Optimum Scheduling and Routing of Material Through Computational Techniques

Pratik Mahesh Suryawanshi

1 Introduction

The scheduling problems can be modelled into virtually innumerable problem types [2]. Thus identifying the correct problem definition is as important as having an impeccable solution for the problem under consideration. The annals in operations research have suggested various solutions for scheduling and routing problems based on their area of execution and demand of the application. Modern production scenario has moved on to agile manufacturing and just-in-time manufacturing practices where every minute change in the process reflect on the final outcome of the production house. Thus, the objective of scheduling and routing is not to finish the activity in a minimum possible time but at the right time, where neither the activities lag nor it leads to the planned schedule. There are various practices adopted by the industry for this task which involves stern planning, careful execution and sustainable outcome. Over the years the computation has developed a human-like execution capability, which involves decision-making and trend prediction, and such computation based on large statistical data primarily known as artificial intelligence has promised the potential to augment the digitization of human culture [14]. The fourth industrial revolution expects vertical as well as horizontal communication to occur on such a level that virtually every unit of the system communicates and feedbacks everything. In such a scenario, traditional techniques tend to stall the outreach of modern principles which primordially are caused due to the mainstay of these techniques. The fourth industrial revolution is characterized by 'data collection' and 'data analysis', which is completely different from the second or third industrial revolution. In both of these scenarios, 'data' is not priorly concerned over other production aspects. These modern demands of the industrial revolution can be coped up with by amalgamating them with computational techniques which are characterized to operate

P. M. Suryawanshi (✉)
Marvelous Engineers Pvt. Ltd., E-12/G-36, Gokul-Shirgaon M.I.D.C., Kolhapur, Maharashtra, India

A. Chakrabarti et al. (eds.), *Industry 4.0 and Advanced Manufacturing*, Lecture Notes in Mechanical Engineering, https://doi.org/10.1007/978-981-19-0561-2_5

effectively with these requirements [3]. At the human level, it is not possible to process this large data and that too simultaneously. Thus, considering these points a schema was brainstormed that would consolidate all the needs of the industry without compromising the current system. The proposed solution was the development of a computational program by the development team that would aid the supply chain and production management team in their daily and overall scheduling and routing problems. These computational techniques can study large data in a very short time and provide optimum solutions with a fair acceptance rate, thus reducing the workload of managers so that they can focus on uncertainty management. The data from the company was first identified and then modelled into a mathematical model which was subjected to constraints so as to gain the desired outcome.

2 Scheduling and Routing Techniques

Initially in the company, no specific scheduling or routing technique was used. The sole work was done by the experienced supply chain and production managers who used to macro and micro plan the production. Thus, the initial task performed was analysis and identification of a suitable technique that can be effectively applied. After studying various traditional techniques [2, 10] it was proposed that these techniques with certain modifications can be implemented in the company as an algorithm that would automatically compute the schedule and aid the activity in the company. Total tardiness was the principal method used to model the problem which was further enhanced by other optimization techniques. The apparent tardiness cost (ATC) minimization model was also considered to avoid earliness in production. The heuristic method as elaborated [6] was first used to find the coefficients; later mixed integer programming model was designed which describes the objective function subject to the greedy randomized adaptive search procedure (GRASP) which selects feasible solution from the set of best-restricted candidate list (RCL), out of which a random set is selected again to perform a local search with small changes in the solution to find the better value of objective function [8]. This aided the macro-level scheduling while the micro-level scheduling which included consideration of single machines, operating parallelly with some being identical and some not, was modelled with best-suited methodologies [1] with release date consideration [4]. For the sake of computation, algorithms were developed [5, 15], which were first written as pseudocode and then programmed. Following are the analytical treatments applied to develop the objective function, constraints and map them to the company's variables:

m = Machine number (0,1,2…i)

m = 0,1,2,3,4,5; where 0,1,2,3 are CNC and 4,5 are VMC

j = Jobs or batches (0,1,2…n)

