University of Benghazi
Faculty of Education Almarj

جامعة بنغازي
كلية التربية – المرج
ISSN 2518-5845

Global Libyan Journal
المجلة الليبية العالمية

العدد السابع والأربعون / أبريل / 2020

Solve job shop scheduling problem with sequence dependent setup times using a local search algorithm with different neighborhood search structure

Mohammad Mahmoud Hamed , Hamzah A.Mohamed , Somaia Eeshim
Department of industrial and Manufacturing systems

University of Benghazi
Faculty of Education Almarj

جامعة بنغازي
كلية التربية – المرج
ISSN 2518-5845

Global Libyan Journal

المجلة الليبية العالمية

العدد السابع والأربعون / أبريل / 2020

Solve job shop scheduling problem with sequence dependent setup times using a local search algorithm with different neighborhood search structure

Abstract:

In this study examines job shop scheduling problems with sequence dependent setup times under objective function minimization of makespan (JSSP/SDST/$C_{\max}$). An effective meta-heuristic, local search is a meta-heuristic method for solving computationally hard optimization problems. Local search can be used on problems that can be formulated as finding a solution maximizing or minimizing a criterion among a number of candidate solutions. Local search algorithms move from solution to solution in the space of candidate solutions (the search space) by applying local changes, until a solution deemed optimal is found or a time bound is elapsed. The performance of the local search depends on its neighborhood search structure (NSS). We used five methods from neighborhood search: Swap, Migration Mechanism (MM), Inversion, shift, and a proposed robust neighborhood search method. The results showed that the new PNS method gives less makespan value with different problems size (15x15, 20x15, 20x20, 30x15, 30x20, 50x15, 50x20 and 100x20) taken from the OR- library compared to previous well known neighborhood search methods.

Key word: Job shop scheduling, Sequence-dependent setup times, local search, neighborhood search structure, makespan.

حل مشكلة الجدولة ورشة العمل مع أوقات الإعداد التابعة للتسلسل باستخدام خوارزمية البحث المحلي بهيكل بحث حي مختلف

الملخص:

في هذه الدراسة يدرس مشاكل جدولة ورشة العملJSSP مع أوقات الإعداد المعتمدة على التسلسل في إطار تقليل الوظيفة الموضوعية لـ makepan (JSSP / SDST /). يعتبر البحث المحلي الفعال-الاستدلالي- طريقة بحثية--meta heuristic لحل مشكلات التحسين الصعب حسابيًا. يمكن استخدام البحث المحلي في المشكلات التي يمكن صياغتها كإيجاد حل يزيد أو يقلل معيارًا بين عدد من الحلول المرشحة. تنتقل خوارزميات البحث المحلية من حل إلى حل في مساحة الحلول المرشحة (مساحة البحث) من خلال تطبيق التغييرات المحلية ، حتى يتم العثور على حل يُعتبر الأمثل أو انقضاء مهلة زمنية محددة. يعتمد أداء البحث المحلي على هيكل البحث عن الأحياء (NSS) الخاص به. استخدمنا خمس طرق من البحث عن

الأحياء: المبادلة ، وآلية الترحيل (MM) ، والانقلاب ، والتحول ، وطريقة بحث حي قوية مقترحة. أوضحت النتائج أن طريقة PNS الجديدة تعطي قيمة أقل للقيمة مع حجم المشاكل المختلفة (15 x15، 20x15، 20x20، 30x15، 30x20، 50x15، 50x20 و x20100) مأخوذة من موقع OR- مقارنة بطرق البحث السابقة المعروفة جيدًا. في أوقات الإعداد المستقلة قارنا النتيجة مع أفضل حل من موقع OR ، وكانت النتائج تشير إلى أن هيكل البحث الحي المقترح يقترب من أفضل حل مقارنة مع الطرق الأخرى.

المصطلحات الأساسية: جدولة ورشة العمل ، وأوقات الإعداد المعتمدة على التسلسل ، والبحث المحلي ، وهيكل البحث عن الحي.

