		5617	5760	2.55	5695	1.39	
TA079		5871	6032	2.74	5940	1.18	
TA080		5845	5918	1.25	5903	0.99	
TA081	100*20	6202	6541	5.47	6493	4.69	69
TA082		6183	6523	5.50	6414	3.74	
TA083		6271	6639	5.87	6518	3.94	
TA084		6269	6557	4.59	6480	3.37	
TA085		6314	6695	6.03	6541	3.60	
TA086		6364	6664	4.71	6622	4.05	
TA087		6268	6632	5.81	6510	3.86	
TA088		6401	6739	5.28	6685	4.44	
TA089		6275	6677	6.41	6537	4.18	
TA090		6434	6677	3.78	6640	3.20	

Table 10. Computational Results on 200-job Taillard's Benchmark (MR1)

Problem	*n*m*	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA091	200*10	10862	10942	0.74	10892	0.28	248
TA092		10480	10716	2.25	10614	1.28	
TA093		10922	11025	0.94	11017	0.87	
TA094		10889	11057	1.54	10921	0.29	
TA095		10524	10645	1.15	10575	0.48	
TA096		10329	10458	1.25	10338	0.09	
TA097		10854	10989	1.24	10934	0.74	
TA098		10730	10829	0.92	10798	0.63	
TA099		10438	10574	1.30	10501	0.60	
TA100		10675	10807	1.24	10758	0.78	
TA101	200*20	11195	11594	3.56	11195	3.39	481
TA102		11203	11675	4.21	11203	3.98	
TA103		11281	11852	5.06	11281	4.32	
TA104		11275	11803	4.68	11275	3.62	
TA105		11259	11685	3.78	11259	3.03	
TA106		11176	11629	4.05	11176	3.82	
TA107		11360	11833	4.16	11360	3.57	
TA108		11334	11913	5.11	11334	3.43	
TA109		11192	11673	4.30	11192	3.70	
TA110		11288	11869	5.15	11288	4.05	

When the number of machines are relatively low MR1 can find almost all the BKS. This is due to the fact that the complexity of problem is not significant. Therefore, it can be stated that a small amount of variation in a greedy rule can lead to extremely high quality solutions when the problem is dealing with a few machines. However, when the number of machines increases to 10 and 20 machines the algorithm is still better than NEH but not quite strong. This due to the fact that when number of machine increases the importance of second phase of NEH to yield good solutions increases. Therefore, randomness can improve the performance of NEH but not as well as low machine conditions

MR1 challenges the main idea behind NEH heuristic which is, the jobs with long process times must be scheduled as soon as possible. Dong et al. (2008) and Li et at. (2004) show that there exist other critical factor to consider in sequencing stage of NEH, like standard deviation of process times for a given job.

The main finding of MR1 is that first phase of NEH has an amazing impact on the performance of second phase of NEH. Several greedy rules are proposed before this research but MR1 shows that a greedy rule is ineffective since randomness drastically improves the performance of a greedy rule. MR1 has taken a huge step in answering the question of what really makes a sequence competent and effective and what features should be considered in sequencing stage of NEH to guide the second phase of NEH to yield high quality solutions.

4-2 Meta-RaPS in the Second Phase of NEH Computational Results

Second phase of NEH heuristic is the cheapest insertion method. In the original NEH, a job will always be located in a position that minimizes makespan. Meta-RaPS adds randomness into the insertion phase to avoid all jobs to be inserted in the cheapest position. The integration of NEH insertion phase into Meta-RaPS construction phase is illustrated in section 3-1-2. The method is tested on Tillard's benchmark with 5 independent runs and the best solution is reported from all runs.

Performance of the algorithm for each instance is measured by the deviation of C_{max} from the best known solution (BKS) in OR-library as of April 2017.

$$Dev = \frac{Cmax_{sol} - BKS}{BKS} * 100$$

NEH deviation of makespans and MR1 deviation of makespans from BKSs and the average computational time of MR2 for each Tillard's instance is shown in Table 11, Table 12, Table 13, and Table 14. If a BKS solution is found it is highlighted in bold.

Table 11. Computational Results on 20-job Taillard's Benchmark (MR2)

Problem	*n*m*	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA001	20*5	1278	1286	0.63	1297	1.49	1
TA002		1359	1365	0.44	1383	1.77	
TA003		1081	1159	7.22	1116	3.24	
TA004		1293	1325	2.47	1302	0.70	
TA005		1235	1305	5.67	1283	3.89	
TA006		1195	1228	2.76	1230	2.93	
TA007		1239	1278	3.15	1246	0.56	
TA008		1206	1223	1.41	1216	0.83	
TA009		1230	1291	4.96	1253	1.87	
TA010		1108	1151	3.88	1122	1.26	
TA011	20*10	1582	1680	6.19	1634	3.29	1
TA012		1659	1729	4.22	1682	1.39	
TA013		1496	1557	4.08	1517	1.40	
TA014		1377	1439	4.50	1397	1.45	
TA015		1419	1502	5.85	1444	1.76	
TA016		1397	1453	4.01	1427	2.15	
TA017		1484	1562	5.26	1503	1.28	
TA018		1538	1609	4.62	1577	2.54	
TA019		1593	1647	3.39	1623	1.88	
TA020		1591	1653	3.90	1627	2.26	
TA021	20*20	2297	2410	4.92	2328	1.35	3
TA022		2099	2150	2.43	2155	2.67	
TA023		2326	2411	3.65	2341	0.64	
TA024		2223	2262	1.75	2235	0.54	
TA025		2291	2397	4.63	2331	1.75	
TA026		2226	2349	5.53	2272	2.07	
TA027		2273	2362	3.92	2311	1.67	
TA028		2200	2249	2.23	2227	1.23	
TA029		2237	2320	3.71	2260	1.03	
TA030		2178	2277	4.55	2201	1.06	

Table 12. Computational Results on 50-job Taillard's Benchmark (MR2)

Problem	n*m	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA031	50*5	2724	2733	0.33	**2724**	0.00	16
TA032		2834	2843	0.32	2848	0.49	
TA033		2621	2640	0.72	2625	0.15	
TA034		2751	2782	1.13	2762	0.40	
TA035		2863	2868	0.17	2864	0.03	
TA036		2829	2850	0.74	2831	0.07	
TA037		2725	2758	1.21	2732	0.26	
TA038		2683	2721	1.42	2684	0.04	
TA039		2552	2576	0.94	**2552**	0.00	
TA040		2782	2790	0.29	2786	0.14	
TA041	50*10	2991	3135	4.81	3101	3.68	20
TA042		2867	3032	5.76	2978	3.87	
TA043		2839	2986	5.18	2936	3.42	
TA044		3063	3198	4.41	3103	1.31	
TA045		2976	3160	6.18	3093	3.93	
TA046		3006	3178	5.72	3110	3.46	
TA047		3093	3277	5.95	3179	2.78	
TA048		3037	3123	2.83	3102	2.14	
TA049		2897	3002	3.62	2971	2.55	
TA050		3065	3257	6.26	3162	3.16	
TA051	50*20	3850	4082	6.03	3994	3.74	32
TA052		3704	3921	5.86	3860	4.21	
TA053		3640	3927	7.88	3812	4.73	
TA054		3720	3969	6.69	3884	4.41	
TA055		3610	3835	6.23	3874	7.31	
TA056		3681	3914	6.33	3833	4.13	
TA057		3704	3952	6.70	3893	5.10	
TA058		3691	3938	6.69	3843	4.12	
TA059		3743	3952	5.58	3892	3.98	
TA060		3756	4079	8.60	3931	4.66	

Table 13. Computational Results on 100-job Taillard's Benchmark (MR2)

Problem	n*m	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA061	100*5	5493	5519	0.47	**5493**	0.00	32
TA062		5268	5348	1.52	5275	0.13	
TA063		5175	5219	0.85	5192	0.33	
TA064		5014	5023	0.18	5023	0.18	
TA065		5250	5266	0.30	5255	0.10	
TA066		5135	5139	0.08	**5135**	0.00	
TA067		5246	5259	0.25	5257	0.21	
TA068		5094	5120	0.51	5100	0.12	
TA069		5448	5489	0.75	5454	0.11	
TA070		5322	5341	0.36	5328	0.11	
TA071	100*10	5770	5846	1.32	5820	0.87	44
TA072		5349	5453	1.94	5405	1.05	
TA073		5676	5824	2.61	5708	0.56	
TA074		5781	5929	2.56	5949	2.91	
TA075		5467	5679	3.88	5601	2.45	
TA076		5303	5375	1.36	5358	1.04	
TA077		5595	5704	1.95	5659	1.14	
TA078		5617	5760	2.55	5707	1.60	
TA079		5871	6032	2.74	5983	1.91	
TA080		5845	5918	1.25	5909	1.09	
TA081	100*20	6202	6541	5.47	6533	5.34	71
TA082		6183	6523	5.50	6537	5.73	
TA083		6271	6639	5.87	6582	4.96	
TA084		6269	6557	4.59	6566	4.74	
TA085		6314	6695	6.03	6620	4.85	
TA086		6364	6664	4.71	6673	4.86	
TA087		6268	6632	5.81	6606	5.39	
TA088		6401	6739	5.28	6710	4.83	
TA029		2237	2320	3.71	6570	4.70	
TA030		2178	2277	4.55	6615	2.81	

