Multiobjective Optimisation of Job Shop Scheduling of Renewable Powered Machinery

Abdihakim Bokah
School of Engineering
University of Aberdeen
Aberdeen, UK
abdihakim.bokah@abdn.ac.uk

Alireza Maheri
School of Engineering
Centre for Energy Transition
University of Aberdeen
Aberdeen, UK
alireza.maheri@abdn.ac.uk

***Abstract*— In this paper a multiobjective job shop scheduling (JSS) problem formulation is presented in which the machinery load is fully or partially supplied by renewable resources. The makespan and the unmet load are used as the optimisation objectives. An NSGA-II is used for finding Pareto solutions. The algorithm is integrated in the software MOHRES, which evaluates the performance of the renewable system. Through a case study, the performance of the algorithm is evaluated, and it is shown how the algorithm can successfully find Pareto solutions with minimal makespan and unmet load.**

Keywords—job shop scheduling, hybrid renewable energy systems, sustainable production, JSS, HRES, MOHRES

I. INTRODUCTION

Job shop scheduling (JSS) problem is a class of optimisation problem with a wide range of applications in process planning and manufacturing. More recently there has been a growing interest in solving JSS problem in the context of sustainable and environmental-friendly production [1-4]. One particular aspect of which, relevant to the focus of this paper, is where the energy is supplied entirely or partially by renewable resources [5-7].

As a well-established research area, a vast amount of published papers on JSS problems can be found in the literature. Irrespective of the type of JSS problem, one can classify most of the recent works, with some overlaps, based on their focus on the optimisation technique and the number of optimisation objectives. References [8-15] report the development of robust optimisation methods, including extensions to well-known methods such as genetic algorithms (GA), Tabu Search (TS) Particle Swarm Optimisation (PSO) and Ant Colony (AC), as well as other meta-heuristic and hybrid methods. Solutions to a JSS optimisation problem can be evaluated with respect to a series of criteria. The most commonly used assessment criterion is the overall processing time (makespan). The minimum makespan can be found by solving a single objective optimisation problem. While the majority of the published work is on single objective optimisation formulation, many published works take into account more than one criterion for the evaluation of solutions and solve JSS problem with multiple objectives such as makespan, tardiness and energy consumption [16-20].

In this paper, first we present a multiobjective JSS problem formulation with makespan and unmet load as two conflicting objectives to be minimised. An NSGA-II developed for multiobjective optimisation of JSS problem is then briefly explained followed by a case study in which we evaluate the performance of the optimisation algorithm.

II. JSS PROBLEM DEFINITION AND OPTIMISATION FORMULATION

A vast number of real-life applications can be modelled in the form of a classical JSS problem. In classical JSS problems, which are the focus of this study, machines required to deliver all tasks are of different types. For a total number of N_t tasks to be delivered, there are N_m machines of N_m different types. In other words, tasks are assigned to specific machines, more than one task cannot be processed on the same machine at the same time and a process cannot be interrupted before completion. Visiting the literature, one notices that many reported formulations apply some unrealistic simplifications such as assuming that no two tasks from the same job are assigned to the same machine. Moreover, having a closer look at the broadly used benchmark problems one finds two completely unnecessary simplifications: (i) the number of machines required to deliver a job is the same for all jobs and (ii) all machines are required to deliver each one of the jobs in the job set. The proposed formulation of JSS problem is more general and allows us to solve more realistic cases. In summary, in this formulation:

- the number of tasks in jobs can vary from one job to another,
- the number of machines required to deliver a job can vary from one job to another,
- the number of machines required to deliver a job can be smaller than the number of tasks in that jobs.