j = 0,1,2,3,4,5,6,7,8,9

B = Big positive number

l = Position in sequence (0,1,2…n)

d_j = Due date of batch j

p_{ij} = Processing time of batch j on machine i

w_j = Weight of batch j

X_{il} = 1 if job is in sequence L, otherwise 0

S_{ij} = Starting time of job j on machine i

C_{ij} = Completion time of job j on machine i

T_j = Tardiness of job j

Thus, the objective function would be Eq. (1)

$$\min \sum w_j T_j \tag{1}$$

subjected to

$$C_{ij} \geq S_{ij} + d_j$$
$$S_{(i+1)j} \geq C_{ij}$$
$$S_{ij} + \left(1 + X_{j(l+1)}\right) * B \geq C_{im} - (1 - X_{ml}) * B$$
$$T_{ij} \geq C_{ij} - -d_j T_j \geq 0$$
$$S_{ij} \geq 0$$

3 Software Development Through Python and XML

Based on mathematical models, the most feasible programming language that would satiate the requirements was python because of its cross-platform support, number of open-source libraries and future scope to comply with the Industry 4.0 ready machine learning and deep learning systems. Python (3.9.5) was used in the development of this algorithm. The machine used for this program development is an Intel® Core™ i3-7100 @ 3.6 GHz, 4 GB RAM @ 2400 MHz, Intel® HD Graphics 630 operating on Ubuntu (20.0.4 LTS). The programming is compiled by its default command line-based compiler and typed in gedit. The programming went through standard stages of develop-deploy-test-redefine based on the outcomes observed [11]. Several versions with different features were tested and released out of which the best suited and applied is elaborated in this research piece.

3.1 Libraries Deployed

The primary need was the provision to define and solve the linear programming which was enabled by deploying the library 'mip'. The eminent features of the mip are the ability to solve highly complex linear models, model the system in higher language, dynamic header loading, multiple solvers and such to note [12]. The scheduling model based on Eq. (1) along with its variables and constraints was constructed in mip and CP-SAT solver was used to provide the first possible optimization solutions.

As the data on the shop floor is neither static nor constant, thus it gave channelization in the direction of dynamic modelling and data analysis. This requirement was fit as a solution provided by the 'Google OR-Tools'. OR-Tools is an open-source python library that provides combinatorial optimization with the help of a Glop solver. 'Glop' has its own set of assumptions which compulsory are subjected to the algorithms for which it is subjected to. The dominant feature of this library and solver is its dynamic data handling range and the ability to provide the best solution out of a very large set of possible solutions [9]. This library is not limited to classic linear programming problems but can also be coded as per the user's need for further flexible computation by enhancing the source code provided by the developers.

After the successful trial of the mathematical model, the challenge was to develop a user-friendly U.I. and a platform so that the algorithm can be accessed with minimum effort. For this, the 'tkinter' library was used, which is a standard GUI development library accompanied by python [7]. Using tkinter, a simple GUI was designed considering ergonomic factors which are as demonstrated in Fig. 2. Further, the ergonomic enhancement included eliminating the need to run the algorithm by compiling and executing it. Thus, an executable was compiled which would suit both Linux and Windows operating systems using a very simple yet elegant library called 'auto-py-to-exe', which with minimal efforts produces the executable from desired python script with a bundle of interesting accessories like icon selection, standalone execution and so on [13].

3.2 Data Collection and Handling

It is a general practice that the data required in an industry is mostly stored and handled in Microsoft Office Excel. Task-specific spreadsheets are prepared accordingly which have the production capacity, deadlines and such related data specified. The schedule under analysis is for the 14th and 15th week of the year 2021. This is the time period under consideration because this program is developed in the 11th and 12th week of the year 2021. This time span also has a fixed deadline as the 21st week of the year 2021. Thus, in order to check the program's effectiveness and compare it with the current system, the stated schedule plan was selected as it is and almost immediately executed. The major constraint here was the format in which the data was stored. The company has an integrated ERP software which maintains data pertaining to related

processes such as cycle time, batch quantity, sequence of operations and so on, as represented in Table 1.

