An Energy-Efficient Single Machine Scheduling with Release Dates and Sequence-Dependent Setup Times

M. Fatih Tasgetiren
Department of International Logistics Management
Yasar University, İzmir, Turkey
fatih.tasgetiren@yasar.edu.tr

Uğur Eliiyi
Department of Computer Science
Dokuz Eylül University, İzmir
Turkey
ugur.eliiyi@deu.edu.tr

Hande Öztop
Department of Industrial Engineering
Yasar University, İzmir
Turkey
hande.oztop@yasar.edu.tr

Damla Kizilay
Department of Industrial Engineering
Yasar University, İzmir
Turkey
damla.kizilay@yasar.edu.tr

Quan-Ke Pan
State Key Laboratory
Huazhong University of Science and Technology, Wuhan, P.R. China
panquanke@hust.edu.cn

ABSTRACT

This study considers single machine scheduling with the machine operating at varying speed levels for different jobs with release dates and sequence-dependent setup times, in order to examine the trade-off between makespan and total energy consumption. A bi-objective mixed integer linear programming model is developed employing this speed scaling scheme. The augmented ε-constraint method with a time limit is used to obtain a set of non-dominated solutions for each instance of the problem. An energy-efficient multi-objective variable block insertion heuristic is also proposed. The computational results on a benchmark suite consisting of 260 instances with 25 jobs from the literature reveal that the proposed algorithm is very competitive in terms of providing tight Pareto front approximations for the problem.

CCS CONCEPTS

Applied computing → Operations research → Decision analysis → Multi-criterion optimization and decision-making

KEYWORDS

Multi-objective optimization, Energy efficient scheduling, Heuristic optimization, Sequence dependent setup times, Speed scaling.

1 ENERGY EFFICIENT VBIH ALGORITHM

In the energy-efficient scheduling problem handled in this study, a set of n jobs are to be scheduled on a single machine for minimizing the makespan and total energy consumption (TEC). The notation, parameters, decision variables, and the mixed integer programming model (MILP) are given below:

J Set of jobs $\{0,1,2,\dots,n\}$
L Set of speed levels
p_j Processing time of job $j \in J$
r_j Release date of job $j \in J$
s_{ij} Sequence dependent setup time for changing from job i to job j
v_l Speed factor of speed level $l \in L$
λ_l Conversion factor for speed level $l \in L$
φ Conversion factor for idle time
μ Power of machine
y_{jl} 1 if job j is processed with speed level l, 0 otherwise
x_{ij} 1 if job i precedes job j, 0 otherwise
t_j Time at which job j starts its setup
θ Idle time on machine
C_{max} Maximum completion time (makespan)
TEC Total energy consumption

$$Minimize\ C_{max} \quad (1)$$

$$Minimize\ TEC \quad (2)$$

$$\sum_{i\in J} x_{ij} = 1 \quad \forall j \in J, i \neq j \quad (3)$$

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in J, i \neq j \quad (4)$$

$$\sum_{l\in L} y_{jl} = 1 \quad \forall j \in J \quad (5)$$

$$r_j \leq t_j \quad \forall j \in J \quad (6)$$

$$t_j + \sum_{l\in L} \frac{p_j}{v_l} y_{jl} + \sum_{i\in J: i\neq j} s_{ij}\, x_{ij} \leq C_{max} \quad \forall j \in J \quad (7)$$

$$t_i + \sum_{l\in L} \frac{p_i}{v_l} y_{il} + \sum_{q\in J: q\neq i} s_{qi} x_{qi} + U(x_{ij} - 1) \leq t_j \quad \forall i \in J, j \in J / j_0\ i \neq j \quad (8)$$

$$\theta = C_{max} - \sum_{j\in J} \sum_{l\in L} \frac{p_j}{v_l} y_{jl} - \sum_{j\in J} \sum_{k\in J: k\neq j} s_{jk}\, x_{jk} \quad (9)$$

$$TEC = \sum_{j\in J} \sum_{l\in L} \frac{\mu p_j \lambda_l}{60 v_l} y_{jl} + \frac{\varphi\mu}{60}\left(\sum_{j\in J} \sum_{k\in J: k\neq j} s_{jk} x_{jk} + \theta\right) \quad (10)$$

$$t_0 = 0 \quad (11)$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in J \qquad y_{jl} \in \{0,1\} \quad \forall j \in J, l \in L$$
$$t_j \geq 0 \quad \forall j \in J \quad (12)$$