Table 14. Computational Results on 200-job Taillard's Benchmark (MR2)

Problem	n*m	BKS	NEH	NEH Deviation	Meta-RaPS	Meta-RaPS Deviation	Average time (s)
TA091	200*10	10862	10942	0.74	10897	0.32	254
TA092		10480	10716	2.25	10673	1.84	
TA093		10922	11025	0.94	11017	0.87	
TA094		10889	11057	1.54	10929	0.37	
TA095		10524	10645	1.15	10586	0.59	
TA096		10329	10458	1.25	10396	0.65	
TA097		10854	10989	1.24	10947	0.86	
TA098		10730	10829	0.92	10789	0.55	
TA099		10438	10574	1.30	10512	0.71	
TA100		10675	10807	1.24	10767	0.86	
TA101	200*20	11195	11594	3.56	11572	3.37	310
TA102		11203	11675	4.21	11665	4.12	
TA103		11281	11852	5.06	11779	4.41	
TA104		11275	11803	4.68	11663	3.44	
TA105		11259	11685	3.78	11630	3.30	
TA106		11176	11629	4.05	11609	3.87	
TA107		11360	11833	4.16	11782	3.71	
TA108		11334	11913	5.11	11783	3.96	
TA109		11192	11673	4.30	11630	3.91	
TA110		11288	11869	5.15	11763	4.21	

The main drawback of NEH is that when the algorithm progresses at a given time only one job is considered for insertion and when a job is inserted to a position its position will remain constant in all next insertions. At each insertion step NEH behaves as a greedy rule and fixes the job in the cheapest position. However, it is clearly obvious that the best position for a job at its insertion step will not lead to a good solution when the next job is going to be inserted.

MR2 challenges the greedy behavior of NEH insertion phase. While it is a good idea to consider more than a job for insertion at each step, MR2 takes a different approach. While it is clearly ineffective to fix the jobs in the cheapest position and this related to mathematical properties of PFSP, it seems promising to considering other positions for insertion. Mathematics of PFSP indicates that the makespan of a flowshop problem equals to the longest path from the matrix of scheduled jobs from job 1 on machine 1 to job *n* on machine *m*. Therefore, changing the position of jobs changes the columns of the matrix and consequently the length of the path from first node to the last node. Fixing a job in a position that creates the shortest path at each step will be ineffective when a new column is added to the matrix. This is due to the fact that when a new job arises the length of the paths after its insertion will not the same as before its insertion. Therefore, MR2 attempts to fix a job in a position that does not seem favorable is its insertion step but will perform better when the next job or jobs will be inserted.

4-3 MMR Computational Results

MMR has three more parameters to tune. Therefore, sets $e = \{5,6,7\}$, $\delta = \{40\%, 50\%, 60\%, 70\%\}$, and $l = \{ 0.05, 0.1, 0.15, 0.2, 0.25, 0.3 , 0.35, 0.4, 0.45, 0.5\}$ are considered for the domain of these parameters. To evaluate the effectiveness of memory the parameters p and r are similar as MR1 (Meta-RaPS in the first phase of NEH). Figure 6, Figure 7 and Figure 8 reflect the Relative Percentage Deviation (RPD) for parameters δ with different capacity (e) of elite list. A random subset of problems with 3 small size problems (20 and 50 jobs) and 3 large size problems (100 and 200 jobs) is selected from Tillard's benchmark. The best setting obtained from parameters δ and e is used in further analysis to tune the parameter l.

Parameters analysis with 95% confidence intervals shows that increasing the size of elite list increases the performance of algorithm and this is due to having more options in the probability matrix, because having more options helps the algorithm avoid some of the local optimums. The behavior of m is interesting. By increasing the threshold, the performance of the algorithm increases but up to a point. Then the algorithm loses its potency and this is due to trapping new solutions in the areas close to the solutions in the elite list. It is shown in Figure 9 that low levels of l are performing better to bias the search. This is due to the fact that low levels of l only impact the positions with higher probabilities while high levels of l impact the behavior of the algorithm entirely. Figure 9 demonstrates that there is no significant difference between high levels of l but with fixing this parameter relatively low (in this research 0.2) better results are obtained. MMR's summary of parameters is shown in Table 15.

