Combinatory Least Slack and Kuhn Tucker Optimization for Multiobjective Flexible Job Shop Scheduling

Dr.T.Hemamalini¹, G.Geetha²
1.Assistant professor, Department of mathematics, Govt.Arts college,Coimbatore-18.
2.Research Scholar,Department of mathematics,Govt.Arts college, Coimbatore-18.
ggeethamuralikumar@gmail.com

Abstract. The Multi-objective Flexible Job Shop Scheduling (MFJSS) problem, in which the order or processing sequence for a set of jobs through several machines is assigned in an optimal patterns have accepted sizeable attentiveness. Different types of scheduling rules and procedures for certain types of MFJSS have made progress from these endeavors. The scheduling problem is cumbersome to be made consistent due to different types of criteria involved. This is because in MFJSS, after the completion of a job on a machine, it may be transited between different machines and transit time may affect scheduling. However, the transit times are frequently ignored in the literature. In this paper, the transit time and the processing time are considered as the independent time into the MFJSS. In this paper, we propose Combinatory Least Slack and Kuhn Tucker Optimization (CLS-KTO) for the MFJSS problem with the objectives to minimize the mean tardiness, makespan and job completion time. The problem is addressed via assignment and scheduling model. First, by using well-designed processing and transit time and least slack variable, Combinatory Least Slack (CLS) algorithm is adapted for the CLS-KTO. Then, an Additive function is formulated by incorporating Combinatory Dispatching Rules into the adapted CLS, where some good machines are assigned for the corresponding job, therefore reducing mean tardiness. Furthermore, in the proposed sequencing part, an optimization model based on the Kuhn Tucker conditions and Lagrange Multiplier is adopted to handle the three objectives. In the experimental studies, the influence of mean tardiness on the performance of the proposed CLS-KTO is first examined. Afterwards, the effectiveness of makespan and job completion time in CLS-KTO is verified. Finally, extensive comparisons are carried out with the state-of-the-art methods for the MFJSS on benchmark OR instances. The results show that CLS-KTO performs better than other algorithms.

Keywords: Combinatory, Least Slack, Machine Assignment, Transit Time, Processing Time,Kuhn Tucker Optimization, Lagrange Multiplier

1. Introduction

Production scheduling optimization results in notable developments in manufacturing. Flexible Job Shop Scheduling (FJSS) is one of the most salient and laborious issues in scheduling. Most of the algorithms applied in dealing with FJSS are classified as meta-heuristic models. Certain models are said to be laborious and time consuming while certain other models are found to be more complicated in nature.
A Modified Iterated Greedy (MIG) was designed in [1] to deal with FJSS problem. The purpose of using this algorithm was to provide a simpler mechanism using meta-heuristic. This was found to be comparatively easy in coding. This was achieved by segregating the conventional IG into two different steps. Each step was utilized to address the problem, i.e., sequencing and routing. Besides, a set of dispatching rules were also used for both sequencing and machine selection during the construction phase. With this, the consumption of CPU time was found to be less, with better makespan. Despite improvement observed by applying the greedy algorithm, only a single objective, i.e., minimizing of makespan or maximal completion of all jobs was said to be achieved. However, with multitudes of objective said to be exist, the mean tardiness also increases. To address this issue, in this work, Combinatory Least Slack Machine Assignment (CLS-MA) model is proposed that even with the presence of multi-objective function reduces the mean tardiness using additive function.

An Energy-efficient Dynamic Flexible Flow-shop Scheduling (EDFFS) was presented in [2] that took into consideration the peak power value along with new arrival jobs. In addition, a priority based hybrid parallel Genetic Algorithm (GA) was designed using predictive reactive complete rescheduling model. Besides with the objective of achieving speedup to solve the short response in case of dynamic environment, the proposed method was also found to be highly consistent. Experiments conducted evident that it not only solved the issue in flexible manner, but also minimized the time requirements in a significant manner. However, with the application of hybrid parallel GA, a tradeoff between solutions quality (i.e., in terms of makespan) and time consumption were said to be attained. This is because of the rescheduling once the makespan was less, the time consumption was high and vice versa. To address this issue of exploitation and exploration, an optimization model is proposed that tends to reduce both the time and the makespan for MFJSS problem.

In this paper, we present how to resolve the multi-objective flexible job shop scheduling issue by a hybridization of two parts, i.e., machine assignment part and sequencing part within an optimization model. This new method follows two principal steps. In the first step, a Combinatory Least Slack is applied for a global exploration of the machine assignment. Then, in the second step, a local search is used by a set of Kuhn Tucker conditions along with the Lagrange Multiplier to obtain a viable schedule, therefore optimizing multi-objective function. Numerical tests were made to evaluate the performance of our method based on Benchmark OR Library [3] for the MFJSS, where the experimental results show its efficiency in comparison with other methods.

1.1 Structure of this paper

The remainder of this paper is organized as follows.