The objective functions (1) and (2) minimize the makespan and TEC, respectively. Constraints (3) and (4) determine the precedence relations for the job sequence. Constraint (5) guarantees that exactly one speed level is selected for each job. While constraint (6) ensures that each job starts earliest at its release date, constraint (7) computes the makespan value. Constraint (8) ensures that the next job in the sequence can be started only after preceding job has been finished., where U is a large number. Constraint (9) calculates the idle time on the machine and constraint (10) computes the total energy consumption of the machine in kilowatt hours. Constraint (11) fixes a dummy job j_0 as the first job in the sequence. Finally, constraint (12) defines all decision variables.

We use a multi-chromosome solution representation as shown below. Speed scaling strategy has three levels, namely, fast, normal and slow (1, 2 or 3).

$x_i(\sigma, v)$

σ	4	1	2	5	3	…	n
v	3	1	2	1	2	…	3

We employ a modified variant of the variable block insertion heuristic (VBIH) used in [1]. The algorithm starts with a constructive heuristic (CH_NEH), it removes a block b of jobs

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07
https://doi.org/10.1145/3205651.3205714

from the current solution, then it makes a number $n-b+1$ of block insertion moves sequentially in the partial solution ($bMove()$), and the best move is retained in order to undergo a single job-insertion-based local search. The obtained solution replaces the current incumbent solution if it is better. Initially, the block size is fixed to $b=1$, and is incremented by one ($i.e., b=b+1$) at each iteration. The block move search is carried out until $b \leq b_{max}$. The outline of the VBIH algorithm is as follows:

$\sigma = \text{CH\_}NEH(x)$
$\sigma_{best} = \sigma$
while $(NotTermination)$ *do*{
  $b = 1$
  do{
    $\sigma_1 = bMove(\sigma)$
    $\sigma_1 = InsertionLocalSearch(\sigma_1)$
    if $\left(f(\sigma_1) < f(\sigma)\right)$ *then do*{
      $\sigma = \sigma_1$
      if $f(\sigma_1) < f(\sigma_{best})$ *then do*{
        $\sigma_{best} = \sigma_1$
    } *endif*
  $b = b + 1$
  }*while* $(b \leq b_{max})$
}*endwhile*
return σ_{best} *and* $f(\sigma_{best})$
endprocedure

Energy-efficient VBIH (EEVBIH) considers the speed levels in all procedures of VBIH algorithm, and can be outlined as follows:

Initialization

1. Set population size, NP=100 and $b_{max}=5$. Obtain the CH_NEH(x) solution, use it in VBIH algorithm as an initial solution and run VBIH for 10% of the total CPU time budget in order to get the best solution σ_{best}.
2. Assign $v_1=1, v_2=2, v_3=3$ to σ_{best} separately and construct the first three solutions in the population
3. Construct the rest of the population by assigning $x_i(\sigma_{ij}, v_{ij}) = x_i(\sigma_{best}, v_{ij}), v_{ij} = \text{rand}()\%3, \forall i = 1,..,NP, \forall j = 1,..,n$.

Makespan Minimization

1. For each individual $x_i(\sigma_{ij}, v_{ij})$ in the population, apply the block insertion move, $bMove()$, and $InsertionLocalSearch()$, considering the job and speed level together.
2. If the new solution dominates the incumbent solution $x_i(\sigma_{ij}, v_{ij})$, replace it with the new solution and update the archive set Ω.