Fig. 1 shows how a typical JSS problem can be defined according to the formulation above (coded in MATLAB). Parameter $job(j).task$ is a vector of length $n_{t,j}$ containing task names (or index) in job j with the right precedence; $job(j).machine$ and $job(j).time$ are also vectors of length $n_{t,j}$ containing the name (or index) of machines associated to the tasks in $job(j).task$ and the processing time for each task. Parameter $machine\_p$ contains the nominal power of each machine. As it can be observed from this example, the six jobs on this problem contain different number of tasks. In case of Job 1, it can be seen that both Tasks 4 and 1 are processed on the same machine (Machine 1). That is, the number of tasks on this job is different from the number of required machines (4 versus 3). In this example, there are 10 machines in the workshop but only 8 of them (Machines 1 to 7 and 9) are required for delivering this set of jobs.

```
job(1).task=[4,9,1,6];
job(1).machine=[1,2,1,4];
job(1).time=[20,30,20,30];

job(2).task=[2,5,7];
job(2).machine=[2,5,3];
job(2).time=[10,20,30];

job(3).task=[103,108,110,111,112];
job(3).machine=[1,7,3,5,4];
job(3).time=[40,30,30,30,20];

job(4).task=[203,208,210,211,212];
job(4).machine=[1,9,3,5,4];
job(4).time=[20,10,30,20,10];

job(5).task=[22,25,27];
job(5).machine=[2,5,3];
job(5).time=[30,20,40];

job(6).task=[301,302,303,304,305,306,307,308,309,310];
job(6).machine=[7,9,3,5,4,2,9,5,6,7];
job(6).time=[40,20,10,20,30,20,20,20,40,30];

machine_p=[700,900,1100,1300,700,1900,1400,800,1100,15
00,1600,1800,400,1500];
```

Fig. 1. A typical JSS problem without unnecessary constraints

In this figure the times are in minutes and powers in watt. Fig. 2 shows an unscheduled Gant chart for the JJS Problem of Fig. 1. In this figure each row shows a job with the tasks from left to right in the right precedence. Machines are colour coded.

The optimisation problem is formulated as:

$$min\, \boldsymbol{Y} \tag{1}$$

subject to:

$$S_{i+1,j} \geq C_{i,j} (= S_{i,j} + P_{i,j}),\ \forall i \in \{1, 2, \dots, n_{t,j}\},\ \forall j \in \{1, 2, \dots, N_j\} \tag{2}$$

$$m_{i,j} = m_{k,l} \Rightarrow (S_{i,j} \geq S_{k,l} + P_{k,l}) \vee (S_{k,l} \geq S_{i,j} + P_{i,j}),\ \forall i \in \{1, 2, \dots, n_{t,j}\} \tag{3}$$

$$S_{1,j} \geq J_j,\ \forall j \in \{1, 2, \dots, N_j\} \tag{4}$$

where,

$\boldsymbol{Y} = [y_1, y_2, \dots, y_{N_{obj}}]$ is the vector of N_{obj} objectives;

N_j is the number of jobs;

$n_{t,j}$ is the number of tasks on job j

$S_{i,j}$, $P_{i,j}$ and $C_{i,j}$ are the start time, processing time and completion time of task i on job j (task $t_{i,j}$);

$m_{i,j}$ is the machine associated to task $t_{i,j}$;

$C_{n_{t,j},j}$ is the completion time of the last task of job j;

J_j is the arrival time of job j;

N_m is the number of machines required to deliver all jobs.

Objectives y_i can be defined as the overall makespan ($L = \max\{C_{n_{t,j},j}\}, \forall\, j \in \{1, 2, \dots, N_j\}$), and the renewable power deficit or the unmet load P_u, which represents the reliability of renewable power supply in a standalone hybrid renewable energy system (HRES) or the cost of energy from the grid, in case of grid-connected HRES.

The unmet load P_u depends on the HRES configuration (the combination of renewable/storage/backup components), size of each component in the HRES, renewable resources (e.g. wind and solar irradiance profiles), and the load profile. The load profile in industrial applications can be divided into two parts (i) primary load and (ii) machinery load. The primary load profile is due to facilities in the workshop such as lighting and HVAC systems. The primary load profile is fixed and independent of job scheduling. On the other hand, the machinery load profile depends on how the jobs are scheduled. The unmet load can be calculated using the following equations:

$$P_u = P_R + P_s - p' - p'' \tag{5}$$

in which, P_R is the renewable power (e.g. wind, PV, fuel cell), P_s is the extractable power from storage system (e.g. battery bank or hydrogen tank), p' is the primary load and p'' is the machinery load. In an hourly averaged basis, the daily unmet load is given by:

$$P_u = \sum_{t=1}^{24} (P_R + P_s - p' - p'')_t \tag{6}$$

$$p''_t = \left[\sum_{m=1}^{N_m} p_m\right]_t \tag{7}$$

in which, p_m is the power consumption of machine m and $[p_m]_t$ is the power consumption of machine m during hour t.