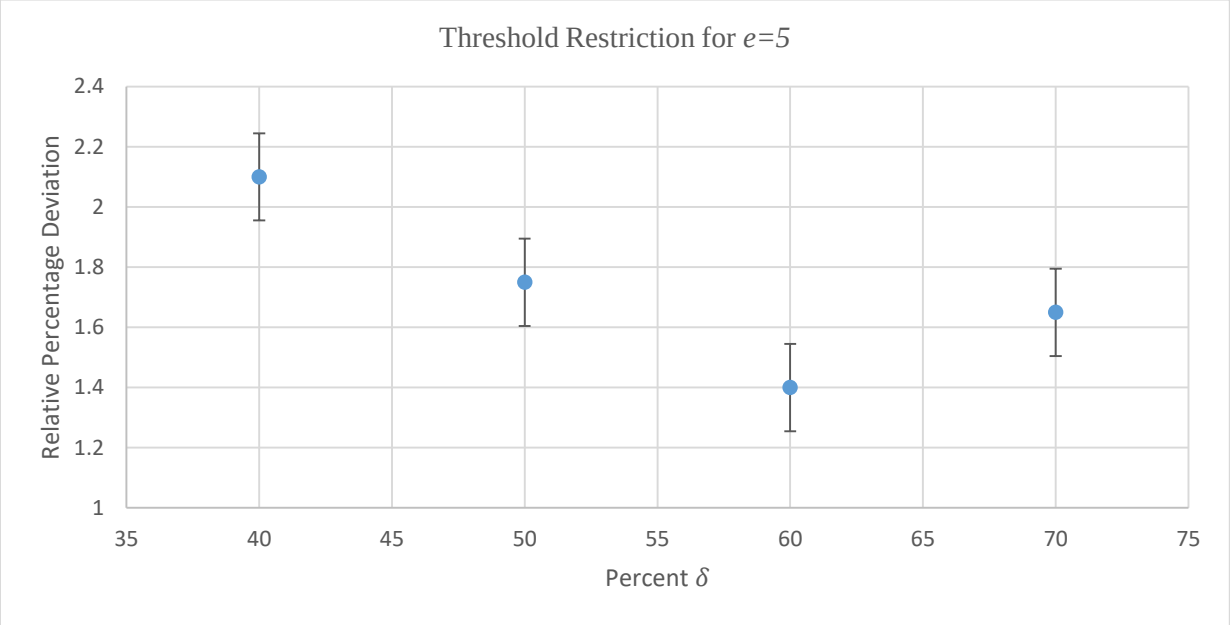

Figure 6. Average RPD to Store 5 Jobs in Elite List and Maximum Threshold δ

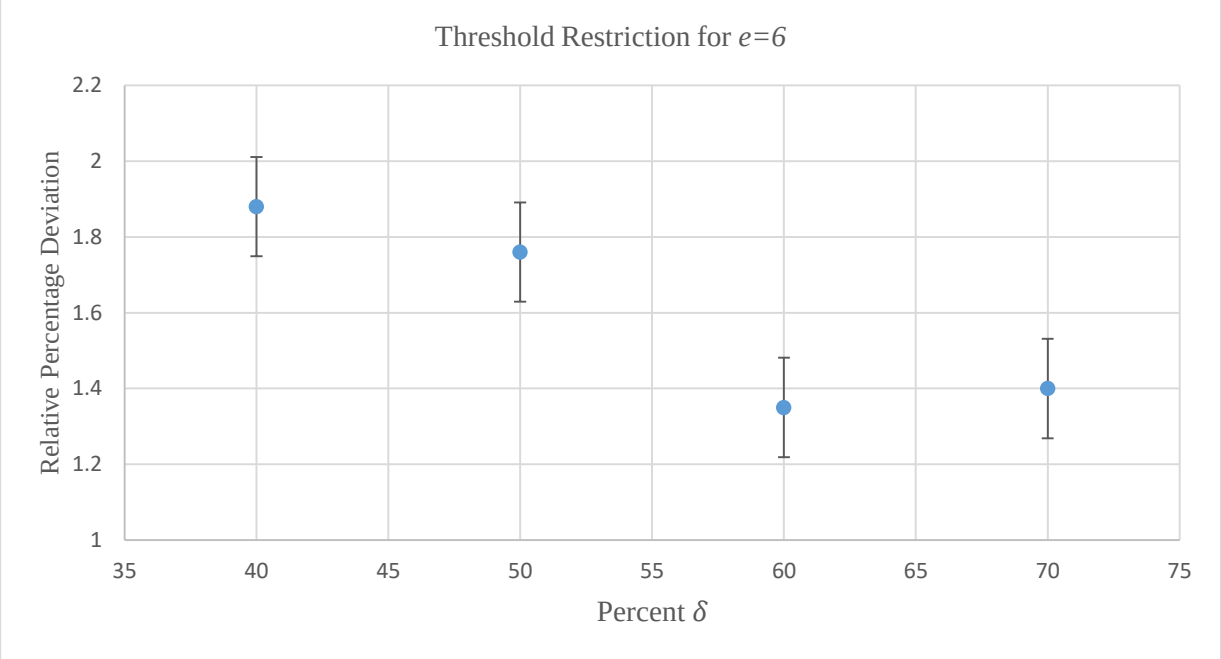

Figure 7. Average RPD to Store 6 Jobs in Elite List and Maximum Threshold δ

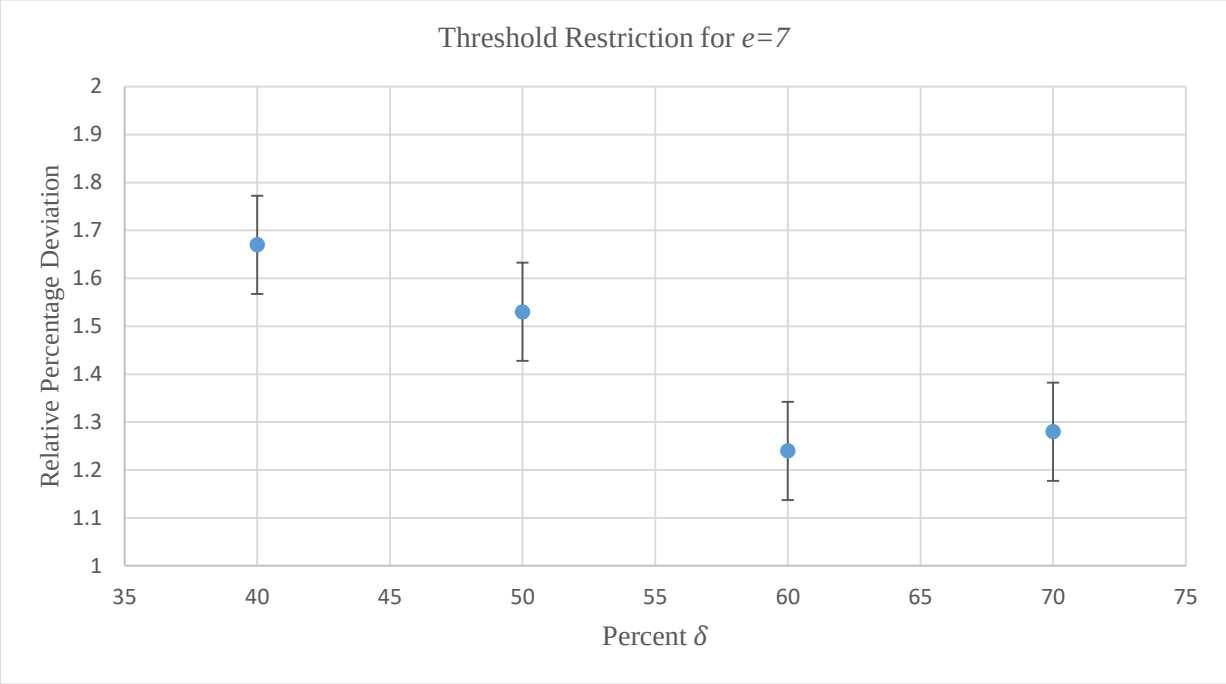

Figure 8. Average RPD to Store 7 Jobs in Elite List and Maximum Threshold δ

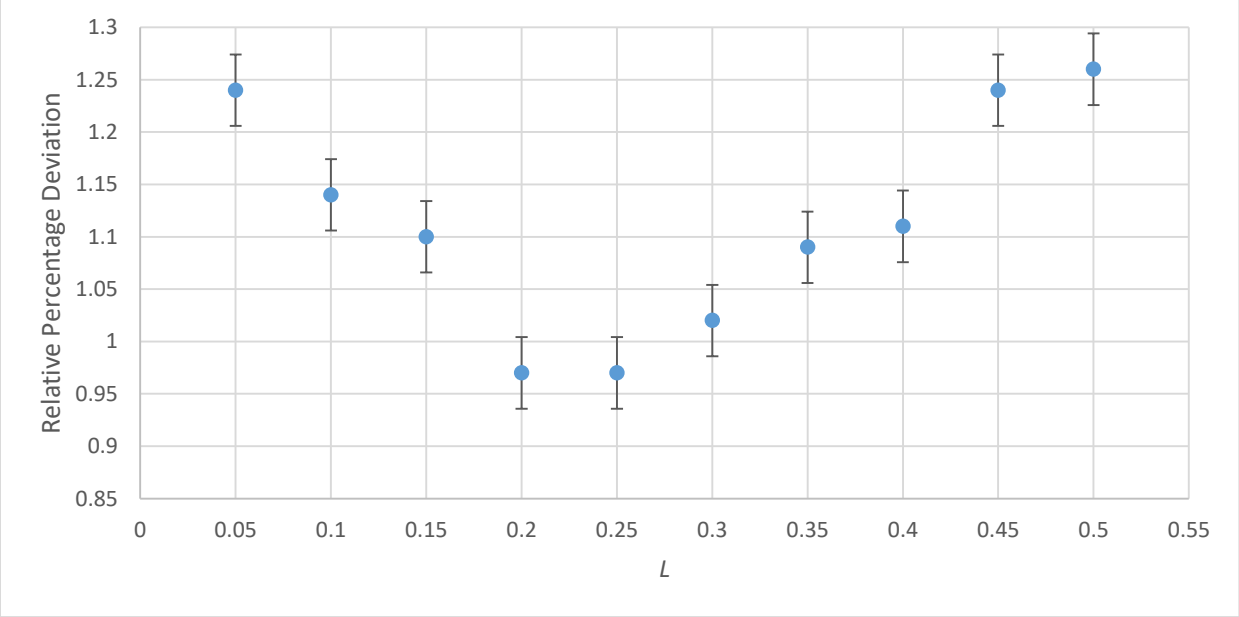

Figure 9. Average RPD for Parameter l

Table 15. Summary of Parameter for MMR

Parameter	**Value**
Priority percentage *(p)*	10%
Restriction Percentage *(r)*	70%
Number of Interations *(I)*	200
Elite list capacity *(e)*	7
Threshold (δ)	60%
Learning parameter *(l)*	0.2

Effectiveness of Memory structure in the construction phase of Meta-RaPS is tested in the first phase of NEH since the computational results confirmed a better performance in MR1. MMR with the application of PFSP is illustrated in section 3-2. Methods are tested on Tillard's benchmark with 5 independent runs and the best solution is reported from all runs. Performance of the algorithm is compared to BKS in OR-library as of April 2017.

MR1 deviation of makespans and MMR deviation of makespans from BKSs and the average computational time of MMR for each Tillard's instance is shown in Table 16, Table 17, Table 18, and Table 19. If MMR has a better performance than MR1 the solution is highlighted in bold.

Table 16. Computational Results on 20-job Taillard's Benchmark (MMR)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	MMR	MMR Deviation	Average time (s)
TA001	20*5	1278	1278	0.00	1278	0.00	12
TA002		1359	1365	0.44	**1359**	0.00	
TA003		1081	1104	2.13	**1085**	0.37	
TA004		1293	1306	1.01	**1297**	0.31	
TA005		1235	1250	1.21	**1236**	0.08	
TA006		1195	1195	0.00	1195	0.00	
TA007		1239	1241	0.16	**1239**	0.00	
TA008		1206	1207	0.08	**1206**	0.00	
TA009		1230	1238	0.65	**1234**	0.33	
TA010		1108	1126	1.62	**1109**	0.09	
TA011	20*10	1582	1604	1.39	1596	0.88	13
TA012		1659	1691	1.93	1682	1.39	
TA013		1496	1514	1.20	1505	0.60	
TA014		1377	1398	1.53	1379	0.15	
TA015		1419	1442	1.62	1425	0.42	
TA016		1397	1421	1.72	1412	1.07	
TA017		1484	1494	0.67	1486	0.13	
TA018		1538	1557	1.24	1555	1.11	
TA019		1593	1620	1.69	1615	1.38	
TA020		1591	1617	1.63	1598	0.44	
TA021	20*20	2297	2322	1.09	**2321**	1.04	14
TA022		2099	2106	0.33	**2105**	0.29	
TA023		2326	2347	0.90	2356	1.29	
TA024		2223	2227	0.18	2231	0.36	
TA025		2291	2323	1.40	**2314**	1.00	
TA026		2226	2263	1.66	**2244**	0.81	
TA027		2273	2312	1.72	**2298**	1.10	
TA028		2200	2220	0.91	2220	0.91	
TA029		2237	2260	1.03	**2252**	0.67	
TA030		2178	2210	1.47	**2204**	1.19	