Energy Minimization

1. For each individual $x_i(\sigma_{ij}, v_{ij})$ in the population, keep the same permutation from previous stage.
2. For each individual x_i, find another individual x_k randomly and apply uniform crossover operator to speed vectors only by using solutions x_i and x_k
3. If the new solution dominates the incumbent solution $x_i(\sigma_{ij}, v_{ij})$, replace it with the new solution and update the archive set Ω.

End

Report non-dominated solutions from the archive set **Ω**

2 COMPUTATIONAL RESULTS

To evaluate the performance of the proposed EEVBIH algorithm, we employed the benchmark suit in [2]. Note that we employed only 25 jobs with all range values consisting of 300 instances. However, only the results for the first 260 instances are reported, as CPLEX could not solve 40 instances within the given time limit. The three processing speed factors associated with the slow, normal and fast speed levels are set as v_l = {0.8, 1, 1.2}. Similarly, the conversion factors are λ_l = {0.6, 1, 1.5} for slow, normal and fast speed levels, respectively. The machine power is 60 kW and the conversion factor for idle time is 0.05.

Non-dominated solution sets are obtained for all instances using the augmented ε-constraint method, considering C_{max} as objective and TEC as a constraint. Ranges of C_{max} and TEC are obtained from payoff tables using lexicographic optimization. Next, the model with C_{max} objective is repetitively solved by reducing the constraint on TEC with a specific ε level, which is defined by dividing the range of TEC objective function to 20 equal intervals. In each iteration, IBM ILOG CPLEX 12.6 is used to solve model on a Core i7, 2.60 GHz, 8 GB RAM computer. Due to the NP-hard nature of the problem, a 3-minute time limit is set for each iteration. The proposed EEVBIH algorithm is coded in C++ language. Thirty replications are made for each instance. In each replication, the algorithm is run for 25 seconds. Among these thirty runs, non-dominated solutions are reported.

Non-dominated solution sets of EEVBIH (E) and time-limited CPLEX (M) are compared with each other in terms of spacing (*S*), coverage (*C*) and cardinality measures. The average measure results based over the release date range factors are reported for both methods below. For some instances, time-limited CPLEX found some Pareto-optimal solutions. However, for most of the instances, model cannot be solved optimally in any iteration and there exist huge optimality gaps.

R	|E|	|M|	C_{EM}	C_{ME}	S_E	S_M	# Pareto-opt
0.2	201	20	0.10	0.25	1.15	0.09	49
0.6	177	18	0.22	0.21	1.18	0.17	8
1.0	163	16	0.41	0.17	1.19	0.24	-
1.4	151	15	0.41	0.18	1.18	0.32	-
1.8	134	13	0.58	0.11	1.23	0.39	-
Overall	169	17	0.32	0.19	1.18	0.23	57

As shown above, EEVBIH finds approximately ten times as many non-dominated solutions. For the comparison of the coverage metric, EEVBIH performs better than the time-limited CPLEX, since the solutions found by EEVBIH weakly dominate 32% of those generated by time-limited CPLEX. In terms of spacing, both methods have small *S* values.

3 CONCLUSION

In this study, a multi-objective MILP model is developed as well as an EEVBIH algorithm is proposed to solve the problem. We employed 300 benchmark instances with 25 jobs in order to test the performance of the algorithm. Time-limited CPLEX found some Pareto-optimal solutions for 57 instances only. However, for most of the instances, model cannot be solved optimally in any iteration, and there exists huge optimality gaps. The developed EEVBIH algorithm is able to find solutions that weakly dominate 32% of those generated by time-limited CPLEX, which is a significant improvement.

REFERENCES

[1] Tasgetiren, M. F., Pan, Q-K, Kizilay, D. & Gao, K. 2016. A variable block insertion heuristic for the blocking flowshop scheduling problem with total flowtime criterion. *Algorithms* 9(4): 71.

[2] Ovacik, I. M. & Uzsoy, R. 1994. Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*; 32(6):1243–63.