Using the data extracted from the ERP and considering the batch dispatch required, the time analysis was performed which yielded Table 2. In this table the actual inputs which are to be given to the program are determined based on general computation and are expressed as below. The selection of a particular machine for a specific job is decided based on the capacity, possible setting and operator availability for the scheduling time under consideration.

Table 1 Raw data extracted from ERP

Batch number	Part number	Process sequence and execution time (minutes)					
A0	4550	CNC Setting	CNC Operation	VMC Setting	VMC Operation		
		150	19	120	17		
A1	4550	CNC Setting	CNC Operation	VMC Setting	VMC Operation		
		150	19	120	17		
B2	1730	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	11.5	70	9.5	90	10.5
B3	1730	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	11.5	70	9.5	90	10.5
B4	1730	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	11.5	70	9.5	90	10.5
B5	1730	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	11.5	70	9.5	90	10.5
C6	4700	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	15	70	7	80	20.75
C7	4700	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	15	70	7	80	20.75
C8	4700	CNC Setting	CNC Operation	CNC Setting	CNC Operation	VMC Setting	VMC Operation
		70	15	70	7	80	20.75
D9	0270	CNC Setting	CNC Operation	VMC Setting	VMC Operation		
		120	25	120	20		

Table 2 Machine number and operation time (minutes) for respective batches

Batch number	Machine for operation 1	Operation 1	Machine for operation 2	Operation 2	Machine for operation 3	Operation 3
A0	0	3304	4	2942	NA	NA
A1	0	1119	4	987	NA	NA
B2	1	1025	2	859	5	962
B3	1	1094	2	916	5	1025
B4	1	576	2	488	5	552
B5	1	450	1	384	5	437
C6	3	2665	2	1281	4	3670
C7	3	1690	2	826	4	2321
C8	3	2230	2	1078	4	3068
D9	0	1495	5	1220	NA	NA

This data was then extracted to Excel spreadsheets in raw form. This data in raw form did not have a peculiar format or structure. Thus, a generalized format was proposed as shown in Fig. 1, which was maintained across the departments pertaining to this research. The data required for analysis is maintained in the Excel files, which was first condensed down to required variables. The data corresponding to variables under consideration was then exported into a comma-separated values (.csv) file for the sake of ease of data handling for the program. The .csv files can be opened in any text editor or an office Excel handler, thus the need for another data handler was eliminated without compromising the effectiveness of the code. The extracted scheduling .csv file is given as an input to the program through its navigation panel.

	Operation 1		Operation 2		Operation 3	
Batch 1	Machine Number for Operation 1 of Batch 1	Time to be consumed by Operation 1of Batch 1	Machine Number for Operation 2 of Batch 1	Time to be consumed by Operation 2 of Batch 1	Machine Number for Operation 2 of Batch 1	Time to be consumed by Operation 3 of Batch1
	Operation 1		Operation 2			
Batch 2	Machine Number for Operation 1 of Batch 2	Time to be consumed by Operation 1of Batch 2	Machine Number for Operation 2 of Batch 2	Time to be consumed by Operation2 of Batch 2		
	Operation 1		Operation 2		Operation 3	
Batch 3	Machine Number for Operation 1 of Batch 3	Time to be consumed by Operation 1of Batch 3	Machine Number for Operation 2 of Batch 3	Time to be consumed by Operation 2 of Batch 3	Machine Number for Operation 3 of Batch 3	Time to be consumed by Operation 3 of Batch 3
	Operation 1					
Batch 4	Machine Number for Operation 1 of Batch 4	Time to be consumed by Operation 1of Batch 4				
	Operation 1		Operation 2		Operation 3	
Batch 5	Machine Number for Operation 1 of Batch 5	Time to be consumed by Operation 1of Batch 5	Machine Number for Operation 2 of Batch 5	Time to be consumed by Operation 2 of Batch 5	Machine Number for Operation 3 of Batch 5	Time to be consumed by Operation 3 of Batch 5