An Introduction

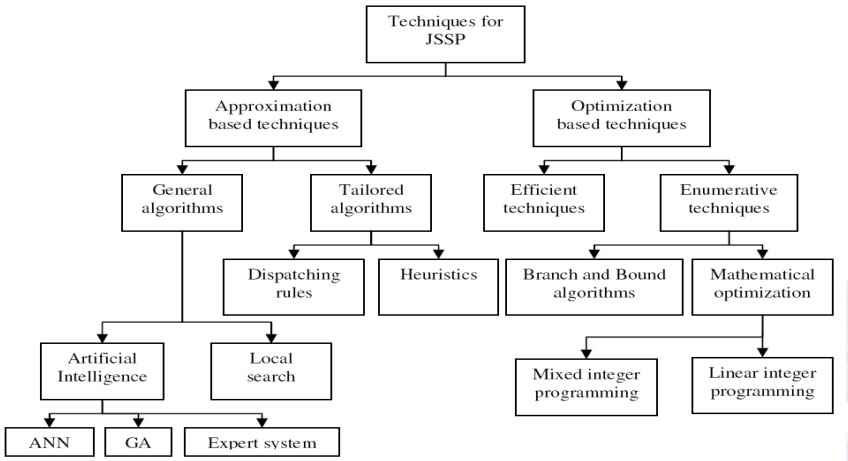
Job-Shop Scheduling Problem (JSSP) is one of the well-known hardest combinatorial optimization problems. JSSP being amongst the worst members of the class of NP-hard problems non-deterministic polynomial [7], there is still a lot of room for improvement in the existing techniques. Improving scheduling systems for greater customer satisfaction and operations efficiency requires an optimization criterion minimizing the makespan. Simultaneously, the sequence dependent setup time environment is a very common scheduling problem in both manufacturing and service organizations. Setup time is defined as the time interval between the end of processing time of the current job and the beginning of processing time of the next job.

Sequence dependent setup times are a tool for modeling a problem where there are different classes of operations which require machines to be reconfigured. For example, two tasks in a machine shop may both be performed on the same drill press, but require different drill bits. In Job Shop Scheduling sequence dependent setup times occur when a machine setup time or cost for a particular job is determined by not only by that job, but also by the previous job that the machine is currently set up for. Information on sequence-dependent setup times or costs for N jobs can be stored in *N*-by-*N* matrix with all diagonal entries being zero. Diagonal entries must be zero since they correspond to the condition that the machine is already setup for the next job; hence the setup time is zero.

Solution Techniques to Handle JSSP

A number of solution techniques to handle the JSSP have been developed over the years. A broad classification of the scheduling techniques is given in [15]. Initially the techniques are divided into two classes as approximation and optimization techniques.

Figure (1): solution approaches for JSSP.



Literature review

[17] decomposed the job shop scheduling problem -with release dates, due dates, and sequence-dependent setup times with the scheduling objective to minimize the weighted sum of squared tardiness- into a series of single-machine scheduling problems within a shifting bottleneck framework. Machines are scheduled one at a time in the order of criticality. The measure of criticality developed is closely related to the job shop scheduling objective and is based on the marginal contribution of a machine to the overall scheduling objective function. The single-machine scheduling problem is solved using a Lagrangian relaxation based approach (LRSSS). The modified shifting bottleneck approach to job shop scheduling with sequence-dependent setups (MSBSS) is compared with the EDD (earliest due date), ATCS (apparent tardiness cost), and SIMSET (similar setup times) dispatching rules. The MSBSS approach results in superior solution quality compared to the results of the other dispatching rules in reasonable computational time.

[3] provided an extensive review of the scheduling literature on models with setup times (costs) from then to date covering more than 300 papers. Scheduling problems are classified into those with batching and non-batching considerations, and with sequence-independent and sequence-dependent setup times, and are categorized according to shop environments, including single machine, parallel machines, flow shop, no-wait flow shop, flexible flow shop, job shop, open shop, and others.

[13] proposed a mathematical model which presents a good performance to obtain feasible solutions. The mathematical model is unable to reach the optimum results in larger problems. Thus, they developed a heuristic model based on priority rules, Because of the inability to find optimum solutions in reasonable computational times, and 3 different innovative lower, which could be implemented to evaluate different heuristics and meta-heuristics in larger problems. The performance of the heuristic model evaluated with a well known example in the literature insures that the model seems to have a strong ability to solve job shop scheduling with sequence dependent setup times problems and to obtain good solutions in reasonable computational times.

[4] investigated scheduling job shop problems with sequence-dependent setup times under minimization of makespan. They developed an effective meta-heuristic, simulated annealing with novel operators, to potentially solve the problem. They proposed an effective neighborhood search structure based on insertion neighborhoods as well as analyzing the behavior of simulated annealing with different types of operators and parameters by the means of Taguchi method. An experiment based on Taillard benchmark is conducted to evaluate the proposed algorithm against some effective algorithms. The results showed that the proposed algorithm outperforms the other algorithms.

[2] proposed a new method for solving job-shop scheduling problem using hybrid Genetic Algorithm (GA) with Simulated Annealing (SA). This method introduces a reasonable combination of local search and global search for solving JSSP.

[11] used simulated annealing algorithm for multi-objective flexible job shop scheduling problem with overlapping in operations to find a suitable solution. To evaluate performance of the algorithm, a mixed integer linear programming Model is developed, and solved it with the classical method (branch and bound). The results showed that in small size problems, the solutions of the proposed algorithm and the mathematical model were so close, and in medium size problems, they only had lower and upper bounds of solution and our proposed algorithm had a suitable solution. Also, they used total machine work loading time and critical machine work loading time as evaluation criteria.