- In Section 2, a brief review of relevant works, in conjunction with flexible job shop scheduling problems and multi-objective flexible job shop scheduling problems is presented. We then present our main contributions of this work.
- In Section 3, we formulate a two-step model involving multi-objective model with minimum makespan, mean tardiness and job completion time for MFJSS problem. Besides in section 3 the Combinatory Least Slack (CLS) algorithm is proposed to solve the machine assignment and Combinatory Least Slack algorithm for job sequencing.
- In Section 4 experimental setup with detailed parameter analysis is presented.
- In Section 5, we summarize our work in this paper.

2. Related works

Flexible job shop scheduling is to properly allocate the resources operations to optimize tardiness, execution time, latency and so on. However, in recent years, ultimatum and outlay for energy have progressed to increase. A novel rescheduling mechanism was designed in [4] with the objective to minimize the makespan and energy consumption. Yet another new algorithm was
presented in [5] based on Gravitational Search Algorithm (GSA). Here, the FJSSS was designed by means of scheduled job was designed using GSA algorithm, that escaped from local convergence and therefore resulting in the enhancement of identifying better solution.

A non permutation flow shop scheduling based on single-machine-based adjustment procedures was proposed in [6], followed by which a novel two-machine-based model was also introduced. Based on these two models, lower and upper bounds were designed, ensuring smooth scheduling. Yet another flexible job shop scheduling for flight deck based on Mixed Integer Linear Programming (MILP) formulation was designed in [7]. However, with local convergence, computational efficiency was not said to be attained. An improved Differential Evolution (DE) algorithm was also integrated with typical local search strategies to enhance the computational efficiency. Two different types of programming models based on mixed integer programming (MIP) and constraint programming (CP) was proposed in [8] with the objective of providing high quality solutions with optimal time. However, with the increase in the idle time limit, the solution quality was said to be compromised. To address this issue, in [9], a two-level meta heuristic algorithm was designed, called, as the lower level algorithm and higher level algorithm. On one hand, the upper level algorithm also referred to as the population-based algorithm was designed in such a manner to be a parameter controller for lower-level algorithm, whereas the lower level also referred to as the local search algorithm searched for an optimal schedule in the solution space. This in turn contributed towards acceptable idle time limit.

A bi-objective mathematical model was designed in [10] that considered overlapping (i.e. incurring an extra processing cost) by means of Epsilon constraint method. This in turn not only minimized the makespan but also reduced the computing time in a significant manner. Yet another bi-objective model was designed in [11] by means of logic inequalities, therefore identifying a suitable trade-off between runtime and solution quality.

In [12], a large-scale stochastic job shop scheduling problem was investigated involving similar jobs, where the processing times of the same step were drawn in an independent manner from a priori probability distribution, with the objective remaining in reducing the makespan. A stochastic job shop scheduling model was designed by introducing a deterministic model along with the policy for tracking. With this, the model indicated that the policy was found to be asymptotically optimal. Yet another integrated approach involving teaching and learning based model was designed in [13] via routing and sequencing.

In the recent few years, workshop scheduling has specifically concentrated on the performances of production efficiency, involving time and quality, etc. However, certain other factors involving environmental metrics have attracted the attention of several researchers. In [14], an energy-efficient job shop scheduling problem was presented by means of Grey Wolf Optimization algorithm with Double-searching Mode to reduce both the energy-consumption cost and tardiness.

A permutation flow shop problem is considered as the most complex combinatorial optimization problem. Over the last few years, different types of algorithms are presented to solve the static permutation flow shop issues. However, with highly dynamic involvement of orders, a memetic algorithm-based rescheduling method was designed in [15] concerning both single and multiple orders besides including random interruptions of resources. Besides, a two stage teaching learning optimization method under machine breakdown was presented in [16]. Here, a non idle time insertion technique was designed in addition to the rescheduling technique, therefore contributing towards robust and predictive stable schedules. Though multi-objective functions were arrived at, but with higher premature convergence, optimization was not said to be highly feasible for the solution space. In [17], a Bee Evolutionary Guiding Non-dominated Sorting Genetic Algorithm II (BEG-NSGA-II) was designed with the objectives of both minimizing the maximal completion time and the workload of the most loaded machine. A two stage optimization mechanism was designed where, NSGA-II
algorithm was used to attain initial population \( N \) using bee evolutionary algorithm. Next, Pareto-optimal solutions were obtained, therefore improving the searching ability and avoid premature convergence. Yet another Multi Adaptive Genetic Algorithm was proposed in [18] to improve rationality. A task scheduling model integrating Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) was designed in [19], therefore improving the operation efficiency. A multi-objective schedule optimization for road network was investigated in [20].

2.1 Contribution of this paper

Based on the above literature overview, we conclude that mainly single objective models have been considered for solving flexible job shop scheduling. With respect to mathematical models, some researches introduce greedy and parallel genetic algorithm formulations. However, multi-objective constraints were not attained. Besides, seen with a trade-off between makespan and time, optimal scheduling is not said to be arrived at. Recently, combinatorial optimization models have shown to execute exceptionally well for figuring out multi-objective scheduling problems. In this paper, we therefore introduce two models, a Combinatory Least Slack Machine Assignment (CLS-MA) model and a Kuhn Tucker Multi-objective Optimization model for MFJSS problem that aims to minimize makespan, mean tardiness and job completion time. We evaluate the performance of both models by using Combinatory Least Slack and Kuhn Tucker Optimization (CLS-KTO). Furthermore, we compare the performance of CLS-KTO with a recently proposed MIG [1] and EDFFS [2].