Table 17. Computational Results on 50-job Taillard's Benchmark (MMR)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	MMR	MMR Deviation	Average time (s)
TA031	50*5	2724	**2724**	0.00	2724	0.00	134
TA032		2834	2843	0.32	**2838**	0.14	
TA033		2621	2622	0.04	2622	0.04	
TA034		2751	2761	0.36	2761	0.36	
TA035		2863	2864	0.03	**2863**	0.00	
TA036		2829	**2829**	0.00	2829	0.00	
TA037		2725	**2725**	0.00	2725	0.00	
TA038		2683	**2683**	0.00	2683	0.00	
TA039		2552	2554	0.08	**2552**	0.00	
TA040		2782	**2782**	0.00	2782	0.00	
TA041	50*10	2991	3086	3.18	**3026**	1.17	135
TA042		2867	2953	3.00	**2950**	2.90	
TA043		2839	2950	3.91	**2922**	2.92	
TA044		3063	3107	1.44	**3096**	1.08	
TA045		2976	3075	3.33	3075	3.33	
TA046		3006	3104	3.26	**3093**	2.89	
TA047		3093	3178	2.75	3178	2.75	
TA048		3037	3088	1.68	**3083**	1.51	
TA049		2897	2964	2.31	**2960**	2.17	
TA050		3065	3169	3.39	**3167**	3.33	
TA051	50*20	3850	4007	4.08	**3991**	3.66	170
TA052		3704	3864	4.32	**3856**	4.10	
TA053		3640	3796	4.29	**3792**	4.18	
TA054		3720	3857	3.68	**3823**	2.77	
TA055		3610	3879	7.45	**3828**	6.04	
TA056		3681	3816	3.67	**3812**	3.56	
TA057		3704	3874	4.59	**3825**	3.27	
TA058		3691	3898	5.61	**3841**	4.06	
TA059		3743	3878	3.61	**3845**	2.73	
TA060		3756	3874	3.14	3874	3.14	

Table 18. Computational Results on 100-job Taillard's Benchmark (MMR)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	MMR	MMR Deviation	Average time (s)
TA061	100*5	5493	5493	0.00	5493	0.00	900
TA062		5268	5280	0.23	5270	**0.04**	
TA063		5175	5179	0.08	5176	**0.02**	
TA064		5014	5021	0.14	5017	**0.06**	
TA065		5250	5250	0.00	5250	0.00	
TA066		5135	5135	0.00	5135	0.00	
TA067		5246	5249	0.06	5248	**0.04**	
TA068		5094	5097	0.06	5097	0.06	
TA069		5448	5449	0.02	5450	0.04	
TA070		5322	5328	0.11	5328	0.11	
TA071	100*10	5770	5807	0.64	5802	**0.55**	1100
TA072		5349	5394	0.84	5385	**0.67**	
TA073		5676	5713	0.65	5692	**0.28**	
TA074		5781	5895	1.97	5879	**1.70**	
TA075		5467	5562	1.74	5553	**1.57**	
TA076		5303	5335	0.60	5331	**0.53**	
TA077		5595	5648	0.95	5645	**0.89**	
TA078		5617	5695	1.39	5689	**1.28**	
TA079		5871	5940	1.18	5951	1.36	
TA080		5845	5903	0.99	5893	**0.82**	
TA081	100*20	6202	6493	4.69	6449	**3.98**	1500
TA082		6183	6414	3.74	6411	**3.69**	
TA083		6271	6518	3.94	6477	**3.28**	
TA084		6269	6480	3.37	6470	**3.21**	
TA085		6314	6541	3.60	6524	**3.33**	
TA086		6364	6622	4.05	6590	**3.55**	
TA087		6268	6510	3.86	6518	3.99	
TA088		6401	6685	4.44	6679	**4.34**	
TA089		6275	6537	4.18	6509	3.73	
TA090		6434	6640	3.20	6610	**2.74**	

Table 19. Computational Results on 200-job Taillard's Benchmark (MMR)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	MMR	MMR Deviation	Average time (s)
TA091	200*10	10862	10892	0.28	10874	**0.11**	2450
TA092		10480	10614	1.28	10582	**0.97**	
TA093		10922	11017	0.87	11000	**0.71**	
TA094		10889	10921	0.29	10893	**0.04**	
TA095		10524	10575	0.48	10558	**0.32**	
TA096		10329	10338	0.09	10373	0.43	
TA097		10854	10934	0.74	10923	**0.64**	
TA098		10730	10798	0.63	10781	**0.48**	
TA099		10438	10501	0.60	10484	**0.44**	
TA100		10675	10758	0.78	10741	**0.62**	
TA101	200*20	11195	11195	3.39	11523	**2.93**	3800
TA102		11203	11203	3.98	11631	**3.82**	
TA103		11281	11281	4.32	11695	**3.67**	
TA104		11275	11275	3.62	11631	**3.16**	
TA105		11259	11259	3.03	11516	**2.28**	
TA106		11176	11176	3.82	11524	**3.11**	
TA107		11360	11360	3.57	11718	**3.15**	
TA108		11334	11334	3.43	11710	**3.32**	
TA109		11192	11192	3.70	11576	**3.43**	
TA110		11288	11288	4.05	11664	**3.33**	

A paired t-test is performed to test the effectiveness of memory design and validate if memory implementation is statistically improving the performance of Meta-RaPS. Average performance of MR1 and MMR for each Tillard's benchmark problem size alongside the p-value is shown in Table 20. The null and the alternative hypothesis are as follows:

$$\begin{cases} H_{null}: & There\ is\ no\ signifant\ different\ between\ MMR\ and\ Meta-RaPS \\ H_{alternative}: & MMR\ outperforms\ Meta-RaPS \end{cases}$$

Or

$$\begin{cases} H_{null}: & RPD_{MMR} = RPD_{Meta-RaPS} \\ H_{alternative}: & RPD_{MMR} < RPD_{Meta-RaPS} \end{cases}$$

It can be seen in Table 20 that in all cases (except in the problem sizes 50*5 and 10*5) MMR is outperforming Meta-RaPS with 95% confidence and the null hypothesis is rejected in favor of alternative hypothesis. To validate the overall performance of MMR and Meta-RaPS the p-values from different problems sizes should be combined. If p-values are independent, Fisher (1925) suggests a method to integrate extreme values (p-values) from different tests to one Chi-square (x^2) test. Fisher's Method is as follows where p_i is the p-value from i^{th} test and n is number of tests to combine:

$$-2\sum_{i=1}^{n}\log(p_i) \sim x_{2n}^2$$

The p-value from Fisher's Method is shown in the third column and the last row of Table 20. Fisher's p-value strongly states that MMR and Meta-RaPS are statistically different.

Table 20. Meta-RaPS and MMR Computations Results Comparison

Problem size	Meta-RaPS1	MMR (RPD)	p-value
20*5	0.731	0.118	0.008
20*10	1.462	0.758	0.000
20*20	1.069	0.704	0.016
50*5	0.083	0.054	0.074
50*10	2.824	2.41	0.032
50*20	4.443	3.75	0.003
100*5	0.069	0.036	0.067
100*10	1.095	0.967	0.011
100*20	3.906	3.583	0.003
200*10	0.604	0.475	0.022
200*20	3.691	3.22	0.000
All	1.81	1.461	0.000

4-4 Tie Breaking Strategy Computational Results

Tie breaking strategy is proposed since Meta-RaPS is facing considerable amount of ties in the candidate list (CL) when applied in the second phase of NEH. Therefore, adding a mechanism to select a better position when there exists a tie is favorable. Integration of tie breaking strategy and MR2 is illustrated in section 3-3. Methods are tested on Tillard's benchmark with 5 independent runs and the best solution is reported from all runs. The algorithm is coded in Matlab 2011 with an Intel CORE-i5 CPU @ 2.5GHz and 4GB installed memory.

Performance of Tie breaking strategy for each instance is measured by the deviation of C_{max} from the BKS in OR-library as of April 2017 as follows:

$$Dev = \frac{Cmax_{sol} - BKS}{BKS} * 100$$

MR2 makespans' deviation and MR2 with tie breaking strategy makespans' deviation from BKSs and the average computational time for MR2 with tie breaking mechanism for each Tillard's instance is shown in Table 21, Table 22, Table 23, and Table 24. If a tie breaking strategy finds a better solution it is highlighted in bold.