[1] considered flexible job-shop scheduling problem (FJSP) with sequence-dependent setup times to minimize makespan and mean tardiness. Neighborhood structures related to the

sequencing problem and the assignment problem were employed to generate neighboring solutions. To evaluate the performance of the proposed algorithm, 20 test problems in different sizes are randomly generated.

[14] proposed a mixed integer linear programming model which includes process planning and scheduling tasks simultaneously in a flexible assembly job shop with sequence dependent setup times. The objective function is minimizing maximum completion time (makespan) of final products. Also, in order to exploit the maximum flexibility in the shop floor for process planning and scheduling tasks, identical parts have been considered as distinct parts. Finally, several hypothetical test problems have been solved to show the merit of the proposed mathematical model.

[12] considered a dynamic flexible job shop scheduling in which there are sequence-dependent setup times and machines are prone to failure. They proposed three new routing rules, and compared them with another two rules from the literature through simulation experiments.

[8] considered job shop scheduling with sequence-dependent setups in which jobs are grouped into job families, but they are processed individually. This type of job shop scheduling can be found in various manufacturing systems, especially in remanufacturing systems with disassembly, reprocessing and reassembly shops. To minimize the deviations of the job completion times within each job family, the objective of minimizing the total family flow time. A mixed integer programming model is suggested, and then, due to the complexity of the problem, a heuristic algorithm is suggested. Computational experiments were done on small-sized test instances.

[10] presented a new multi-objective job shop scheduling hybrid genetic algorithm (HGA) with sequence-dependent setup times. The objectives are to minimize the makespan and sum of the earliness and tardiness of jobs in a time window. To validate the efficiency of our proposed HGA, a number of test problems are solved.

[5] considered the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times. They transformed the problem into an equivalent network problem, and formulated two mixed integer goal programming models. In the first model (Model A) the sequence dependent setup times are non-separable. In the second one (Model

B) they are separable. Model B is- obtained from Model A with a minor modification. The formulation of the models is described on a small sized numerical example and the solutions are interpreted. Finally, computational results are obtained on test problems.

[9 in this paper addresses the classic job shop scheduling where sequence dependent setup times. Used a tabu search algorithm with a sophisticated neighborhood structure for solve this problem. Based on modified disjunctive graph, he further investigates and generalizes structural properties for the problem under study. In this paper, the solution of the JSSP-SDST using the local search algorithm which is one of the approximation based techniques- will be studied.

Steps of local search algorithm (LS):

1. Encoding schemes.

Encoding schemes are used to make a candidate solution recognizable for algorithms. A proper encoding scheme plays a key role in maintaining the search effectiveness of any algorithms [6]. To evaluate the degree of sensitivity (robustness) of our LS on its initial solution (IS), we bring the choice of IS into the calibration section and compare the performance of two SAs starting from a random generated IS and from any methods for example shortest processing time (SPT).

2. Neighborhood search structure (NSS).

Neighborhood search structure generates a new solution from current candidate solution by making a slight change in it. Five different NSSs are considered:

1. *SWAP* operator in which the random keys (RKs) of two randomly selected operations are swapped. In the algorithm we using the swap neighborhood structure simply consists of exchanging the position of two jobs in the sequence, for example, of a sequence of 5 jobs, namely j1, j2, j3, j4 and j5.slect tow random jobs j4 and j2 then swapped it, new

sequence j1, j4, j3, j2 and j5. Provided that the new solution is feasible, and repeat the previous step until proved the solution when the counting of iterations.

2. *SHIFT* operator in which the RKs of one randomly picked operation and put in first position. We select random a number, then put it in first position. For example, of a sequence of 5 jobs, namely j1, j2, j3, j4 and j5. Select RK from 1 to 5, e.g. 3 the new sequence is j3, j1, j2, j4 and j5.

3. I*NVERSION* operator in which the RKs between two randomly selected cut points are reversed. Select RK between 1 to 5 then cut point and reverses. E.g. Select RK 2 the new sequence is j1, j5, j4, j3 and j2.

4. *Migration Mechanism* (MM) a new farther neighbor is generated from the current solution by relocating two randomly selected operations into two new randomly selected positions (i.e. The RKs of two randomly selected operations are randomly regenerated). Select two random. E.g. 5, 3 then randomly regenerated 2, 1 the new sequence j3, j5, j1, j4 and j2.

5. *The proposed neighborhood search structure* (PNS) is implemented in two steps; the first step is to generate a random number to cut the sequence at a random position. So, the sequence is splinted into a tail part. Then, the tail part is inverted and replaced by the head part. And the head part is replaced at the tail part. The best generated farther neighbor is accepted to move, whether it has the better objective function than the current solution or not. In the example above select random number e.g in the first step select cut point 3, then inverted and replaced by the head part, new sequence is j5, j4, j3, j1 and j2.