3. Combinatory Least Slack and Kuhn Tucker Optimization (CLS-KTO) for the MFJSS

In this section, we present the details of the proposed CLS-KTO method, including the problem definition, machine assignment part that allocates each operation to a machine from a set of accessible machines and the sequencing part that sequences all the assigned operations on the machines so as to obtain a viable schedule to optimize the multi-objective function.

CLS-KTO method involves two steps. In the first step, machine assignment is performed based on processing and transit time. Followed by which a least slack rule is applied and accordingly the machine assignment is said to take place. Next, in the second step, job sequencing is said to be accomplished by means of introducing a complimentary slack variable along with the Kuhn Tucker operations and Continual Level Foraging (CLF). The elaborate description of the proposed CLS-KTO method is given below.

3.1 Problem definition

The MFJSS problem is formulated as follows. Let us consider a set of ‘\( n \)’ jobs ‘\( J = J_i = \{J_{i1}, J_{i2}, ..., J_{im}\}, \) where \( i = 1, 2, ..., n \) ‘ and a set of ‘\( m \)’ machines ‘\( M = M_k = \{M_{k1}, M_{k2}, ..., M_{kn}\}, \) where \( k = 1, 2, ..., m \)’. Each job ‘\( J_i \)’ comprises of a pre-arranged sequence of operations ‘\( J_i = O_{i1}, O_{i2}, ..., O_{ini} \)’ to be performed one after another, where ‘\( ni \)’ denotes total number of operations that job ‘\( J_i \)’ has to be accomplished. Besides, each operation ‘\( O_{ij} \)’ is said to be performed or processed by any machine ‘\( M_{ij} \subset M \)’ and ‘\( P_{ijk} \)’ represents the processing time of the ‘\( jth \)’ operation for job ‘\( J_i \)’ that is processed by machine ‘\( k \)’ respectively, which is considered as the machine assignment part.

Here, ‘\( T_{sitTime_{ij}} \)’ represents the transit time of job ‘\( J_i \)’ between machines ‘\( M_i \)’ and ‘\( M_j \)’ respectively and ‘\( C_i \)’ represent the completion time of job ‘\( i \)’. Along with machine assignment, the order of machines is determined which is considered as scheduling part based on multi-objective that lessen the makespan (i.e., minimize maximal completion time), total workload and critical workload. In this work with MFJSS as the design model, three objectives are mathematically expressed as given below.

\[
C_{max} = \min_{1 \leq k \leq n} \{C_k\} \quad (1)
\]

\[
W_f = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} P_{ijk} + T_{sitTime_{ijk}} \quad (2)
\]

\[
W_M = \left[ \max_{1 \leq k \leq n} \left( \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} P_{ijk} \right) \right] \quad (3)
\]
From the above equation (1) \( C_{\text{max}} \) represents the minimization of maximal completion time of ‘k’ machines, equation (2) \( W_T \), the total workload is represented by the summation of processing time \( P_{ijk} \) and the transit time \( T_{\text{sit}} \), corresponding to the operation \( O_{ij} \) on the machine \( M_k \) respectively. Finally, the equation (3) represents the critical workload \( W_{\text{cr}} \) referring to the highest workload between machines. The main objective of this problem is to find a schedule minimizing the makespan, mean tardiness and job completion time.

### 3.2 Combinatory Least Slack Machine Assignment (CLS-MA) model

With the assumption of three objectives, in this section, machine assignment is performed by applying Combinatory Least Slack Machine Assignment (CLS-MA) model. To explain the MFJSS with CLS-MA model, a sample problem involving three jobs and five machines is provided in table 1. The numbers here denotes the processing time whereas ‘−’ represent that the operation is not said to be routed on the subsequent machine.

**Table 1 Processing time for 3 Job 5 Machine instance**

| Job | Operation | \( M_1 \) | \( M_2 \) | \( M_3 \) | \( M_4 \) | \( M_5 \) |
|-----|-----------|-----------|-----------|-----------|-----------|-----------|
| \( j_1 \) | \( O_{11} \) | 2         | 9         | 4         | 5         | 1         |
| \( j_1 \) | \( O_{12} \) | −         | 6         | −         | 4         | −         |
| \( j_2 \) | \( O_{21} \) | 1         | −         | 5         | −         | 6         |
| \( j_2 \) | \( O_{22} \) | 3         | 8         | 6         | −         | −         |
| \( j_2 \) | \( O_{23} \) | −         | 5         | 9         | 3         | 9         |
| \( j_3 \) | \( O_{31} \) | −         | 6         | 6         | −         | −         |
| \( j_3 \) | \( O_{32} \) | 3         | −         | −         | 5         | 4         |

As given in the above machine assignment, according to the minimum processing time, the machines are being allocated. But if a machine has already being allocated to the previous operation, and is in the busy state, then, the machine with next minimum processing time is being allocated and so on.