Table 21. Computational Results on 20-job Taillard's Benchmark (Tie Breaking)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	Tie Breaking	Tie breaking Deviation	Average time (s)
TA001	20*5	1278	1297	1.49	1297	1.49	1
TA002		1359	1383	1.77	1383	1.77	
TA003		1081	1116	3.24	1107	**2.41**	
TA004		1293	1302	0.70	1302	0.70	
TA005		1235	1283	3.89	1283	3.89	
TA006		1195	1230	2.93	1230	2.93	
TA007		1239	1246	0.56	1246	0.56	
TA008		1206	1216	0.83	1212	**0.50**	
TA009		1230	1253	1.87	1245	**1.22**	
TA010		1108	1122	1.26	1122	1.26	
TA011	20*10	1582	1634	3.29	1629	**2.97**	2
TA012		1659	1682	1.39	1682	1.39	
TA013		1496	1517	1.40	1515	**1.27**	
TA014		1377	1397	1.45	1399	1.60	
TA015		1419	1444	1.76	1437	**1.27**	
TA016		1397	1427	2.15	1427	2.15	
TA017		1484	1503	1.28	1503	1.28	
TA018		1538	1577	2.54	1569	**2.02**	
TA019		1593	1623	1.88	1623	1.88	
TA020		1591	1627	2.26	1619	**1.76**	
TA021	20*20	2297	2328	1.35	2325	**1.22**	5
TA022		2099	2155	2.67	2125	**1.24**	
TA023		2326	2341	0.64	2341	0.64	
TA024		2223	2235	0.54	2233	**0.45**	
TA025		2291	2331	1.75	2321	**1.31**	
TA026		2226	2272	2.07	2272	2.07	
TA027		2273	2311	1.67	2311	1.67	
TA028		2200	2227	1.23	2226	**1.18**	
TA029		2237	2260	1.03	2259	**0.98**	
TA030		2178	2201	1.06	2202	1.10	

Table 22. Computational Results on 50-job Taillard's Benchmark (Tie Breaking)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	Tie Breaking	Tie breaking Deviation	Average time (s)
TA031	50*5	2724	2274	0.00	2724	0.00	19
TA032		2834	2848	0.49	2848	0.49	
TA033		2621	2625	0.15	2625	0.15	
TA034		2751	2762	0.40	2762	0.40	
TA035		2863	2864	0.03	2889	0.91	
TA036		2829	2831	0.07	2831	0.07	
TA037		2725	2732	0.26	2725	**0.00**	
TA038		2683	2684	0.04	2683	**0.00**	
TA039		2552	2552	0.00	2552	0.00	
TA040		2782	2786	0.14	2782	**0.00**	
TA041	50*10	2991	3101	3.68	3093	**3.41**	26
TA042		2867	2978	3.87	2977	**3.84**	
TA043		2839	2936	3.42	2952	3.98	
TA044		3063	3103	1.31	3109	1.50	
TA045		2976	3093	3.93	3092	**3.90**	
TA046		3006	3110	3.46	3099	**3.09**	
TA047		3093	3179	2.78	3185	2.97	
TA048		3037	3102	2.14	3088	**1.68**	
TA049		2897	2971	2.55	2971	2.55	
TA050		3065	3162	3.16	3152	**2.84**	
TA051	50*20	3850	3994	3.74	4002	3.95	39
TA052		3704	3860	4.21	3845	**3.81**	
TA053		3640	3812	4.73	3828	5.16	
TA054		3720	3884	4.41	3890	4.57	
TA055		3610	3874	7.31	3897	7.95	
TA056		3681	3833	4.13	3811	**3.53**	
TA057		3704	3893	5.10	3888	**4.97**	
TA058		3691	3843	4.12	3851	4.33	
TA059		3743	3892	3.98	3901	4.22	
TA060		3756	3931	4.66	3931	4.66	

Table 23. Computational Results on 100-job Taillard's Benchmark (Tie Breaking)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	Tie Breaking	Tie breaking Deviation	Average time (s)
TA061	100*5	5493	5493	0.00	5493	0.00	48
TA062		5268	5275	0.13	5268	**0.00**	
TA063		5175	5192	0.33	5192	0.33	
TA064		5014	5023	0.18	5023	0.18	
TA065		5250	5255	0.10	5255	0.10	
TA066		5135	5135	0.00	5139	**0.08**	
TA067		5246	5257	0.21	5256	**0.19**	
TA068		5094	5100	0.12	5099	**0.10**	
TA069		5448	5454	0.11	5454	0.11	
TA070		5322	5328	0.11	5322	**0.00**	
TA071	100*10	5770	8520	0.87	5816	**0.80**	58
TA072		5349	5405	1.05	5399	**0.93**	
TA073		5676	5708	0.56	5728	0.92	
TA074		5781	5949	2.91	5924	**2.47**	
TA075		5467	5601	2.45	5573	**1.94**	
TA076		5303	5358	1.04	5346	**0.81**	
TA077		5595	5659	1.14	5666	1.27	
TA078		5617	5707	1.60	5711	1.67	
TA079		5871	5983	1.91	5955	**1.43**	
TA080		5845	5909	1.09	5903	**0.99**	
TA081	100*20	6202	6533	5.34	6530	**5.29**	89
TA082		6183	6537	5.73	6519	**5.43**	
TA083		6271	6582	4.96	6580	**4.93**	
TA084		6269	6566	4.74	6550	**4.48**	
TA085		6314	6620	4.85	6653	5.37	
TA086		6364	6673	4.86	6653	**4.54**	
TA087		6268	6606	5.39	6582	**5.01**	
TA088		6401	6710	4.83	6754	5.51	
TA029		2237	6570	4.70	6592	5.05	
TA030		2178	6615	2.81	6600	**2.58**	

Table 24. Computational Results on 200-job Taillard's Benchmark (Tie Breaking)

Problem	n*m	BKS	Meta-RaPS	Meta-RaPS Deviation	Tie Breaking	Tie breaking Deviation	Average time (s)
TA091	200*10	10862	10897	0.32	10885	**0.21**	380
TA092		10480	10673	1.84	10635	**1.48**	
TA093		10922	11017	0.87	11009	**0.80**	
TA094		10889	10929	0.37	10931	0.39	
TA095		10524	10586	0.59	10600	0.72	
TA096		10329	10396	0.65	10402	0.71	
TA097		10854	10947	0.86	10941	**0.80**	
TA098		10730	10789	0.55	10761	**0.29**	
TA099		10438	10512	0.71	10502	**0.61**	
TA100		10675	10767	0.86	10762	**0.81**	
TA101	200*20	11195	11572	3.37	11574	**3.39**	435
TA102		11203	11665	4.12	11649	**3.98**	
TA103		11281	11779	4.41	11768	**4.32**	
TA104		11275	11663	3.44	11683	3.62	
TA105		11259	11630	3.30	11600	**3.03**	
TA106		11176	11609	3.87	11603	**3.82**	
TA107		11360	11782	3.71	11766	**3.57**	
TA108		11334	11783	3.96	11723	**3.43**	
TA109		11192	11630	3.91	11606	**3.70**	
TA110		11288	11763	4.21	11745	**4.05**	

A paired t-test is performed to test the effectiveness of tie breaking strategy and validate if the implementation of tie breaking is statistically improving the performance of Meta-RaPS. Performance of MR2 and MR2 with tie breaking strategy is compared in Table 25 where p-value of each problem size is shown in the last column of the Table. The null and the alternative hypothesis are as follows:

$$\begin{cases} H_{null}: & There\ is\ no\ statistical\ different\ between\ MR2\ and\ MR2\ with\ Tie\ breaking \\ H_{alternative}: & Tie\ Breaking\ strategy\ outperforms\ MR2 \end{cases}$$

Or

$$\begin{cases} H_{null}: & RPD_{MR2\ with\ Tie\ Breaking} = RPD_{MR2} \\ H_{alternative}: & RPD_{MR2\ with\ Tie\ Breaking} < RPD_{MR2} \end{cases}$$

P-values of Table 25 illustrates the fact that although the tie breaking strategy descreses the average performance of the algorithms in many cases but this improvement is not statistically significant and this is due to the fact that the improvement does not happen in all instances. Therefore, there is no reason to reject the null hypothesis .To validate the overall performance of MR2 and MR2 with tie breaking strategy, p-values from different problem sizes should be combined. If p-values are independent, Fisher (1925) suggests a method to integrate extreme values (p-values) from different tests to one Chi-square (x^2) test. Fisher's Method is as follows where p_i is the p-value from i^{th} test and n is number of tests to combine:

$$-2\sum_{i=1}^{n}\log(p_i) \sim x_{2n}^2$$

Fisher's p-value represents the fact that even though the tie breaking strategy does not outperform MR2 and alternative hypothesis is rejected in favor of null hypothesis, but with a 95% confidence interval the tie breaking strategy indeed outperforms MR2 in overall.

Table 25. MR2 and MR2 with Tie Breaking Strategy Comparison

Problem size	Meta-RaPS2	Tie Breaking	p-value
20*5	1.853	1.671	0.051
20*10	1.94	1.758	0.024
20*20	1.4	1.187	0.084
50*5	0.159	0.203	0.331
50*10	3.03	2.977	0.302
50*20	4.639	4.715	0.266
100*5	0.129	0.108	0.154
100*10	1.462	1.324	0.077
100*20	4.82	4.820	0.499
200*10	0.762	0.682	0.059
200*20	3.832	3.691	0.020
All	2.18	2.103	0.007

4-5 Algorithms Comparison

Two basic Meta-RaPS designs alongside a design with memory and learning and a design with tie breaking strategy are discussed in this thesis. This chapter provides a comprehensive comparison for all of these four designs. The second part of this chapter provides a comparison between the performance of MMR and the best algorithms of PFSP.

4-5-1 Meta-RaPS Designs Compariosn

Meta-RaPS in the first phase of NEH (MR1) performs better than Meta-RaPS in the second phase of NEH (MR2) with C_{max} objective function. Meta-RaPS in the second phase of NEH has a much better performance than NEH heuristic but the performance is not competitive with Meta-RaPS in the first phase. Therefore, a tie breaking strategy is added to MR2 to help the algorithm.