The local search algorithm generates an initial solution using shortest processing time method and then neighborhood of this solution and afterward compares objective functions of them with each other: if the new solution is better than the current solution, it substitutes current solution with the new solution and then generates another neighborhood.

Principles of proposed local search algorithm

1. Start with initial configuration (a method to generate an initial configuration using shortest processing time SPT).
2. Repeatedly search the neighborhood and select a neighbor as a candidate (a method to generate an initial configuration).
3. Evaluate some cost function (or fitness function) and accept candidate if "better"; if not, select another neighbor.
4. Stop if quality is sufficiently high, if no improvement can be found or after some fixed time**.**

Figure (2): Solution approach SA algorithm flowchart.

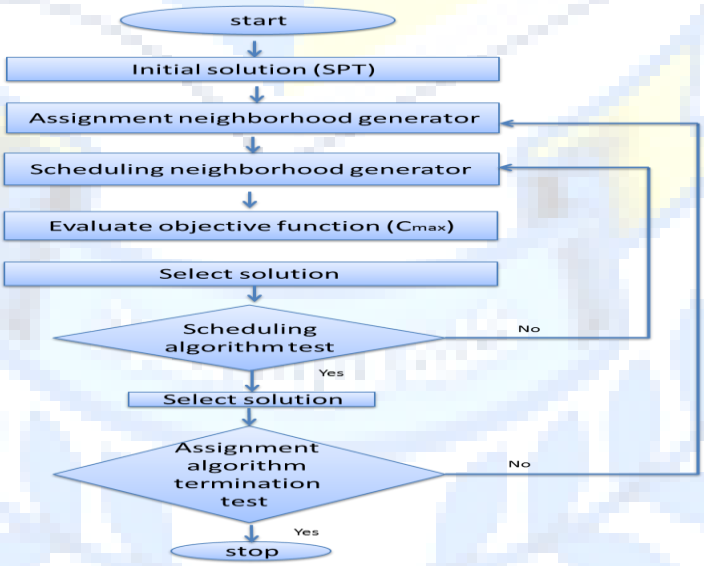


Figure (3): General outline of the proposed local search algorithm.

```
Using C# program.
Procedure local search
X = initialization                % initial solution by SPT
X best = X
// Program Variables:
#region variables
 int m;                          // No. of machines
 int j;                          // No. of Jobs
 int[,] dataSetup;               // Array of Dependent setup times
textBox1.Text += "     " + data Setup [a, b] + "\t";
int[ ] no_opers_job;
int[] no_opers_machine;
int total_operations = 0;
int total_scheduled_operations = 0;
int[][] dataSeqTemp;
 int[][] dataSeqTemp_startTimes;
 int[][] dataSeqTemp_endTimes;
  int makeSpan = 0;
private void ChangeSequence( )= no. of iteration;       // criteria of stopping
      {
            Neighborhood search structure
 //The following steps is repeated, with changing the method of changing sequence
    //SWAP, MM, Inversion, or Shift
     //0: SWAP, 1: MM, 2: INVERSION, 3: SHIFT, 4: Proposed NS

private void SWAP()
      {
         int maxLength = 0;
         for (int i = 0; i < m; i++)
            if (dataSeqCopy[i].Length > maxLength)
               maxLength = dataSeqCopy[i].Length;
Random random = new Random();
         randomNo[0] = random.Next(start, end);
         randomNo[1] = random.Next(start, end);
private void MM( )
      {
         int maxLength = 0;
        for (int i = 0; i < m; i++)
        if (dataSeqCopy[i].Length > maxLength)
      maxLength = dataSeqCopy[i].Length;
    int start = 0, end = maxLength;
randomNo[3] = random.Next(start, end);
 while (randomNo[0] == randomNo[3] || randomNo[2] == randomNo[3]
 randomNo[3] = random.Next(start, end);
  swapTwoColumns(randomNo[0], randomNo[1]);
   swapTwoColumns(randomNo[2], randomNo[3]);
private void SHIFT( )
      {
         int maxLength = 0;
         for (int i = 0; i < m; i++)
            if (dataSeqCopy[i].Length > maxLength)
               maxLength = dataSeqCopy[i].Length;
         int start = 0, end = maxLength - 1;
private int INVERSION( )
      {
         //This function is to
         int maxLength = 0;
         for (int i = 0; i < m; i++)
            if (dataSeqCopy[i].Length > maxLength)

private void ProposedNS()
      {
int[][] newDataSeq = new int[m][];
         for (int i = 0; i < m; i++)
         {
            newDataSeq[i] = new int[dataSeq[i].Length];
         }

         for (int i = 0; i < m; i++)
         {
            int k = 0;
            for (int j = index; j < dataSeq[i].Length; j++)
            {
               newDataSeq[i][k] = dataSeqCopy[i][j]; k = k + 1;}
```