Hourly averaged renewable power P_R depends on the hourly averaged renewable resources and the type, size and power model of renewable components. The hourly averaged renewable resources depend on the site and is a given input. Similarly, the primary load p' depends on the site and is a given input. The hourly averaged extractable power from the storage system P_s during hour t depends on the size and type of the storage system as well as the history of renewable production and power consumption during hours 1 to $t-1$.

III. NSGA-II FOR MULTIOBJECTIVE OPTIMISATION OF JSS PROBLEM

A nondominated sorting genetic algorithm (NSGA-II) is developed specifically for solving JSS problems formulated as above. The chromosome is a matrix of size $N_m \times n_{t,m,max}$, where $n_{t,m,max}$ is the maximum number of tasks on each machine given by Equation 8. It should be noted that in the generalised JSS formulation, the number of tasks assigned to a machine, $n_{t,m}$, can vary from one machine to another.

$$n_{t,m,max} = max\{n_{t,m},\ \forall m \in \{1, 2, \dots,\ N_m\}\} \tag{8}$$

Fig. 3 shows a typical chromosome, here populated with a sampler solution of the JSS problem of Fig. 1. In this chromosome, each gene refers to a task number (or name). Genes with values 0 are blank tasks, allowing different jobs have different number of tasks. Each row of this matrix represents a machine and the set of tasks, which are allocated to that machine in the right order of operation. Using a 2D chromosome allows us to benefit from the advantages of geometric crossover in boosting the performance of the GA. The crossover operator is shown in Fig. 4.

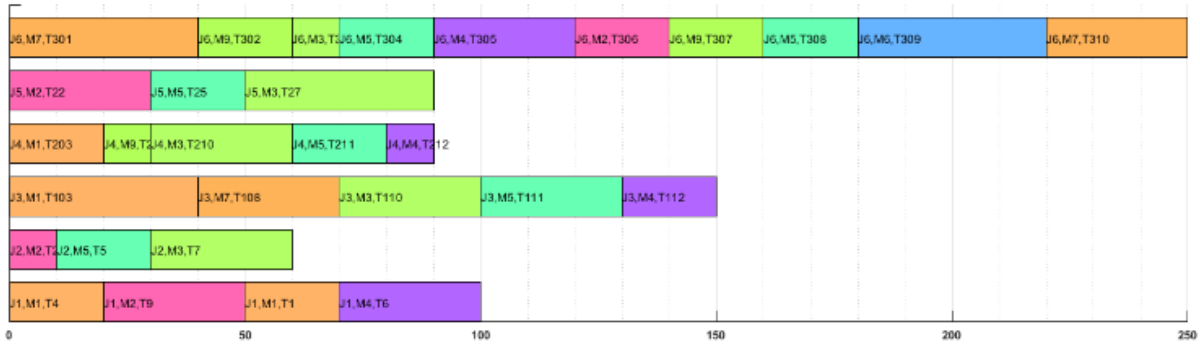

Fig. 2. Unscheduled Gant chart of JJS Problem of Fig. 1 on jobs

Machine 1	103	203	4	1	0	0
Machine 2	2	9	22	306	0	0
Machine 3	7	303	210	27	110	0
Machine 4	305	6	212	112	0	0
Machine 5	5	304	25	211	308	111
Machine 6	309	0	0	0	0	0
Machine 7	301	108	310	0	0	0
Machine 9	302	208	307	0	0	0

Fig. 3. A typical 2D chromosome representing solutions of JSS problem.