The main objective of this research is to design a generic memory structure for the construction phase of Meta-RaPS. Memory design is discussed in chapter 3-2-1, PFSP is introduced as the application of memory design, this design is entitled “MMR”. Performance of these four designs is compared to each other in Figure 10.

Figure 10 depicts the fact that when the number of machines increases all designs effectiveness decreases. MMR has the best performance in all cases when compared to other three designs. Therefore, it can be concluded that memory implementation in the construction phase of Meta-RaPS improves the performance of the meta-heuristic.

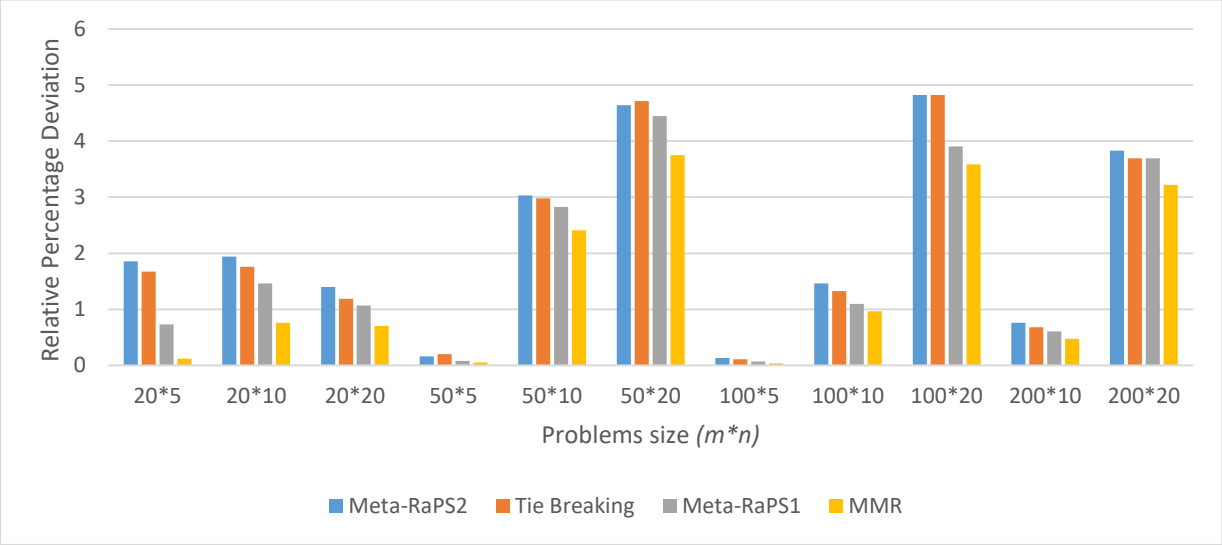

Figure 10. Meta-RaPS Designs Comparison

4-5-2 MMR Comparison with Other Algorithms

MMR has the strongest performance among all discussed Meta-RaPS designs for PFSP. To validate the effectiveness of MMR the best algorithms in the literature with the criterion of makespan are compared to MMR in Table 26. A simulated annealing algorithm (SAOP) by Osman and Potts (1986), a Tabu Search by Widmer and Hertz endtitled Spirit (1989), an ant colony optimization by Rajendran and Ziegler entitled M-MMAS (2004), a hybrid meta-heuristic by Zobolas, Tarantilis, and Ioannou (2009), and an iterative greedy search by Ruiz and Stützle (2007) are selected from the literature.

Table 26. MMR Comparisons with Competitive Algorithms in the Literature

Algorithm	NEH	SAOP	SPIRIT	GA_AA	IG_RS	NEGAVNS	MMR
20*5	3.26	1.09	4.77	0.94	0.04	0.00	0.12
20*10	4.60	2.63	5.61	1.54	0.25	0.01	0.76
20*20	3.73	2.38	4.44	1.43	0.21	0.02	0.70
50*5	0.73	0.52	2.19	0.36	0.04	0.00	0.05
50*10	5.07	3.51	6.04	3.72	1.06	0.82	2.41
50*20	6.66	4.52	7.63	4.69	1.82	1.08	3.75
100*5	0.53	0.30	1.06	0.32	0.05	0.00	0.04
100*10	2.28	1.48	3.01	1.72	0.39	0.14	0.97
100*20	5.35	4.63	6.74	4.91	2.04	1.40	3.58
200*10	1.26	1.01	2.07	1.27	0.34	0.16	0.47
200*200	4.42	3.81	4.97	4.21	1.99	1.25	3.22
All	3.44	2.35	4.41	2.28	0.75	0.44	1.46

Table 26 highlights the effectiveness of MMR in PFSP with makespan criterion. Among all the algorithms just IG_RS and NEHA_VNS perform better than MMR. However, it should not be forgetten that NEHA_VNS is a hybrid local search algorithm and IG_RS is not a pure construction heuristic since it starts iterating over a given solution.

5-BEYOND THE SCOPE

The main purpose of this research is to investigate the effectiveness of embedding local search memory and learning techniques in the construction phase of a memoryless meta-heuristic entitled "Meta-RaPS. The computational results show that a memory mechanism improves the performance of Meta-RaPS and also, the algorithms is competitive with the best algorithms in the literature but there exist better algorithms.

Meta-RaPS has two phases, the construction phase and the improvement phase. In the section 4-1 is it mentioned that the scope of this research is restricted to the construction phase of Meta-RaPS. Therefore, the improvement parameter (i) does not need to be tuned and is considered zero. In this section a simple local search technique is introduced to confirm adding a local search method to an algorithm will improve the performance of the algorithm

5-1 Iterated Greedy Algorithm

Lourenco, Martin and Stutzle (2002) describe iterated greedy algorithms as a mechanism as follows:

Generate a complete feasible solution, destruct the solution with removing some variables and then applying a greedy heuristic to construct a complete feasible solution. When a new solution is constructed, a criterion decides whether to accept the solution or alternatively not.

Ruiz et al. (2007) introduces the first version of iterated greedy for PFSP noted as IG_RS and the algorithm is as follows:

1- Apply the destruction phase to a permutation π to remove d jobs randomly without repetition and keep the jobs in a set π_r with the order of removing. Now there are two sets, one with $\pi - d$ jobs which is entitled π_d and another set with d jobs noted as π_r
2- Third step of NEH heuristic is the construction phase. Start with the first job in set π_r and insert the job in all positions of π_d. Make a job's position permanent in the cheapest position of π_d. Continue the process for all d available jobs in π_d.

Ruiz et al. (2007) considers the above described procedure as the construction phase of IG_RS. Hence, a local search mechanism is added to process to enhance the results. The utilized local search mechanism is neighborhood insertion which is removing a random job from permutation π and insert it in a new position k.

Construction phase of IG_RS is considered as a the improvement phase of MMR to enhance the solutions and the local search method in IG_RS is not part of the improvement phase of MMR

5-2 Computational Results

IG_RS adds a new parameter d (the number of jobs to remove) to the algorithm. Ruiz et al. (2007) obtained $d = 4$ as the best value for number of jobs to remove. Therefore, destruction size in this research would be the same as Ruiz et al. (2007) setting.

The construction phase of MMR constructs a feasible solution with the makespan $C_{max} = Z$, a constructed solution will go through the improvement phase if:

$$Z \leq b^* + (b^* - w^*) * i\%$$

Where b^* is the best found solution before the improvement phase, and w^* is the worst found solution before the improvement phase.

In Section 4-1 the parameter i is considered zero while now the parameter should be tuned, rest of the parameter as the same as Table 15. According to Moraga's (2002) tuning process:

1. Selecting a subset of problems: problems are selected randomly from Tillard's benchmark (3 small size problems, 3 large size problems)
2. The domain that parameter i will vary is {30%, 40%, 50%, 60%, 70%, 80%, 90%}
3. While other parameters are constant and the same as Table 15 manipulate the values of i and report a value associate to the best objective functions.

Summary of tuning parameter (i) is shown in Figure 11 and Figure 12. Surprisingly it can be seen that increasing the value of i does not lead to a significant improvement in the objective function

C_{max} while the computational time increases significantly. This is due to allowing more solution through the improvement phase.

The range for RPD in Figure 12 is around 0.09 so it can be concluded that the increasing the parameter I does not improve the algorithm's performance significantly while it increases the computational times exponentially. Interpretation of Figures 11 and 12 indicates that the best value for parameter *i* equals $= 0.5$.

Performance of MMR and MMR with an improvement phase is depicted is Table 27. The results indicate that adding a local search increase the performance of an algorithm. However, it is worth it to mention that the iterated greed local search does not lead to a huge improvement in MMR while Ruiz et al (2007) obtains extremely competitive results with a worst starting point.