A set of benchmark problems is generated based on Taillard's instances. Data required for a problem consist of the number of jobs (n), number of machines (m), range of processing times (p) and range of the sequence-dependent setup times (SDST). The benchmark contains different combinations of the number of jobs n and the number of machines m. The (n, m) combinations are: ({20, 30 and 50}. (15, 20), (15 x 15) and (100 x 20) summing up 8

combinations of (n x m). The processing times in Taillard's instances [16] are generated from a uniform distribution on (1, 99). A four levels for SDST, 25%، 50%, 100% and 125% of maximum possible processing times in problems. So, SDSTs are generated from four uniform distributions over U (1, 25), U (1, 50), U (1, 100) and U (1, 125) for each subset. The algorithm is coded in C# and run on a PC with 2.50 GHz Intel Core (TM) i5 and 4 GB of RAM.

We will make a test of local search algorithm comparison the make sure the efficiency of the program. That is done by comparing the LS with the best solution from OR- library. We will assume assumptions, as shown below:

1. Each job has its own processing route; that is, jobs visit machines in different orders.
2. Each job might need to be performed only on a fraction of m machines, not all of them.
3. Each job can be processed by at most one machine at a time and each machine can process at most one job at a time. When the process of an operation starts, it cannot be interrupted before the completion; that is, the jobs are non-preemptive.
4. The jobs are independent; that is, there are no precedence constraints among the jobs and they can be operated in any sequence.
5. The jobs are available for their process at time 0.
6. There is no machine breakdown (i.e. Machines are continuously available).
7. The sequences of jobs in machines are dependent setup times, $S_{ij} \neq 0$.
8. The objective function when solving or optimizing a JSS is to determine the processing order of all jobs on each machine that minimizes the makespan.

The number of iterations varies depending on the size of the problem; the greater the size of the problem has increased the number of iterations. In this table, shown how the number of iterations increases with the size of the problems.

Table (1): Number of iterations which give the best solution.

Problems size (nxm)	15x15	20x15	20x20	30x15	30x20	50x15	50x20	100x20
Number of iterations	500	600	800	1000	1200	2500	2500	5000

Figure (4): Number of iterations which give the best solution.

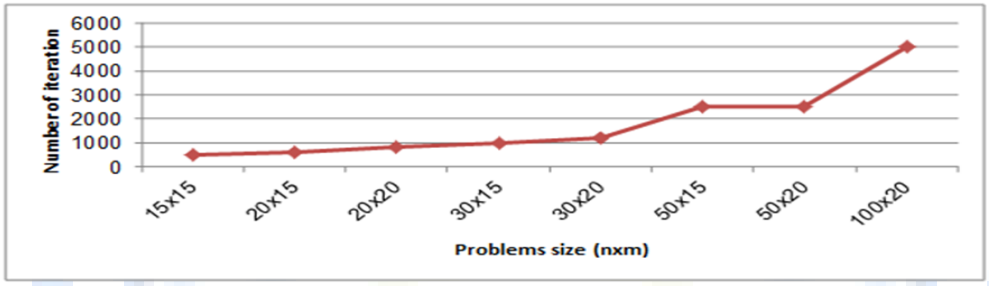
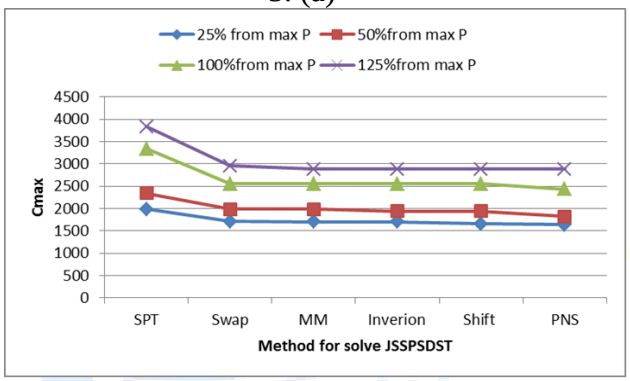


Table (1) Solve JSSP (SDST), with problem sizes (15 x 15, 20 x 15 & 20 x 20, 30 x 1, 30 x 20, 50 x 15, 50 x 20 and 100 x 20).