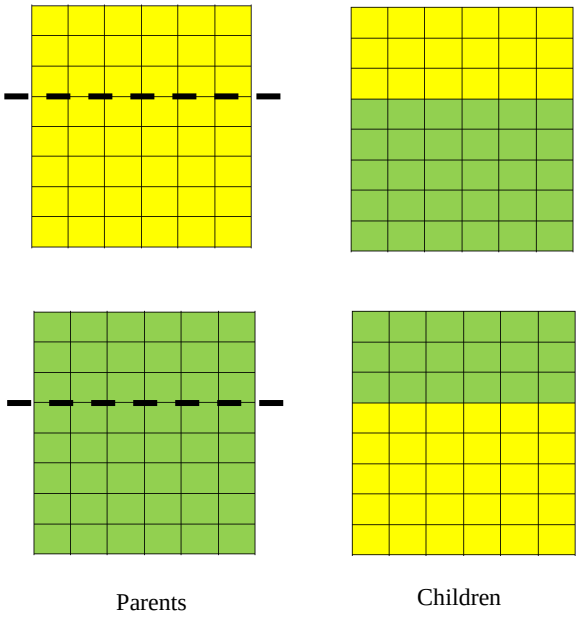

Fig. 4. Geometric crossover of JSS 2D chromosomes

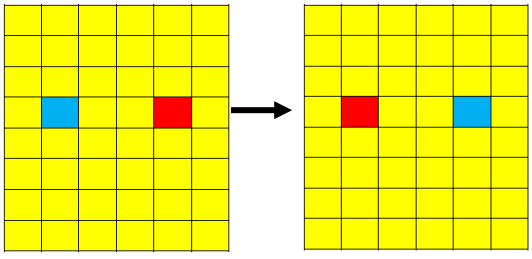

Fig. 5. Mutation-swapping tasks on a randomly selected machine

Using a tournament selection two parents are randomly selected. A randomly cut point divides the parents into upper and lower parts. The upper and lower parts from two parents are then recombined to make two new children. By doing so, children inherent part of the machines from one parent and the other part from another parent.

The mutation operation consists of two steps. In the first step a machine (a row in the 2D chromosome) is randomly selected for mutation and then in the second step two randomly selected genes (tasks on that machine) are swapped to form a new solution (see Fig. 5). Blank tasks (identified by 0 in the chromosome) are excluded from selection.

Within the optimisation process, once an individual is generated, whether as a randomly generated member of the initial population, or as the result of the crossover and mutation operations, a repair algorithm is called. The repair algorithm first checked for the Constraints 2 to 4 above. If any of them are contradicted, the algorithm, where possible, heuristically fixes the individual towards satisfying all the constraints, otherwise rejects the solution. To allow shifting the whole scheduling within the scheduling period, a starting time is also added to the chromosome. Once a correct solution is generated an evaluator is called. Each individual is evaluated against its JSS performance measures (here, the makespan) as well as its HRES performance measures (here, the unmet load). The optimisation programme is integrated to MOHRES, which delivers the evaluation against HRES performance measures.

IV. Case Study

In this case study we aim at the performance evaluation of the optimisation algorithm above in terms of its capability of (i) finding the solution with minimum makespan, (ii) finding the solution with minimum unmet load, and (iii) producing a relatively well populated Pareto front. For this purpose, we assume a PV renewable system without any storage. It should be noted that the algorithm has no restriction on the configuration of the renewable energy system and MOHRES can deal with a generic wind-PV-battery-fuel cell-electrolyser-diesel configuration. The system is connected to the grid and any unmet load is supplied by the grid.

For a PV system the renewable power P_R during hour t is given by:

$$P_{R_t} = P_{PV} = \eta_{PV} A_{PV} I_t \quad (6)$$

where, I_t in W/m^2 is the hourly averaged solar irradiance during hour t, A_{PV} is the total area of the solar panels, and η_{PV} is the overall efficiency of the of the PV panels.

We assume the site has a solar irradiance and primary load profiles as shown in Fig. 6. With a PV size of $A_{PV} = 300\ m^2$ and a nominal efficiency of $\eta_{PV} = 14\%$, we see the unmet load (negative values) and excess power (positive values) profiles due to the primary load in Fig. 7. The overall unmet load due to the primary load without the machinery load is $P_u = 17600\ W$. The total machinery load is 13067 W.

Based on these figures, any additional machinery load before 9:00 and after 19:00 hours will lead to an increase in the unmet load. That is, we expect to see that a successful optimisation algorithm produces a number of Pareto solution with two extreme solutions, one with the minimum makespan in which all jobs are scheduled at the beginning of the analysis period (here 00:01 min) and another extreme solution with minimum unmet load, in which all jobs are scheduled somewhere between 9:00 and 19:00 hours, where there is excess power.