As in the above example, for the first job ‘\( j_1 \)’ with operation ‘\( O_{11} \)’, ‘\( M_5 \)’ is being allocated as it requires minimum process time ‘1 sec’, next, machine ‘\( M_4 \)’ is being allocated, for the first job ‘\( j_1 \)’ with operation ‘\( O_{12} \)’. However, in case of the second job ‘\( j_2 \)’, as already machine ‘\( M_1 \)’ is busy during the second job ‘\( j_2 \)’ with operation ‘\( O_{22} \)’, machine ‘\( M_3 \)’ is assigned instead of ‘\( M_1 \)’. Besides, in our work, as transit time is also involved between different machines and a sample is provided in table 2, where transit time involved between five different machines are shown.

**Table 2 Transit time between different machines**

| Machines | \( M_1 \) | \( M_2 \) | \( M_3 \) | \( M_4 \) | \( M_5 \) |
|----------|-----------|-----------|-----------|-----------|-----------|
| \( M_1 \) | 0         | 2         | 3         | 2         | 4         |
| \( M_2 \) | 2         | 0         | 3         | 4         | 3         |
| \( M_3 \) | 3         | 3         | 0         | 5         | 6         |
| \( M_4 \) | 2         | 4         | 5         | 6         | 3         |
| \( M_5 \) | 4         | 3         | 2         | 1         | 0         |

From the above table 2, the transit time, between ‘\( M_1 \& M_1is 0 \)’, transit time between ‘\( M_1 \& M_2is 2 \)’, transit time between ‘\( M_1 \& M_3is 3 \)’, transit time between ‘\( M_1 \& M_4is 2 \)’ and transit time between ‘\( M_1 \& M_5is 4 \)’. In a similar manner, the transit time, between ‘\( M_2 \& M_1is 2 \)’, transit time between
With the above consideration of processing and transit time, combinatory dispatching rules using additive function based on Least Slack Rule is designed. In other words, the simple rule is to formulate additive function that involves the sum of processing time and total jobs in the queue. The pseudo code representation of Combinatory Least Slack is given below.

Let us consider that there are three jobs, \(J_1\), \(J_3\) and \(J_4\), waiting in the queue at machine \(M_1\). Let us further assume that jobs \(J_1\) and \(J_4\) go to machine \(M_3\) for the next operation and that job \(J_3\) goes next to machine \(M_2\). Let the total jobs currently in the queue at machine \(M_2\) be \(TQ_2\) and the job currently in the queue at machine \(M_3\) be \(TQ_3\). Let us further assume that the processing and transit time of three jobs, \(J_1\), \(J_3\) and \(J_4\) on \(M_1\) be \(P_{11}\), \(P_{31}\), \(P_{41}\), \(TslitTime_{11}\), \(TslitTime_{31}\) and \(TslitTime_{41}\). Suppose \(M_1\) becomes free and we need to choose a job, then, in our work, additive function is used and is formulated as given below.

\[
Z_1 = P_{11} + TslitTime_{11} + TQ_3 \quad (4)
\]
\[
Z_3 = P_{31} + TslitTime_{31} + TQ_2 \quad (5)
\]
\[
Z_4 = P_{41} + TslitTime_{41} + TQ_3 \quad (6)
\]

From the above equation (4), for job 1, because job 1 goes to machine 3 for its next operation with minimum transit time 0, from equation (5), for job 3, because job 3 goes to machine 2 for its next operation with minimum transit time 3 and finally from equation (6), for job 4, because job 4 goes to machine 3 for its next operation with minimum transit time 2 and so on. The mathematical equation for additive function is then expressed as given below.

\[
Z_i = P_{ij} + TslitTime_{ij} + TQ_k \quad (7)
\]

From the above equation (7), the additive function \(Z_i\) is expressed as the summation of processing time \(P_{ij}\), with transit time being \(TslitTime_{ij}\) for the corresponding job in the queue \(TQ_k\) respectively. Finally, according to our proposed Combinatory Dispatching Rules i.e., Least Slack Rule is mathematically written as given below.

\[
s = (d - t) * Z_i \quad (8)
\]

From the above equation (8), the slack time for a machine \(s\) is expressed by means of machine deadline \(d\), real time since the machine assignment starts \(t\) and the additive function \(Z_i\).

The pseudo code representation of Combinatory Least Slack is given below.

**Input:** Jobs \(J_i = \{J_1, J_2, \ldots, J_n\}\), Machine \(M = M_k = \{M_1, M_2, \ldots, M_m\}\), processing time \(P_{ijk}\), Transit time \(TslitTime_{ij}\), completion time \(C_i\)

**Output:** Machine Assigned with Least Slack \(M_{LS}\)

1: Begin
2: For each Jobs \(J\) with machine \(M\) with transit time \(TslitTime_{ij}\) and completion time \(C_i\)
3: Evaluate maximal completion time
using (1)
4: Evaluate total workload using (2)
5: Evaluate critical workload using (3)
6: Measure additive function using (7)
7: Evaluate least slack rule using (8)
8: Return (assign machine to corresponding job with least slack value)
9: End for
10: End

Algorithm 1 Combinatory Least Slack

As given in the above algorithm, with benchmark OR-Library obtained as input, first, the multi-objective is defined. Next, an additive function is introduced as a combinatory rule, including, the processing time, transit time and the total job in queue. Followed by which, the least slack rule is evaluated. Finally, the machine is assigned to the corresponding job with least slack value. With this, an assignment of machine for processing all jobs with minimum mean tardiness is said to be achieved.