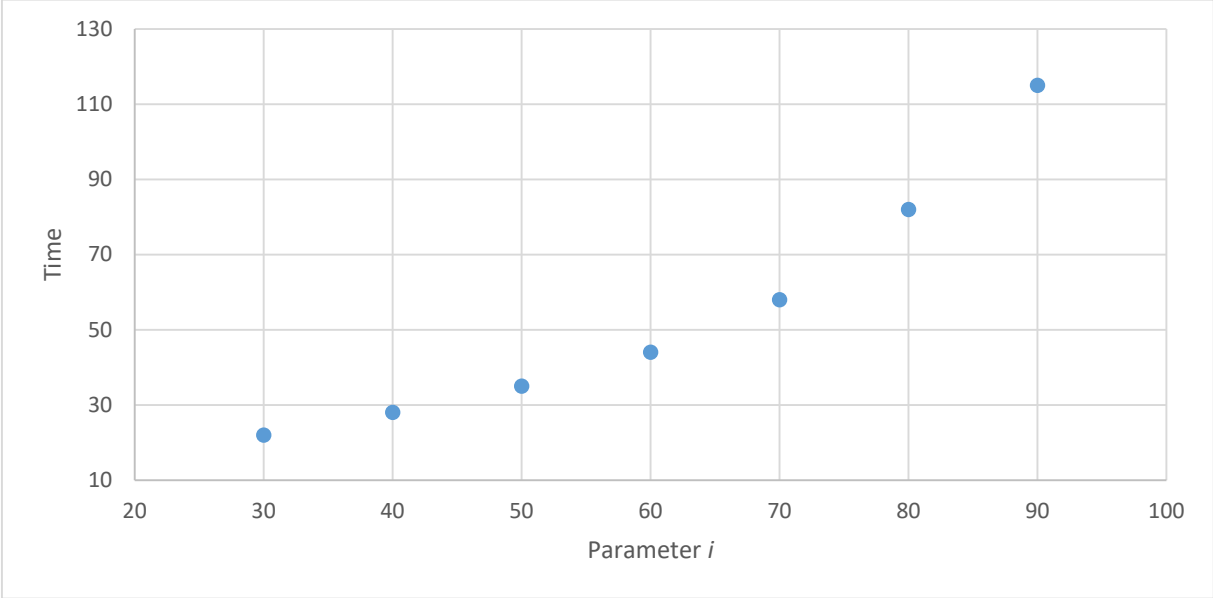

Figure 11. Computational Time for Different Values of i

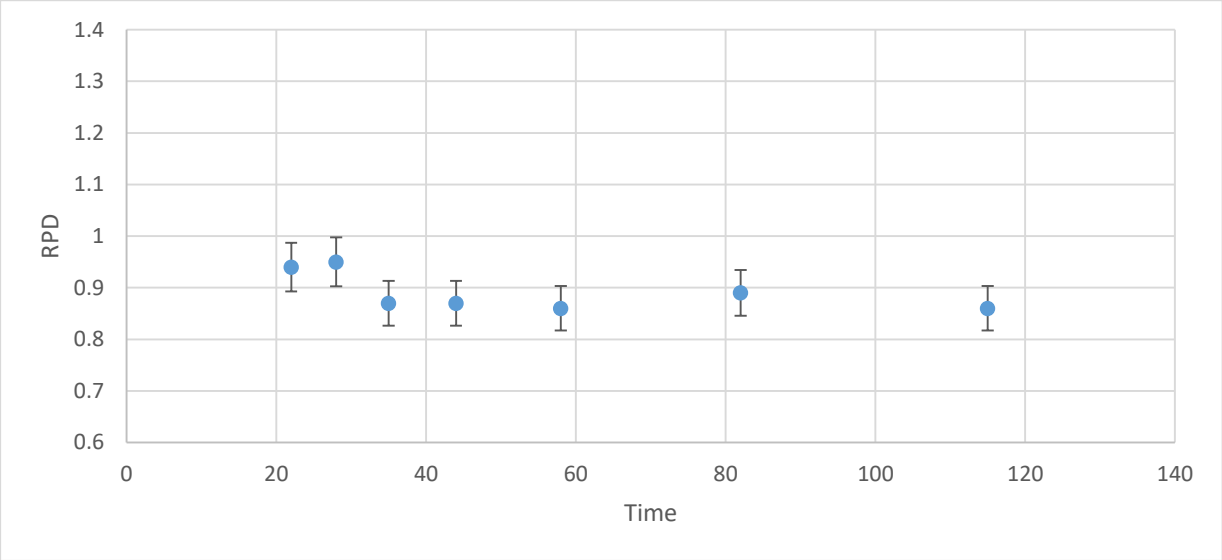

Figure 12. Average RPD for an Associated Time (Time is Related to i)

Table 27. RPD for MMR with Locals Seach and MMR without Local Search

Problem size (m*n)	MMR (%)	MMR with Local Search (%)
20*5	0.12	0.08
20*10	0.76	0.82
20*20	0.7	0.55
50*5	0.05	0.02
50*10	2.41	2.20
50*20	3.75	3.66
100*5	0.04	0.05
100*10	0.97	0.96
100*20	3.58	3.56
200*10	0.47	0.41
200*20	3.22	2.96
All	1.461	1.388

The belief of the authors is that Ruiz et al (2007) obtains better performance in iterated greedy due to spending more time on destructing and constructing a solution. To validate this hypothesis the algorithms is run on the hardest problem sizes (50*10 and 50*20) in Tillard's benchmark with different time spans. The computational results confirm that increasing the time on improves the effectiveness the improvement phase. Table 28 shows two different settings for iterated greedy allowed time span in the local search. The results signify the hypothesis that if the time increases, the object function improves.

Table 28. Local Search Performance for Different Time Spans

Problem	n*m	BKS	Local Search with 20 iterations	RPD	Local Search with 80 Iterations	RPD
TA041	50*10	2991	3080	2.98	3022	1.04
TA042		2867	2934	2.34	2946	2.76
TA043		2839	2914	2.64	2919	2.82
TA044		3063	3095	1.04	3088	0.82
TA045		2976	3051	2.52	3064	2.96
TA046		3006	3066	2.00	3055	1.63
TA047		3093	3154	1.97	3144	1.65
TA048		3037	3076	1.28	3071	1.12
TA049		2897	2954	1.97	2954	1.97
TA050		3065	3166	3.30	3160	3.10
Average			**3049**	**2.20**	**3042**	**1.98**
TA051	50*20	3850	3985	3.51	3964	2.96
TA052		3704	3848	3.89	3838	3.62
TA053		3640	3788	4.07	3771	3.60
TA054		3720	3823	2.77	3818	2.63
TA055		3610	3825	5.96	3812	5.60
TA056		3681	3812	3.56	3802	3.29
TA057		3704	3821	3.16	3800	2.59
TA058		3691	3838	3.98	3804	3.06
TA059		3743	3842	2.64	3836	2.48
TA060		3756	3873	3.12	3866	2.93
Average			**3845**	**3.66**	**3831**	**3.28**

6-CONCLUSION

The purpose of this research is to design a generic memory mechanism for the construction phase of a memoryless meta-heuristic algorithm entitled “Meta-RaPS” with the application of PFSP (Permutation Flowshop Scheduling Problem) to minimize an objective function known as makespan. The first memory design in construction phase of Meta-RaPS was introduced by Lan et al (2007) with the application of SCP (Set Covering Problem). However, a few researches have been carried out to implement memory and/or learning in Meta-RaPS since then. Even fewer researches have been done in the realm of meta-heuristics to incorporate memory mechanisms in the construction phase of optimization algorithms. The main contribution of this research is to provide a novel generic memory structure for the construction phase of Meta-RaPS and adjusting the structure according to characteristic of PFSP. In addition to memory structure, the illustrated methods to construct high-quality solutions and computational results are the other contributions of this research. This thesis demonstrated that the performance of a memoryless algorithm can be enhanced with an embedded artificial intelligence.

6-1 Summary

Objective of this research is to improve the effectiveness of the construction phase of Meta-RaPS with the application of PFSP. NEH is known to be the best construction heuristic for PFSP. NEH has two phases: sequencing phase, and the cheapest insertion phase. The computational results

confirm that Meta-RaPS is a powerful algorithm to solve PFSP. Meta-RaPS in the first phase of NEH has a better performance than Meta-RaPS in the second phase of NEH. Therefore, this setting is used for further analysis in memory design. In Section 3-2 a generic memory design for construction phase of Meta-RaPS is introduced and then a learning mechanism is added to the algorithm to bias the search toward high quality solutions, the algorithm is noted as MMR. MMR is tested with famous Tillard's benchmark and the computational results strengthens the hypothesis that adding memory and learning mechanisms to a memoryless algorithm can enhance its performance and its capability of finding high-quality solutions. It is observed that when the number of machines are relatively low MMR can find almost all the BKSs (Best Known Solution) of PFSP.

Comparison of MMR and the best algorithms in the literature shows that although MMR is just a construction algorithm but there are few algorithms with better performance for PFSP with makespan criterion. Meta-RaPS has two phases, the construction phase and the improvement phase but MMR does not benefit from the improvement phase. Therefore, to increase the competency of MMR a simple local search technique, Iterated greedy which is introduced by Ruiz et al (2007) as a construction heuristics, is utilized to prove if a local search technique is added to an algorithm, the performance increases.