We use the example of Fig. 1 as our JSS problem case. This problem has a known minimum makespan of 250 minutes (Job 6 starting at t=0 has a makespan of 250 minutes for back to back tasks).

The optimisation problem of Equation 1 becomes:

$$min\{L, P_u\} \tag{7}$$

subject to Constraints 2 to 4.

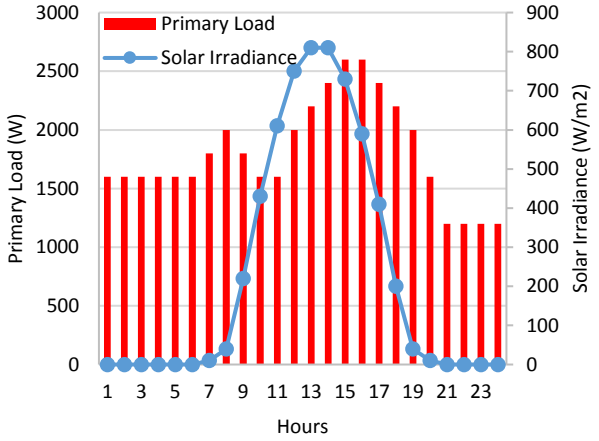

Fig. 6. Solar irradiance and primary load profiles

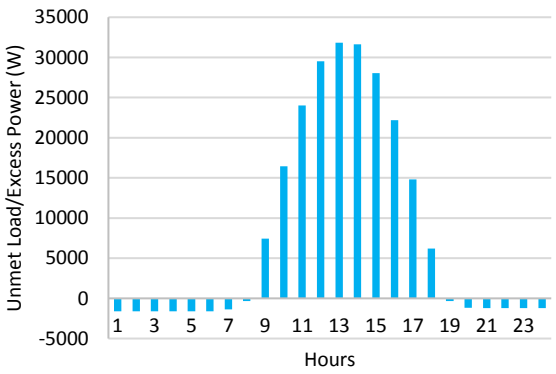

Fig. 7. Unmet load (negative values) and excess power (positive values) profiles

TABLE I. PARETO SOLUTIONS OF OPTIMISATION PROBLEM (9)

Sol #	Start of Operation (min)	Makespan/End of Operation (min)	Operation Time (min)	Overall Unmet Load (W)	Added Unmet Load due to Machinery Load (W)
1	0	250	250	30667	13067
2	0	260	260	30667	13067
3	0	350	350	30667	13067
4	200	490	290	30433	12833
5	110	500	390	30200	12600
6	0	510	510	29967	12367
7	220	520	300	29650	12050
8	230	530	300	29333	11733
9	250	540	290	28833	11233
10	250	550	300	28117	10517
11	210	560	350	27600	10000
12	310	570	260	26200	8600
13	300	600	300	25783	8183
14	330	620	290	25067	7467
15	330	630	300	24300	6700
16	350	640	290	23900	6300
17	350	650	300	23383	5783
18	370	670	300	22683	5083
19	380	680	300	22283	4683
20	400	690	290	21717	4117
21	440	700	260	19833	2233
22	440	730	290	19600	2000
23	450	740	290	19100	1500
24	470	760	290	18100	500
25	480	770	290	17600	0

In this case study, any solution satisfying these constraints is a feasible solution. However, it should be noted that a correct solution (a solution satisfying Constraints 2 to 4) is not necessarily a feasible solution if we include HRES-performance related constraints to this optimisation problem. The optimisation NSGA II algorithm was tuned and the optimisation parameters $P_c = 0.3$, $P_m = 0.8$, $N_{pop} = 50$ and $N_{gen} = 50$ were found to lead to identical extreme solutions in five consecutive runs.

Table I shows the makespan and the added unmet load of the 25 nondominated solutions as well as their start of operation time, total operation time and the overall unmet load. The makespan is measured from 00:01 minutes to the time when all tasks are completed, while the total operation time is the actual time taken to deliver the tasks. Fig. 8 shows the Pareto front, in which Solution #1 is the best in terms of the makespan and Solution #25 is the best solutions with respect to the unmet load.