### 3.3 Kuhn Tucker Optimized Lagrange Scheduling

In this section, a sequencing model using Kuhn Tucker Multi-objective Optimization is proposed that sequences all assigned operations on machines so as to obtain a viable schedule to optimize the multi-objective function. A Kuhn-Tucker Optimized Lagrange Scheduling model is used in this work with the objective of solving combinatorial optimization. Here, with the machine assigned to corresponding job with least slack, local search is performed for optimized scheduling (i.e. minimizing both makespan and job completion time) using Kuhn Tucker conditions. Besides, in this work, a complementary slackness is used to obtain an optimal solution to the dual (minimizing makespan and job completion time) problem.

The basic mathematical programming problem as described in the above section is that of selecting values (i.e., machines) of ‘n’ variables so as to minimize a function (in such a way that jobs are assigned to specific machine) of those variables subject to ‘m’ inequality constraints. This is mathematically expressed as given below.

\[
\text{Min } f_0(M_{LS}) \quad (9)
\]

**Subject to**

\[
f_i(M_{LS}) \leq 0 \quad (i = 1,2,...,m) \quad (10)
\]

To the above conventional optimization model, an additional complimentary slack variable ‘CS’ is used to obtain an optimal solution to the dual problem. This is rewritten as given below.

\[
\text{Min } f_0(M_{LS}) \quad (11) \text{ Subject to } f_i(M_{LS}) + CS_i^2 \leq 0 \quad (i = 1,2,..,m)
\]

With the above complimentary slack variable, a solution area for MFJSS and to find optimum scheduling is said to be attained. Under the assumption of constraint qualifications (i.e. to obtain optimal solution to dual problem), the Lagrange theory with an additional complimentary slack is extended as given below. If the function ‘\(f_0(M_{LS})\)’ attains at point ‘\(M_{LS}^0\)’ a local minimum subject to the set ‘\(K = \{M_{LS} | f_i(M_{LS}) \leq 0 (i = 1,2,...,m)\}\)’ then there exists a vector of Lagrange multipliers ‘\(LM^0\)’ such that the following conditions are satisfied:

\[
\frac{\partial f_0(M_{LS}^0)}{\partial (M_{LS})} + \sum_{i=1}^{m} LM_i \frac{\partial f_i(M_{LS}^0)}{\partial (M_{LS})} = 0, (j = 1,2,...,n) \quad (13)
\]

\[
f_i(M_{LS}^0) \leq 0, (i = 1,2,...,n) \quad (14)
\]
From the above equations (13), (14), (15) and (16), local minimum is said to be attained, that are called the Kuhn-Tucker conditions. Here, a Continual Level Foraging (CLF) principle is applied that involves two steps. First, a sub-set of scheduling decisions is retracted in a random manner from the current solution using (13) and (14). Then, in the second step, a new solution is gradually measured using (15) and (16) via partial solution (i.e. obtained through (13) and (14)). The pseudo code representation of Kuhn-Tucker Lagrange Optimized Scheduling is given below.

\[
LM^0_i(M^0_{1S}) = 0, (i = 1, 2, ..., n)
\]
\[
LM^0_i \geq 0, (i = 1, 2, ..., n)
\]

From the above equations (13), (14), (15) and (16), local minimum is said to be attained, that are called the Kuhn-Tucker conditions. Here, a Continual Level Foraging (CLF) principle is applied that involves two steps. First, a sub-set of scheduling decisions is retracted in a random manner from the current solution using (13) and (14). Then, in the second step, a new solution is gradually measured using (15) and (16) via partial solution (i.e. obtained through (13) and (14)). The pseudo code representation of Kuhn-Tucker Lagrange Optimized Scheduling is given below.

| Input: |
| Machine Assigned with Least Slack ‘\( M_{1S} \)’ |
| Output: |
| Multi-objective optimized scheduling |

1: Initialize complimentary slack variable ‘CS’, lagrange multipliers ‘LM’
2: Begin
3: For each Machine Assigned with Least Slack ‘\( M_{1S} \)’
4: Obtain mathematical programming model by minimizing as in (9) subject to constraints as in (10)
5: Modify optimal solution according to dual problem by minimizing as in (11) subject to constraints as in (12)
6: Obtain Kuhn-tucker conditions by introducing lagrange multiplier
7: Return()
8: End for
9: End

Algorithm 2 Kuhn-Tucker Lagrange Optimized Scheduling

As given in the above Kuhn-Tucker Lagrange Optimized Scheduling algorithm, first, complimentary slack variable and lagrange multiplier is initialized. Then, with the machine assigned with the least slack value being selected, a mathematical model is selected based on inequality constraints with multiple objectives. Next, based on duality principle, the mathematical model is reformulated according to the complimentary slack. Finally, based on the Kuhn tucker conditions, optimized scheduling is arrived at by means of CLF principle, therefore contributing to makespan and job completion time.