Fig. 1 Standardized format for data collection and handling

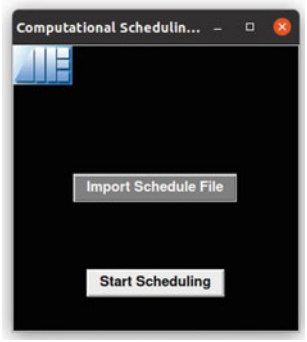

Fig. 2 Application user interface

The data from the scheduling file is imported into the algorithm in the form of a multi-dimensional array, and on those arrays the computational constraints are applied to get the best possible outcome (Fig. 3).

3.3 Sample of Code Deployed

The process of development is continuous which implies that there is not a confirmed or final version but the most acceptable outcome is considered to be standard and thus is elaborated further in this paper.

3.3.1 Pseudocode

```
Start
Input Selection
      Input "schedule.csv into program" with ',' as de-
      limiter and '/n' as end of line
Processing Input
      Len = number of lines in schedule.csv file
      Map input as tuple for 'Len/2' with ',' as sepa-
      rator
      Save as data frame
Assigning Variables
      Assign Machine Id(mid) to every first term of tu-
      ple
      Assign Process Time(duration) to every second
      term of tuple
```

```
      If third term present:
            Append as deadline(dl) in tuple entry on
      (task_id)
      Assign Process Start Time (start_var)
      Assign Process Completion(end_var)
            Assign number of tuples as operation se-
      quence(task_id)
      Assign number of Len as number of batches(job_id)
      Assign batch number as Len changes
      all_tasks = {List of lists[job_id, task_id]}
Algorithm
      For every task:
            end_var - start_var = duration
      Precedence for no overlap:
            For all [job_id, task_id + 1].start >=
            all_tasks[job_id, task_id].end)
            No overlap: If (job_id[1],
            task_id[1].start_var <= (job_id[2],
            task_id[2]).start_var for mid[1]==mid[1]:
            [start_var[1] + duration[1] <=
            start_var[2]]
      Deadline Constraint:
            For all [job_id, task_id].end_var <=
            [job_id, task_id].dl
      Objective Function:
            For all minimize[sum[task_id.duration]
Display the most minimum outcome in command line
File Outputs
      Create and write data in text file
      Open schedule.xml file
      Append the solution to corresponding variables
END
```

3.3.2 Final User Interface and Deployment

Following is the simplified app UI. It has two options, the one on top to import the schedule .csv file while the below one starts the scheduling when clicked.

After selecting the CSV file containing the scheduling data in the required manner, 'Start Scheduling' is clicked on the U.I. after which the algorithm computes the shortest possible cycle time for given inputs. This output is displayed in the command line interface (C.L.I.) as shown in Fig. 3. Also, the schedule is produced as a text file output which is created by the program and saved in the program's directory.

Further, this schedule obtained is used to edit a .xml file present in the same directory which produces the Gantt chart, a very vital tool for floor level reference and analysis. Here the variables are appended as 'Component ID [batch number, operation number]'. This .xml file can be opened using Project Libre or Microsoft Project and shows the output as represented in Fig. 4.

```
Optimal Schedule Length: 12033
Machine 0: job_9_0          job_1_0          job_0_0
           [0,1495]         [1495,2614]      [2614,5918]
Machine 1: job_3_0          job_2_0          job_4_0          job_4_1          job_5_0          job_6_1
           [0,1094]         [1094,2119]      [2119,2695]      [2695,3183]      [3183,3633]      [6585,7866]
Machine 2: job_7_1          job_2_1          job_5_1          job_3_1          job_8_1
           [1690,2516]      [2516,3375]      [3633,4017]      [4017,4933]      [4933,6011]
Machine 3: job_7_0          job_8_0          job_6_0
           [0,1690]         [1690,3920]      [3920,6585]
Machine 4: job_9_1          job_1_1          job_7_2          job_8_2          job_0_1
           [1495,2715]      [2715,3702]      [3702,6023]      [6023,9091]      [9091,12033]
Machine 5: job_4_2          job_5_2          job_2_2          job_3_2          job_6_2
           [3183,3735]      [4017,4454]      [4454,5415]      [5415,6440]      [7866,11536]
```