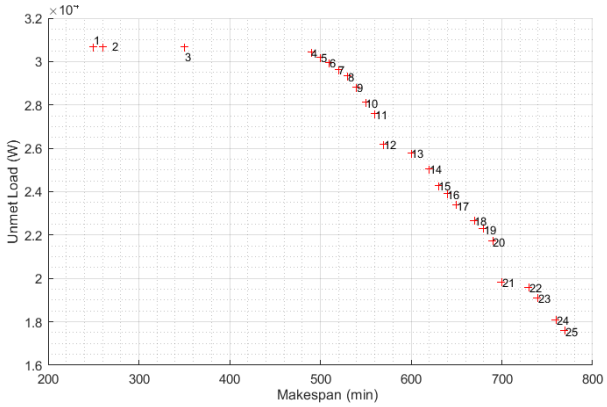

Fig. 8. Pareto front solutions

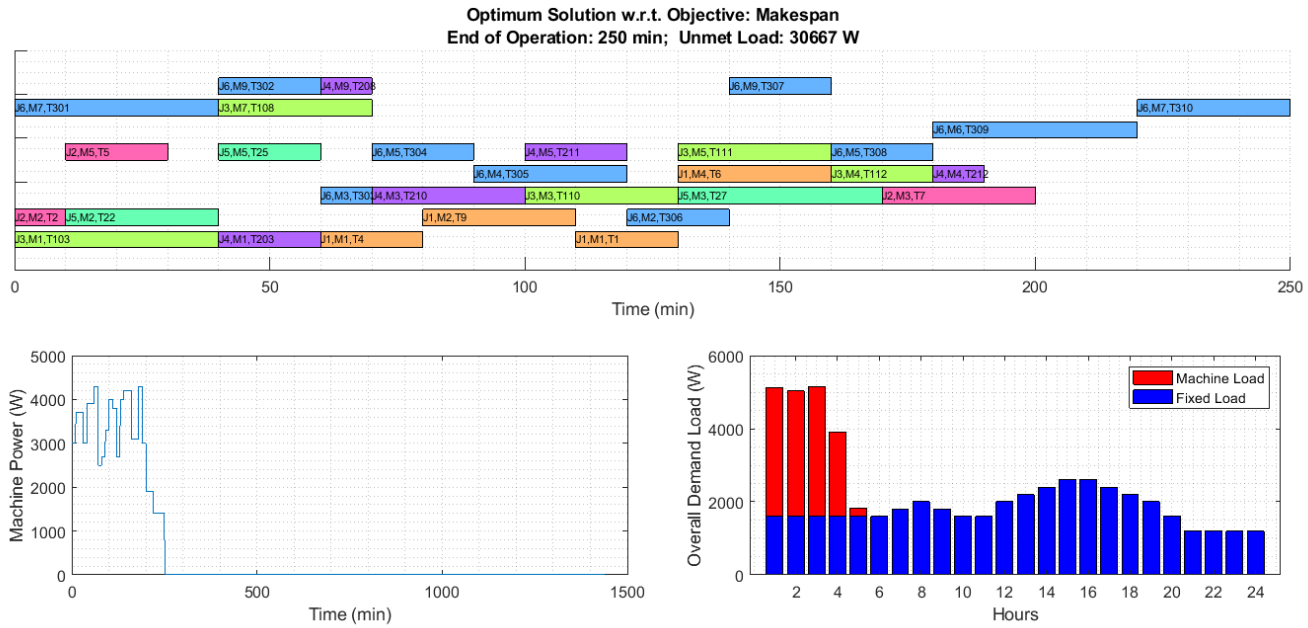

Fig. 9. Gant chart and load distribution of Solution #1, the optimum solution w. r. t. makespan L