Meta-RaPS in second phase of NEH faces significant amount of ties in the candidate list. Therefore, a tie breaking strategy is proposed to overcome the problem. The computational results show that the tie breaking algorithm is improving the performance of Meta-RaPS but not in all

cases and all problem sizes. This is due to the fact that, there is no global tie breaking strategy that can always enhance an algorithm's performance.

6-2 Future Research

Objective of this research is to enhance the performance of Meta-RaPS with implementing a memory and learning mechanism in the construction phase of this meta-heuristic algorithm. Memory mechanism is developed from the principles of an effective AMP by Tillard et al. (2001) and the learning mechanism is developed from principle of Tabu Search by Glover et al (1997) to influence the search toward good solutions. Future research can focus on designing a generic learning mechanism for the construction phase of Meta-RaPS.

Randomness improves the performance of an algorithms since it assists the algorithm to expand the domain of the search. Meta-RaPS has an advantage over randomness in several ways; however, there exist other ways to add more randomness to Meta-RaPS and expand the search further. A strategy to add randomness to current state of Meta-RaPS can be selecting the priority sequencing rule from a pool of rules in each iteration. For example is the case of PFSP this research uses Dong's priority rule is all 200 iterations while at each iteration the algorithm can select the priority rule randomly from some of the well-known greedy rules like Palmer's index, CDS, Avg, and AvgDev. This is due to the fact that memory mechanisms focus on the intensification aspect of AMP structure while another factor that plays a critical role in reaching to a global optimum is diversification. Therefore, having diverse high-quality solutions in the elite list helps to balancing the intensification and the diversification of the algorithm.

MMR is implemented in the first phase of NEH since Meta-RaPS has a better performance on NEH sequencing phase. Developing other applications of memory based Meta-RaPS based on the principles of MMR for other well-known NP-hard problems to verify the effectiveness of MMR is promising.

Moreover, adding a simple local search to the algorithm confirms that even a basic local search technique increases the effectiveness of MMR. Therefore, more studies should be done to investigate implementation of sophisticated techniques in the improvement phase of MMR.

6-REFERENCES

Allaoui, H., Artiba, A. (2006). Scheduling two-stage hybrid flow shop with availability constraints. *Computers & Operations Research*, 33, 1399-1419.

Arin, A., Rabadi, G (2013) Memory and Learning in Metaheuristics, *Artificial Intelligent Evolution Computer and Metaheuristics*, SCI 427, pp. 435–476.

Ben-Daya, M., Al-Fawzan, M. (1998). A tabu search approach for the flow shop scheduling problem. European *Journal of Operational Research*, 109: 88-95.

Campbell, H.G., Dudek, R.A. & Smith, M.L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management Science*, 16: 630-637.

Chen, C.-L., Vempati, V.S. & Aljaber, N. (1995). An application of genetic algorithms for flow shop problems. *European Journal of Operational Research*, 80: 389-396.

Coy, S. P., Golden, B. L., Runger, G. C., & Wasil, E. A. (2001). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1), 77-97.

Dannenbring, D.G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, 23: 1174-1182.

DePuy, G., Moraga, R., & Whitehouse, G. (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.

DePuy, G., Whitehouse, G., & Moraga, R. (2002). Using the Meta-RaPS Approach to Solve Combinatorial Problems. *Proceedings of the 2002 Industrial Engineering Research Conference.* Orlando, Florida.

Dong, X.-Y., Huang, H.-K. & Chen, P. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, 35: 3962-3968.

Fisher, R. A. (1925). Statistical methods for research workers. Genesis Publishing Pvt Ltd.

Garcia, C., Rabadi, G (2011) A Meta-RaPS algorithm for spatial scheduling with release times. *International Journal of Planning and Scheduling,* 1(1/2), 19–31

Glover, F., Laguna, M (1997) Tabu Search, University of Colorado, Boulder. Kluwer Academic Publishers, Boston

Glover, F., & Laguna, M. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3), 653–684.

Gupta, J.N.D. (1971). An improved combinatorial algorithm for the flowshop scheduling problem. *Operations Research*, 19: 1735-1758.

Graham, R., Lawler, E., Lenstra, J., & Kan, R, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics. 287-326.

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

Ignall, E., Schrage L. E (1965). Application of branch-and-bound techniques to some flowshop problems. *Operation Research*, 13:400–412

Johnson, L.A., Montgomery, D.C. (1974). *Operation research in production planning, scheduling, and inventory control.* (pp. 35-40). New York: John Wiley & Sons, Inc.

Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly,* 1: 61-68.

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

Lan, G., DePuy, G., & Whitehouse, G. (2007). An effective and simple heuristic for the set covering problem. *European Journal of Operational Research, 176*, 1387– 1403.

Li X, P., Wang Y, X, & Wu, C (2004). Heuristic algorithms for large flowshop scheduling problems. *Proceedings of the 5th world congress on intelligent control and automation*. Hangzhou China, 2999–3003.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. *In Handbook of metaheuristics* (pp. 320-353). Springer US.

Moccellin, J, A, V., Dos Santos, M, O. (2000). An adaptive hybrid meta-heuristic for permutation flowshop scheduling. *Control and Cybernetics*, 29: 761-771.

Moraga, R, J (2002) Meta-RaPS: an effective solution approach for combinatorial problems. Ph.D. Dissertation, University of Central Florida, FL 32816, USA.

Moraga, R. J., DePuy, G. W., & Whitehouse, G. E. (2005). Meta-RaPS approach for the 0-1 Multidimensional Knapsack Problem. *Computers & Industrial Engineering*, 83-96.

Moraga, R, J., Whitehouse, G., DePuy, G., Neyveli, B., & Kuttuva, S. (2001). Solving the Capacitated Vehicle Routing Problem using the Meta-RaPS approach. *Proceedings of the Conference on Computers and Industrial Engineering*, (pp. 76-81). Montreal, Canada.

Murata, T., Ishibuchi, H. & Tanaka, H. (1996). Genetic algorithms for flowshop scheduling problems. *Computer and Industrial Engineering,* 30: 1061-1071.

Nawaz, E, M., Enscore Jr, E. & Ham, I (1983). A heuristic algorithm for the mmachine, n-job flow-shop sequencing problem. OMEGA, 11(1):91–95

Nowicki, E., Smutnicki, C. (1996). A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research,* 91: 160- 175.

Ogbu, F., Smith, D. (1990). The application of the simulated annealing algorithms to the solution of the n/m/Cmax flowshop problem. *Computers & Operations Research,* 17: 243–253.

Osman, H., Laporte, G. (1996) Metaheuristics: A bibliography. Annals of Operations Research, 63:513–623, 1996.

Osman, I., Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *OMEGA, the International Journal of Management Science,* 17: 551-557.

Palmer, D.S. (1965). Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum. *Operational Research Quarterly,* 16: 10 107.

Pinedo, M, L (2008). Scheduling: Theory, Algorithms, and systems (3rd ed.). New York, NY: Springer.

Rabadi, G (2016). Heuristics, metaheuristics and Approximate Methods in Planning and Scheduling. New York, NY: Springer.

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

Rahman, H., Sarker, R., & Essam, D. (2015). A real-time order acceptance and scheduling approach for permutation flow shop problems. *European Journal of Operational Research,* 247, 488-503.

Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research,* 155(2), 426-438.

Reeves, C.R. (1995). A genetic algorithm for flowshop sequencing. *Computers & Operations Research,* 22: 5-13.

Ribas, I., Companys, R. & Tort-Martorell, X (2010). Comparing three-step heuristics for the permutation flow shop problem. *Computers and Operations Research*, 37(12):2062–2070, 2010.

Ruiz, R., Maroto, C. (2006). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research,* 165: 479- 494.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3), 2033-2049.

Sun, Y., Zhang, C., Gao, L., & Wang, X. (2011). Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. *International Journal of Advanced Manufacturing Technology*, 55, 723-739.

Taillard, E, D (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74, 1990.

Taillard, E, D (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research,* 64: 278-285.

Taillard, E, D. (1998). FANT: Fast Ant System. *Techical Report*, 46-98.

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

Turner, S., Booth, D., 1987. Comparison of heuristics for flow shop sequencing. *OMEGA, The International Journal of Management Science*, 15 (1), 75–78.

Vigas, F, V., Framinan, M, J (2014). On insertion tie-breaking rules in heuristics for permutation flowshop scheduling problem. *Computers & Operations Research*, 45(1), 60-67

Vigas, F, V., Ruiz, R. & Framinan, M, J (2016). A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation. *European Journal of Operation Research*, 257 (3), 707-721.

Widmer, M., & Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, 41(2), 186-193.

Zegarra-Ballón, D, B. (2009). On effectiveness of incorporating memory mechanisms into the construction phase of Meta-RaPS with application to the Unrelated Parellel Machine Problem. *Masters Thesis*. Northern Illinois University, DeKalb, Illinois, USA.

Zegordi, S. H., Itoh, K. & Enkawa, T. (1995). Minimizing makespan for flowshop scheduling by combining simulated annealing with sequencing knowledge. *European Journal of Operational Research,* 85: 515-531.

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2009). Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. *Computers & Operations Research,* 36(4), 1249-1267.