4. Experimental setup

The proposed algorithms and method are implemented in matlab. The algorithm and method is tested on benchmark OR-Library. This dataset cover almost all the problem instances ever adopted in the literature on MFJSS with which a comprehensive evaluation of all the implemented algorithms is made. Table 3 lists the parameter settings of CLS-KTO. Uniform parameter values have been used for the algorithms.

Table 3 Parameter settings of CLS-KTO for the MFJSS

| S. No | Parameter         | Value |
|-------|-------------------|-------|
| 1     | Number of jobs    | 150   |
4.1 Performance metrics

In order to evaluate the performance of the CLS-KTO method for MFJSS, mean tardiness, makespan and job completion time are used as indicators in our experiments. They are expressed as follows.

4.1.1 Mean tardiness

To investigate the impact of mean tardiness, on the proposed CLS-KTO method, the performance comparison between MIG [1] and EDFFS [2] is carried out in this section. In MFJSS, tardiness refers to measure of delay in executing certain operations. In MFJSS with multiple jobs let the deadline be ‘\( D_i \)’ and the completion time be ‘\( C_i \)’ of job ‘\( i \)’. Then, the mean tardiness is expressed as below.

\[
MT = \sum_{i=1}^{n} j_i \times [D_i - C_i] \tag{17}
\]

From the above equation (17), the mean tardiness ‘\( MT \)’ is measured by means of the number of jobs ‘\( j_i \)’ and the difference between the deadline ‘\( D_i \)’ and the completion time ‘\( C_i \)’ respectively. It is measured in terms of milliseconds (ms). In table 4, the mean tardiness for 150 different jobs carried over 10 simulation runs is presented.

| Number of jobs | Mean tardiness (ms) |
|---------------|---------------------|
|               | CLS-KTO  | MIG     | EDFFS   |
| 15            | 120      | 195     | 270     |
| 30            | 135      | 215     | 295     |
| 45            | 155      | 230     | 315     |
| 60            | 195      | 245     | 335     |
| 75            | 200      | 285     | 350     |
| 90            | 210      | 310     | 415     |
| 105           | 220      | 325     | 425     |
| 120           | 245      | 345     | 440     |
| 135           | 310      | 380     | 465     |
| 150           | 315      | 410     | 480     |

From table 4, the mean tardiness of CLS-KTO method is significantly better than [1] and [2].

Figure 1 Graphical representation of mean tardiness
Figure 1 given above illustrates the performance measure of mean tardiness with respect to 150 different jobs selected at different time intervals. With the increase in the jobs, the mean tardiness i.e., amount of delay in executing certain operations also increases. This is because certain operations are said to be dependent whereas certain other operations are found to be independent. Therefore, the increase in mean tardiness is not also found to be proportionally increasing. However, with the simulations conducted with ‘15’ jobs and deadline being assumed to be ‘350ms’, the completion time for CLS-KTO was found to be ‘358ms’, ‘363ms’ using [1] and ‘368ms’ using [2].

With this, the mean tardiness was observed to be ‘120ms’, ‘195ms’ and ‘270ms’ using CLS-KTO, [1] and [2] respectively. From this performance evaluation, it is inferred that the mean tardiness using CLS-KTO is found to be minimum when compared to [1] and [2]. This is because of the incorporation of Combinatory Least Slack Machine Assignment (CLS-MA) model. By applying this model, in addition to the processing time, a transit time is also considered as the independent time into the MFJSS. With these two time, an additive function is employed in CLS-KTO that in turn assists in reducing the mean tardiness by 29% compared to [1] and 45% compared to [2].

4.1.2 Makespan

Makespan is defined as the time difference between start and end time of Multi-objective Flexible Job Shop Scheduling (MFJSS). In other words, it is defined as the amount of time from start to finish for completing a set of jobs. This is mathematically expressed as given below.

\[ MS = \sum_{i=1}^{n} J_i \times [ST - FT] \]  

From the above equation (18), the makespan ‘MS’ is measured on the basis of the number of jobs to be assigned to specific set of machines and the time difference between the start time ‘ST’ and finish time ‘FT’. It is measured in terms of milliseconds (ms).

| Table 5 Performance evaluation of the makespan using CLS-KTO, MIG [1] and EDFFS [2] |
|-------------------------------------------|
| Number of jobs | CLS-KTO | MIG | EDFFS |
| 15            | 120     | 180 | 225   |
| 30            | 130     | 205 | 240   |
| 45            | 180     | 225 | 275   |
| 60            | 215     | 240 | 315   |
| 75            | 225     | 280 | 335   |
| 90            | 240     | 330 | 385   |
| 105           | 275     | 345 | 425   |
| 120           | 315     | 410 | 485   |
| 135           | 325     | 455 | 510   |
| 150           | 355     | 490 | 530   |