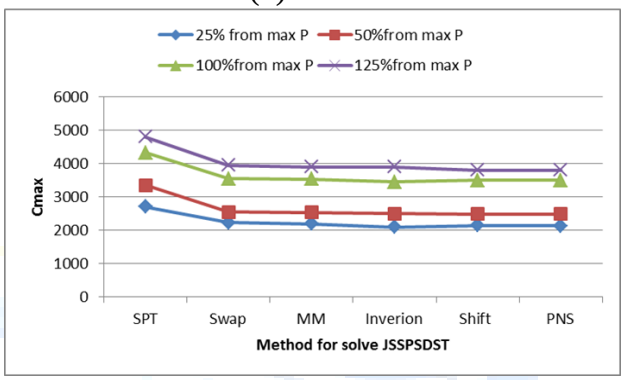
Problems Size (n x m)	SDST from max P	Initial Solution	Local search Algorithm (LSNSS)				
		SPT	Swap	MM	Inversion	Shift	PNS
15x15	25%	1988	1708	1703	1700	1652	1640
	50%	2340	1988	1988	1941	1941	1822
	100%	3339	2556	2556	2556	2556	2429
	125%	3834	2955	2883	2883	2883	2883
20x15	25%	2623	2027	2027	2027	2027	2000
	50%	3112	2303	2303	2302	2302	2247
	100%	3925	3049	3049	3009	2950	2950
	125%	5172	3478	3412	3350	3390	3312
20x20	25%	2707	2233	2200	2100	2150	2138
	50%	3351	2543	2540	2500	2490	2490
	100%	4334	3545	3540	3450	3500	3500
	125%	4807	3951	3899	3899	3810	3808
30x15	25%	3230	2499	2499	2499	2400	2400
	50%	3825	2991	2991	2991	2961	2961
	100%	4715	3880	3880	3880	3799	3799
	125%	5253	4325	4326	4300	4279	4234
30x20	25%	3353	2979	2970	2949	2947	2747
	50%	4195	3473	3473	3473	3440	3440
	100%	5514	4548	4548	4548	4448	4448
	125%	5967	4979	5009	4979	5000	4979
50x15	25%	4715	3940	3940	3940	3800	3800
	50%	5151	4747	4747	4747	4747	4747
	100%	6918	5887	5887	5800	5759	5759
	125%	7972	6922	6922	6920	6900	6816
50x20	25%	5458	4162	4162	4162	4133	4100
	50%	6008	4823	4823	4823	4823	4823
	100%	7830	6484	6484	6484	6484	6400
	125%	9407	7118	7118	7118	7118	7118
100x20	25%	7890	6659	6659	6659	6659	6659
	50%	9832	8114	8114	8114	7997	7902
	100%	12995	11494	11494	11494	11400	11400
	125%	15205	13937	13600	13600	13188	13188

Fig (5): Solve JSSP (SDST), with problem size 15 x 15 (Figure 5.(a)), 20 x 15(Figure 5.(b)), & 20 x 20(Figure 5.(c)).

5. (a)



5. (b)



5. (c)

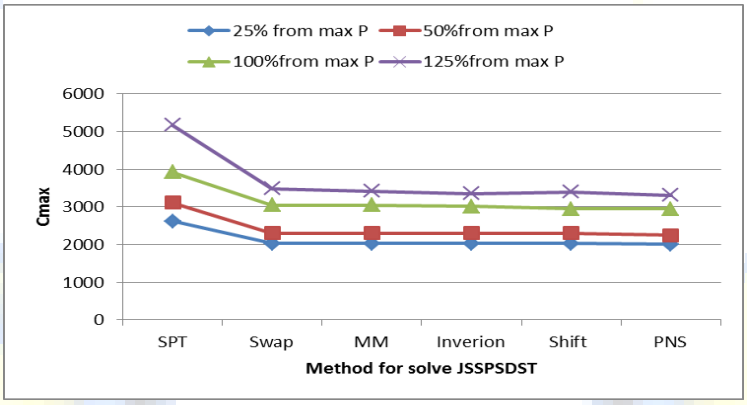
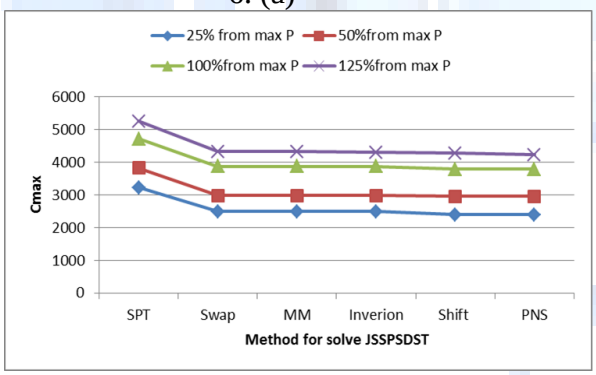


Fig (6): Solve JSSP (SDST), with problem size 30 x 15 (Figure 6.(a)), 30 x 20(Figure 6.(b)).