Fig. 3 Schedule output in the terminal

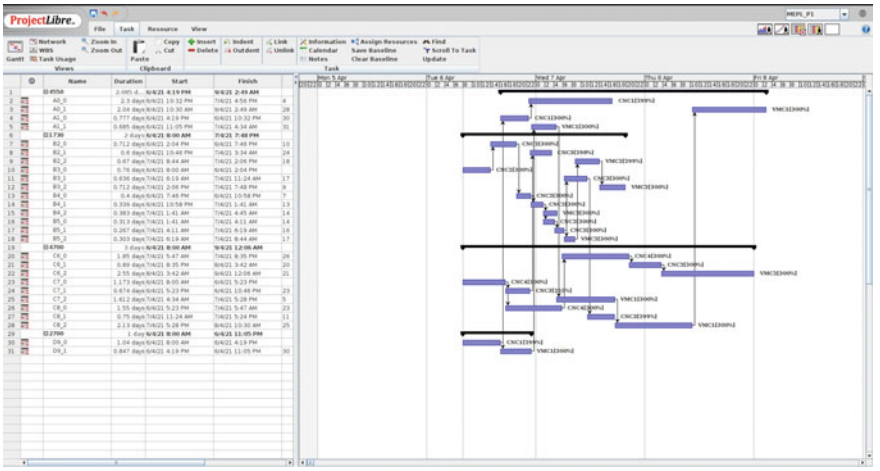

Fig. 4 Optimized scheduled as displayed in project Libre

4 Results and Discussion

The company currently uses a critical ratio and shortest delivery time first scheduling assisted by manual insights of the managers. This technique is used by assigning an algorithm to the Excel sheet. Following is the algorithm and its result applied to the schedule under consideration:

```
  Get Today (start); Batches' process durations (bdu);
Batches' deadlines (bdl)
  For n iterations (n = 1,2,3… = batch number)
  1st Iteration:
      C.R. = (bdl-start)/bdu for all Batches
      (C.R.)least1 = Batch Priority
      For respective batches:
      (bdl) - bduleast1 = (bdl)1
      (bdl)1 = new deadline for respective batch
```

Table 3 Critical ratio-based scheduling result

Batch number	Priority	Duration (min)
A0	3	6246
A1	7	2106
B2	8	1884
B3	5	3035
B4	9	1616
B5	10	1271
C6	1	7616
C7	4	4837
C8	2	6376
D9	6	2715
Total	**–**	**37,702**

Based on the above old algorithm, the following results were obtained by the production department, as shown in Table 3.

This technique lacks the consideration of parallel machining; thus, it was a manual task for the production department to assign duty to every machine for the next couple of days, every day after the second shift ends (16:00 h) by filling individual plan sheet on every machine. This task consumed their productive time to do similarly repeated tasks every day that too with less effectiveness. The scheduling activity is proposed by Mr. Surve (Production Manager, Marvelous Engineers Pvt. Ltd.), who proposed a schedule of 17,280 min considering parallel machining and routing as per his experience and manual problem-solving. For our schedule under consideration, he instructed us to consider a 25% lag in the schedule given by the new algorithm due to the uncertainties like power cuts, accident like such. These results have been condensed in the conclusion part which elaborates the effectiveness of this new algorithm.