Table 5 given above illustrates the makespan observed for 150 different numbers of jobs using the CLS-KTO method, MIG [1] and EDFFS [2].
Figure 2: Graphical representation of makespan

Figure 2 given above shows the performance graph of makespan. With 150 different numbers of jobs considered for simulation, makespan value is recorded, that measures the difference between start and end time of scheduling. Lower makespan value ensures the efficiency of the method. From the simulations conducted with 15 number of jobs, time difference between the start and finish time of allocating the jobs to the corresponding machines using CLS-KTO was observed to be ‘8ms’, ‘12ms’ with MIG [1] and ‘15ms’ with EDFFS [2]. With this, the makespan using CLS-KTO was found to be ‘120ms’, ‘180ms’ and ‘225ms’ using [1] and [2]. From the simulation it is found that the makespan was minimal when applied with CLS-KTO when compared to [1] and [2]. This is because of the application of Kuhn-Tucker Optimized Lagrange Scheduling model. By applying this model, combinatorial optimization, like satisfying the multi-objective was said to be solved. Besides, by introducing a complimentary slack variable to the combinatorial optimization, optimality to the dual problem was said to be arrived with early convergence. Hence, the makespan using CLS-KTO was found to be reduced by 25% when compared to [1] and 37% when compared to [2].

4.1.3 Job completion

The job completion refers to the time consumed to accomplish a job. It is measured as given below.

\[ JCT = \sum_{i=1}^{n} J_i \times (PT_{ij} + T_{sitime_{ij}}) \] (19)

From the above equation (19), job completion time ‘JCT’ is measured according to the processing ‘PT_{ij}’ and transit time ‘T_{sitime_{ij}}’ involved in assigning the jobs ‘J_i’ to corresponding machines. It is measured in milliseconds (ms).

Table 6: Performance evaluation of job completion time using CLS-KTO, MIG [1] and EDFFS [2]

| Number of jobs | Job completion time (ms) |
|----------------|--------------------------|
|                | CLS-KTO     | MIG    | EDFFS |
| 15             | 165         | 225    | 255   |
| 30             | 190         | 250    | 290   |
| 45             | 230         | 280    | 350   |
| 60             | 280         | 320    | 410   |
| 75             | 320         | 390    | 450   |
| 90             | 350         | 410    | 520   |
| 105            | 360         | 450    | 580   |
| 120            | 410         | 480    | 640   |
| 135            | 430         | 530    | 690   |
| 150            | 510         | 550    | 730   |

Table 6 given above illustrates the measure of job completion time arrived at for 150 different numbers of jobs to be scheduled at different time intervals with multi-objective as a means.
Figure 3 given above illustrates the job completion time. With x axis representing the number of jobs in the range of 15 to 150, the job completion time measured was said to be in the range of 100 to 800ms. Increasing the number of jobs obviously resulted in the increase in processing time and transit time and therefore increases in the job completion time. On the other hand, minimum the number of jobs, minimum is the job completion time and vice versa. Simulations conducted with 15 numbers of jobs using all the three methods observed ‘9ms’ and ‘2ms’ as processing and transit time using CLS-KTO, ‘11ms’ and ‘4ms’ as processing and transit time using [1] are ‘12ms’ and ‘5ms’ as processing and transit time using [2]. To summarize, the job completion time was found to be ‘165ms’ using CLS-KTO and ‘225ms’ and ‘255ms’ using [1] and [2] respectively. From this simulation it was inferred that the job completion time to be comparatively better using CLS-KTO when compared to [1] and [2]. The reason for the improvement was the application of Kuhn-Tucker Lagrange Optimized Scheduling algorithm. By applying this algorithm, a mathematical model is selected based on inequality constraints satisfying multi-objectives via least slack value. Then, the mathematical model was reformulated in case of local convergence by means of complimentary slack. Finally, by applying CLF principle, optimal results were arrived at, therefore contributing to job completion time using CLS-KTO by 17% compared to [1] and 34% compared to [2].

5. Conclusion

In this paper, we study the MFJSS problem, with the makespan, total workload, and critical workload criteria, which have a strong mathematical background and is very close to the operation research in which jobs are assigned to resources at specific time in an optimized manner. To propose effective optimization model, called, Combinatory Least Slack and Kuhn Tucker Optimization for the MFJSS problem (CLS-KTO), we first apply the Combinatory Least Slack model for machine assignment additive function under the sequence independent processing and transit time. Then, a novel Kuhn-Tucker Lagrange Optimization based on complementary slack and Lagrange Multiplier is specially developed for the MFJSS. It is worth noting that an optimized structured is adopted to avoid local convergence by means of Continual Level Foraging (CLF) principle to enhance the ability to dealwith multiple objectives. This strategy considers theminimization of makespanand job completion time. To show how our MFJSSs work, the effectiveness of key components is also verified, including both processing time and transit time. Moreover, extensive comparisons are carried out for the proposed CLS-KTOagainst the existing state-of-the-art algorithms using benchmark OR. According to the computational results, the proposed CLS-KTO outperform all the other algorithms by a considerable margin, in terms of job completion time, makespan and mean tardiness with respect to job.