6. (a)



6.(b)

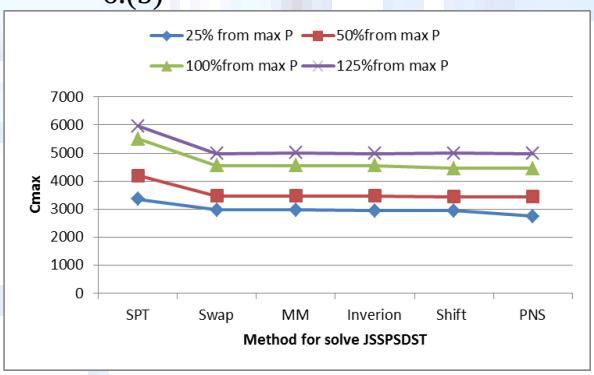
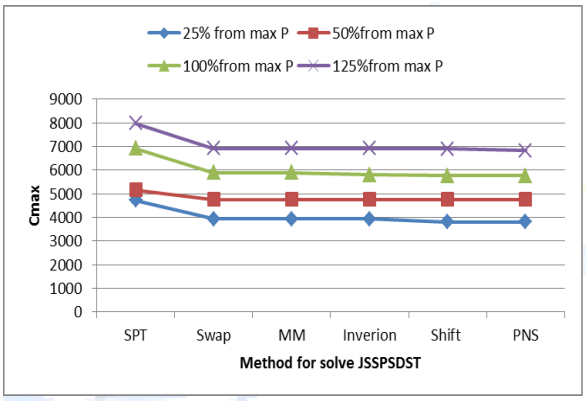


Fig (7): Solve JSSP (SDST), with problem size 50 x 15 (Figure 7.(a)), 50 x 20(Figure 7.(b)).

7.(a)



7.(b)

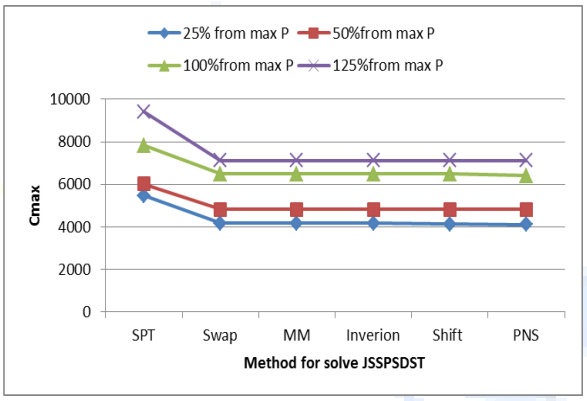
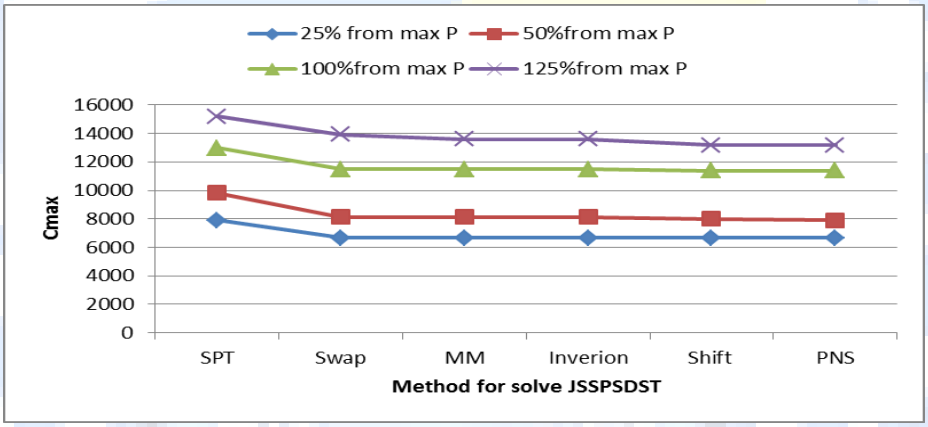


Fig (8): Solve JSSP (SDST), with problem size 100 x 20 (Figure 8).

Figure (8)



7. Conclusion

In this study examined job shop scheduling with the sequence dependent setup times. The optimization criterion is makespan. Considering scheduling problems with setup times is an interest among researchers. Nowadays companies produce multiple goods on common resources and this ends up with the need for setup time activities. Setup time activities usually play a costly role in production processes; therefore planners should consider the principles of setup time activities in their schedules. To tackle the problem we apply meta-heuristic, in the framework of local search, by studying the effect of a novel neighborhood search structure. The numerical experiment included a set of instances generated based on Taillard's

benchmark. We used the four levels from sequence dependent setup times, was a clear difference in values of makespam, this indicates that when increasing the sequence dependent setup times increase the makespan, this is an indication of the impact of setup time in makspan. Were compared four types of neighborhood search structure with proposed NSS, And the results show that the PNS method outperformed the other neighborhood methods taken from the literature.. In this table, we will show the best methods of neighborhood search structure, which are the nearest of the best solution with different problems size.

Table (3): Neighborhood search structure methods neared best solution.