5 Insights and Conclusion

The algorithm suggested that all of the input work can be finished in 12,033 min. When this is compared to the manual schedule designed by Mr. Surve which is 17280 min, there is a 30% time-saving in scheduling. If we consider a 25% lag in the programmed schedule due to uncertainties as per instructions, we still get 13.11% time-saving in the total schedule which is 2266 min. This time-saving is vital in production as it directly indicates the monetary benefits obtained. When the schedule was actually deployed on the floor, it was observed that 2357 min (13.64%) were saved as compared to the old scheduling technique. Also, this program takes parallel machining and deadline computation into consideration.

The algorithm has a major constraint of overlooking real-time scenario on the machine floor which can be eliminated by using machine learning which would identify and modify the inputs and algorithm accordingly. Further, IoT would enable real-time GUI-based scheduling as it would upload data, and at the same time, the data would be processed. Also, IoT would completely eliminate the need for manual data handling and would thus further rectify the SCM department's activities.

Acknowledgements The author would like to express gratitude for the support from Marvelous Engineers Private Limited, Kolhapur. The author appreciates great help from Mr. Sangram V. Patil (Managing Director, Marvelous Engineers Pvt. Ltd.) for his constant guidance and motivation and Mr. Sudhir S. Bakare (Technical Director, Marvelous Engineers Pvt. Ltd.) for his professional suggestions along with technical direction on this research work and whole Marvelous Engineering Pvt. Ltd. Team for their enthusiastic support.

References

1. Biskup D (1999) Single-machine scheduling with learning considerations. Europ J Oper Res, 173–178
2. Brucker P (2001) Scheduling algorithms. 5th ed. Springer, Berlin, Heidelberg, Germany, pp 1–36, pp 61–154, 243–280
3. Dalenogarea LS, Beniteza GB, Ayalab NF, Frank AG (2018) The expected contribution of Industry 4.0 technologies for industrial performance, pp 383–394, Elsevier B.V. https://doi.org/10.1016/j.ijpe.2018.08.019
4. Goemans MX, Queyranne M, Schulz AS, Skutella M, Wang Y (2002) Single machine scheduling with release dates. Siam J Discrete Math 15(2):165–192
5. Lawler EL (1994) Knapsack-like scheduling problems, the moore-hodgson algorithm and the 'tower of sets' property. Mathl Comput Model 20(2):91–106
6. Lee YH, Bhaskaran K, Pinedo M (1997) A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IEEE Trans 29:45–52
7. Lundh F (1999) An introduction to tkinter. www.pythonware.com/library/tkinter/introduction/index.htm
8. Molina-Sánchez LP, González-Neira EM (2016) GRASP to minimize total weighted tardiness in a permutation flow shop environment. Int J Ind Eng Comput 7:161–176. https://doi.org/10.5267/j.ijiec.2015.6.004
9. Perron L, Furnon V (2021) OR-Tools (9.0) [Python Library] https://developers.google.com/optimization/
10. Pinedo ML (2001) Scheduling theory, algorithms and systems. 4th ed. Springer, New York Dordrecht Heidelberg London, pp 243–280. https://doi.org/10.1007/978-1-4614-2361-4
11. Sampson A, McKinley KS, Mytkowicz T (2017) Procedings of the ACM on programming languages, Vol 1, pp 71–97, Elsevier B.V. https://doi.org/10.1145/3133895
12. Toffolo TAM, Santos HG (2021) Python-MIP (1.13.0) [Python Library] https://www.python-mip.com/
13. Vollebregt B (2021) auto-py-to-exe (2.9.0) [Python Library] https://github.com/brentvollebregt/auto-py-to-exe
14. Wagner PBJ, Koke B (2018) International conference on information management and processing (ICIMP), pp 82–88, IEEE. https://doi.org/10.1109/ICIMP1.2018.8325846
15. Xi Y (2013) Heuristic algorithms to minimize total weighted tardiness on the single machine and identical parallel machines with sequence dependent setup and future ready time. Theses Dissertations, 184. https://dc.uwm.edu/etd/184