References

[1] Ghiath Al Aqel, Xinyu Li and Liang Gao, “A Modified Iterated Greedy Algorithm for Flexible Job Shop Scheduling Problem”, Chinese Journal of Mechanical Engineering, Springer, Mar 2019 [Modified Iterated Greedy (MIG)]
[2] Jia Luo, Shigeru Fujimura, Didier El Baz, Bastien Plazolles, “GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem”, Journal of Parallel Distributed Computing, Elsevier, Jul 2018 [Energy-efficient Dynamic Flexible Flow-shop Scheduling (EDFFS)]
[3] J.E.Beasley, "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society 41(11) (1990) pp1069-1072.
[4] M. Nouri, A. Bekrar, D. Trentesaux, “Towards Energy Efficient Scheduling and Rescheduling for Dynamic Flexible Job Shop Problem”, International Federation of Automatic Control, Elsevier, Jul 2018
[5] Behnam Barzegar, Homayun Motameni, Hossein Bozorgi, “Solving Flexible Job-Shop Scheduling Problem Using Gravitational Search Algorithm and Colored Petri Net”, Journal of Applied Mathematics, Hindawi Publishing Corporation, Apr 2012
[6] Anis Gharbi, Mohamed Labidi, and Mohamed Aly Louty, “The Non permutation Flow shop Scheduling Problem: Adjustment and Bounding Procedures”, Journal of Applied Mathematics, Hindawi Publishing Corporation, Nov 2014
[7] Lianfei Yu, Cheng Zhu, Jianmai Shi, Weiming Zhang, “An Extended Flexible Job Shop Scheduling Model for Flight Deck Scheduling with Priority, Parallel Operations, and Sequence Flexibility”, Scientific Programming, Hindawi, Feb 2017
[8] Dominik Kress, David Müller, “Mathematical models for a Flexible Job Shop Scheduling Problem with Machine Operator Constraints” International Federation of Automatic Control, Elsevier, Jul 2019
[9] Pisut Pongchaierks, “A Two-Level Metaheuristic Algorithm for the Job-Shop Scheduling Problem”, Complexity, Wiley, Mar 2019
[10] Feifei Zheng, Zhaojie Wang, “Bi-objective Flexible Job Shop Scheduling with Operation Overlapping Costs”, International Federation of Automatic Control, Elsevier, Sep 2019
[11] Dominik Kress, David Müller, Jenny Nossack, “A worker constrained flexible job shop scheduling problem with sequence-dependent setup times”, OR Spectrum, Springer, Nov 2018
[12] Jinwei Gu, Manzhan Gu, Xiwen Lu, Ying Zhang, “Asymptotically optimal policy for stochastic job shop scheduling problem to minimize makespan”, Journal of Combinatorial Optimization, Springer, Apr 2018
[13] Raviteja Buddala, Siba Sankar Mahapatra, “An integrated approach for scheduling flexible job-shop using teaching–learning-based optimization method”, Journal of Industrial Engineering International, Springer, Jun 2018
[14] Tianhua Jiang, Chao Zhang, Huiqi Zhu, Guanlong Deng, “Energy-Efficient Scheduling for a Job Shop Using Grey Wolf Optimization Algorithm with Double-Searching Mode”, Mathematical Problems in Engineering, Hindawi, Oct 2018
[15] Humyun Fuad Rahman, Ruhul Sarker, Dary, Essam, “Multiple-order permutation flow shop scheduling under process interruptions”, The International Journal of Advanced Manufacturing Technology, Springer, May 2018
[16] Raviteja Buddala, Siba Sankar Mahapatra, “Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown”, The International Journal of Advanced Manufacturing Technology, Springer, Sep 2018
[17] Qianwang Deng, Guiliang Gong, Xuran Gong, Like Zhang, Wei Liu, Qinghua Ren, “A Bee Evolutionary Guiding Nondominated Sorting Genetic Algorithm II for Multiobjective Flexible Job-Shop Scheduling”, Computational Intelligence and Neuroscience, Hindawi, Mar 2017
[18] Yubang Liu, Shouwen Ji, Zengrong Su, Dong Guo, “Multi-objective AGV scheduling in an automatic sorting system of an unmanned (intelligent) warehouse by using two adaptive-genetic algorithms and a multi-adaptive-genetic algorithm”, PLOS ONE | https://doi.org/10.1371/journal.pone.0226161 December 6, 2019
[19] Maryam Mousavi, Hwa Jen Yap, Siti Nurmayar Musa, Farzad Tahiriri, Siti Zawiah Md Dawal, “Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization”, PLOS ONE | DOI:10.1371/journal.pone.0169817 March 6, 2017
[20] Fei Gao, Meizhen Wang, Xuejun Liu, Ziran Wang, “A multi-objective scheduling optimization algorithm of a camera network for directional road network coverage”, PLOS ONE | https://doi.org/10.1371/journal.pone.0206038 October 31, 2018