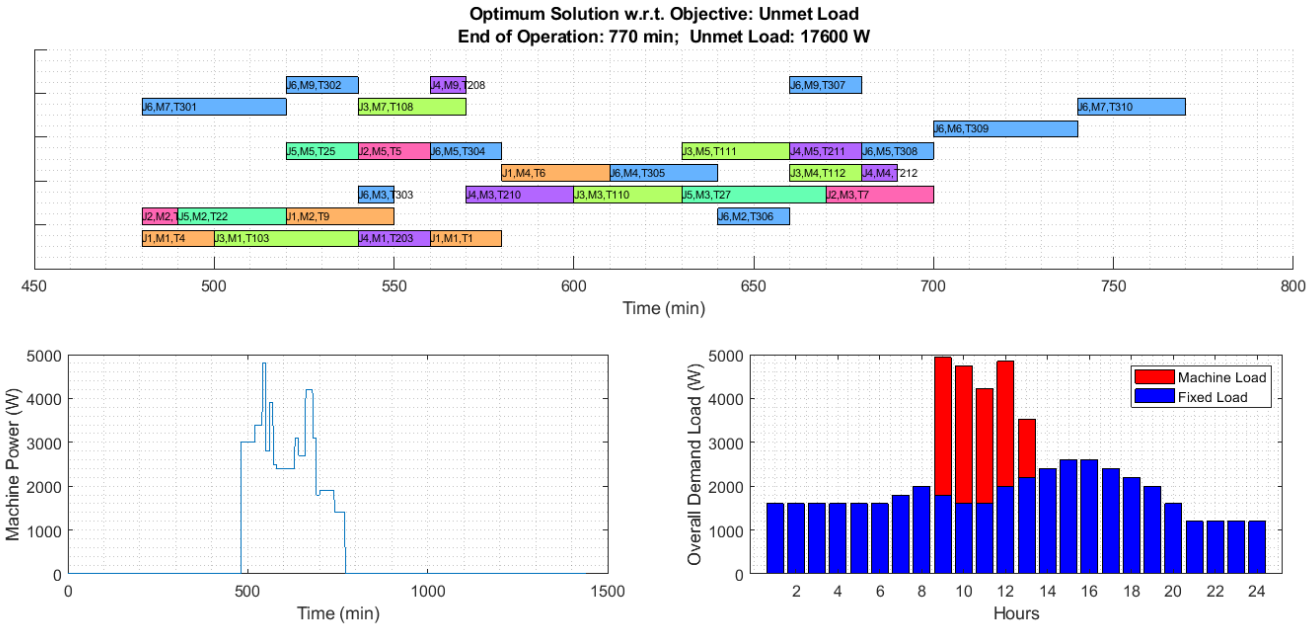

Fig. 10. Gant chart and load distribution of Solution #25, the optimum solution w. r. t. unmet load P_u

The extreme solution w. r. t. the first objective, makespan, has a makespan of 250 minutes which is the known minimum makespan for this JSS problem. The extreme solution w. r. t. the second objective, unmet load, produces no more unmet load besides the already existing one due to the primary load. That is, the unmet load stays at 17600 W. Also, since this solution is a nondominated solutions, it must have the minimum possible makespan. Starting from hours 9:00 it has a makespan of 290 minutes.

Fig. 9 and Fig. 10 show the Gant chart of the extreme solutions (Solutions #1 and #25), as well as their machine load profiles and the overall demand load profiles. In these figures the Gant chart are defined on machines. Each row shows a machine operation with tasks colour coded depending on which job they belong to.

V. CONCLUSION

Solving JSS problem in the context of renewable-powered manufacturing and sustainable and environmental-friendly production has received an increasing interest in the recent years. The multiobjective JSS problem formulation presented in this paper employs an NSGA-II algorithm for solving JSS problem with two objectives of minimising the makespan and minimising the dependency of the process on the grid electricity. Integrating the optimisation algorithm with the software tool MOHRES allows for evaluation of the performance of the renewable system in supplying the

machinery demand load. The case study reported in this paper proves the performance of the optimisation algorithm in terms of its capability of finding the solution with minimum makespan, finding the solution with minimum unmet load, and producing a relatively well populated Pareto front.

REFERENCES

[1] R. Menghi, A. Papetti, M. Germani and M. Marconi "Energy efficiency of manufacturing systems: A review of energy assessment methods and tools," Journal of Cleaner Production, vol. 240, 2019, 118276, ISSN 0959-6526, https://doi.org/10.1016/j.jclepro.2019.118276.