Problems size (n x m)	Neighborhood search structure methods neared best solution
15x15	Proposed neighborhood search structure (PNS).
20x15	Proposed neighborhood search structure (PNS).
20x20	Proposed neighborhood search structure (PNS).
30x15	Proposed neighborhood search structure (PNS) and Inversion.
30x20	Proposed neighborhood search structure (PNS) and Inversion.
50x15	Proposed neighborhood search structure (PNS) and MM.
50x20	Migration Mechanism (MM).
100x20	All methods have same results.

Future work

In future work use the Disjunctive graph mathematical method for solving small and medium problems of job shop scheduling with sequence dependent setup times.

Solve the Flexible job shop scheduling with sequence dependent setup times (FJSSP/SDT) by proposing local search. Solve local search using other methods for initial solution and change objective function.

References

[1]: A.Bagheria, and M. Zandiehb, " **Bi-criteria flexible job-shop scheduling with sequence-dependent setup times—Variable neighborhood search approach**" Journal of Manufacturing Systems 30 (2011) 8–15.

[2]: A.Tamilarasi and T. Anantha kumar "**An enhanced genetic algorithm with simulated annealing for job-shop scheduling**" International Journal of Engineering, Science and Technology, Vol. 2, No. 1, 2010, pp. 144-151.

[3]: Ali Allahverdi, C.T. Ng, T.C.E. Cheng, and Mikhail Y. Kovalyov "**A survey of scheduling problems with setup times or costs**", European Journal of Operational Research 187 (2008) 985–1032.

[4]: B. Naderi, S.M.T. Fatemi Ghomi, and M. Aminnayeri " **A high performing metaheuristic for job shop scheduling with sequence-dependent setup times**", Applied Soft Computing 10 (2010) 703–710.

[5]: Cemal Ozguven, Yasemin Yavuz, and Lale Ozbakır ,"**Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times**", Applied Mathematical Modelling , volume 36 (2012) 846–858.

[6]: Daniel sipper and Robert L. Bulfin "**production planning, control and integration**", McGraw-Hill, 1998.

[7]: Garey, Michael R.; Johnson, David S.and A W. H. Freeman**, "Guide to the Theory of NP-Completeness"**. Computers and Intractability ISBN 0-7167-1045-5 (1979).
[8]:Jung-Hyeon Park, and Dong-Ho Lee " **Job Shop Scheduling with Job Families and Sequence dependent Setups: Minimizing the Total Family Flow Time**" Proceedings of the Asia Pacific Industrial Engineering & Management Systems Conference 2012.

[9]:Liji Shen **"atabu search algorithm for the job shop scheduling problem with sequence dependent setup time"** computer and industrial engineering, reference CAIE 3796. 2 September 2014.

[10]: M. B. Fakhrzad, A. Sadeghieh, and L. Emami "**A New Multi-objective Job Shop Scheduling with Setup Times Using a Hybrid Genetic Algorithm**", IJE TRANSACTIONS B: Applications Vol. 26, No. 2, (February 2013) 207-218.

[11]: Mehrzad Abdi Khalife, Babak Abbasi, and Amir Hossein Kamali Dolat Abadia " **A Simulated Annealing Algorithm for Multi Objective Flexible Job Shop Scheduling with Overlapping in Operations** "Journal of Industrial Engineering, volume 5(2010) 17-28.

[12]: Mohamed A. Shalaby , Tamer F. Abdelmaguid, and Zakaria Y. Abdelrasol "**New Routing Rules for Dynamic Flexible Job Shop Scheduling with Sequence-Dependent**

Setup Times", Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey, July 3 – 6, 2012.

[13]: R. Moghaddas , and M.Houshmand "**Job-Shop Scheduling Problem With Sequence Dependent Setup Times**" Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol II, IMECS 2008, 19-21 March, 2008, Hong Kong.

[14]: Saeid Nourali , Narges Imanipour, and Mohammad Reza Shahriari "**A Mathematical Model for Integrated Process Planning and Scheduling in Flexible Assembly Job Shop Environment with Sequence Dependent Setup Times**", Int. Journal of Math. Analysis, Vol. 6, 2012, no. 43, 2117 – 2132.

[15]: Sultana Parveen* and Hafiz Ullah "**Review on Job-Shop and Flow-Shop Scheduling using Multi Criteria Decision Making**"Journal of Mechanical Engineering, Vol. ME 41, No. 2, December 2010.Transaction of the Mech. Eng. Div., The Institution of Engineers, Bangladesh.

[16]: Taillard," **Benchmarks for basic scheduling problems**", European Journal of Operational Research 64 (1993) 278–285.

[17]: Xiaoqing sun, and James S. Noble "**An approach to job shop scheduling with sequence dependent setups**", Journal of manufacturing systems, volume 18, No; 6, 1999.