[2] L. Jenny, C. Diaz and C. Ocampo-Martinez, "Energy efficiency in discrete-manufacturing systems: Insights, trends, and control strategies," Journal of Manufacturing Systems, vol. 52, pp. 131-145, 2019.

[3] O. Biel and C.H. Mlock, "Systematic literature review of decision support models for energy-efficient production planning," Computers & Industrial Engineering, vol. 101, pp. 243–259, 2016.

[4] L. Gan, P. Jiang, B. Lev and X. Zhou, "Balancing of supply and demand of renewable energy power system: A review and bibliometric analysis," Sustainable Futures, vol. 2, 2020, 100013, ISSN 2666-1888, https://doi.org/10.1016/j.sftr.2020.100013.

[5] A. Nayak, S. Lee and J. W. Sutherland, "Dynamic Load Scheduling for Energy Efficiency in a Job Shop with On-site Wind Mill for Energy Generation," Procedia CIRP, vol. 80, pp. 197-202, 2019, https://doi.org/10.1016/j.procir.2018.12.003.

[6] S. Wang, S. J. Mason and H. Gangammanavar, "Stochastic optimization for flow-shop scheduling with on-site renewable energy generation using a case in the United States," Computers & Industrial Engineering, vol. 149, 2020, 106812, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2020.106812.

[7] R. Ramezanian, M. M. Vali-Siar and M. Jalalian, "Green permutation flowshop scheduling problem with sequence-dependent setup times: a case study," International Journal of Production Research, vol. 57, pp. 3311-3333, 2019.

[8] L. Asadzadeh, "A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy," Computers & Industrial Engineering, vol. 102, pp. 359-367, 2016.

[9] A. Jamili, "Robust job shop scheduling problem: Mathematical models, exact and heuristic algorithms," Expert Systems with Applications, vol. 55, pp. 341-35015, 2016.

[10] W.Y. Ku and J. C. Beck, "Mixed Integer Programming models for job shop scheduling: A computational analysis," Computers & Operations Research, vol. 73, pp. 165-173, 2016.

[11] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian and V. Mahmoodian, "An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs," Computers & Industrial Engineering, vol. 86, pp. 2-13, 2015.

[12] A. Ponsich and C. A. Coello Coello, "A hybrid Differential Evolution—Tabu Search algorithm for the solution of Job-Shop Scheduling Problems," Applied Soft Computing, vol. 13, pp. 462-474, 2013

[13] N. Kundakcı and O. Kulak, "Hybrid genetic algorithms f.or minimizing makespan in dynamic job shop scheduling problem," Computers & Industrial Engineering, vol 96, pp. 31-51, 2016.

[14] X. Li and L. Gao, "An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem," International Journal of Production Economics, vol. 174, pp. 93-110, 2016.

[15] W. Sukkerd and T. Wuttipornpun, "Hybrid genetic algorithm and tabu search for finite capacity material requirement planning system in flexible flow shop with assembly operations," Computers & Industrial Engineering, vol. 97, pp. 157-169, 2016.

[16] R. Zhang and R. Chiong, "Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," Journal of Cleaner Production, vol 112, pp. 3361-3375, 2016.

[17] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," Computers & Industrial Engineering, vol 54, pp. 453-473, 2008

[18] J.Q. Li, Q. K. Pan and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," Applied Mathematical Modelling, vol. 38, pp. 1111-1132, 2014.

[19] V. Kaplanoğlu, "An object-oriented approach for multi-objective flexible job-shop scheduling problem," Expert Systems with Applications, vol. 45, pp. 71-84, 2016.

[20] V. Majazi Dalfard and G. Mohammadi, "Two meta-heuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel machine and maintenance constraints," Computers & Mathematics with Applications, vol. 64, pp. 2111-2117, 2012.

[21] MOHRES documents. www.mohres.com

[22] A. Maheri, "Multi-objective design optimisation of standalone hybrid wind-PV-diesel systems under uncertainties," Renewable Energy, vol. 66, pp. 650-661, 2014.

[23] A. Maheri, "A critical evaluation of deterministic methods in size optimisation of reliable and cost effective standalone hybrid renewable energy systems," Reliab. Eng. Syst. Saf., vol. 130, pp. 159-174, 2014.