# Heuristic Algorithms for Solving Hard Scheduling Problems with Positional Penalties and Controllable Processing Times


Dvir Shabtay — Ben-Gurion University of the Negev

Baruch Mor — ARIEL UNIVERSITY

Liron Yedidsion — amazon


IWDSP POZNAŃ 2021

# Today's Agenda

- Problem Definition
- Known Results from the Literature
- Objectives
- Heuristic Algorithms
- Experimental Study
- Directions for Future Research

# Problem Definition

- We are given a set of $n$ jobs, $J=\{1,\ldots,n\}$, that is available at time zero and is to be non-preemptively scheduled on a single-machine.
- The processing time of job $j$, denoted by $p_j(u_j)$, is a convex decreasing function of the amount of continuous and non-renewable resource, $u_j$, allocated to its processing operation.

# Problem Definition

$$p_j(u_j) = B_j + \left(\frac{w_j}{u_j}\right)^k$$

<u>Parameters</u>:

$B_j$ - a lower bound on the processing time of job $j$.

$w_j$ - the workload of job $j$.

$k$ – parameter common to all jobs.

<u>Decision Variables:</u>

$u_j$ - the amount of resource allocated to job $j$.

# Definition of a Solution

- A solution $\boldsymbol{S}$ to our problem is defined by
  - A job processing permutation:

$$\phi = (\phi(1), \phi(2), \dots, \phi(n))$$

  - A resource allocation strategy:

$$\boldsymbol{u} = (u_1, u_2, \dots, u_n)$$

- Both combinatorial and continuous decisions

# Quality of a Solution

- Scheduling Criterion:

$$F_1(S) = \sum_{i=1}^{n} \xi_i \, p_{\phi(i)}(u_{\phi(i)})$$

$\xi_i$ - a positive integer representing the per unit of processing time penalty for assigning any job to the $i$-th position in $\phi$.

- Resource Allocation Cost

$$F_2(S) = \sum_{j=1}^{n} v_j \, u_j$$

$v_j$ - the cost of one unit of resource allocated to the processing of job $j$.

**Table 1**
A subset of single-machine scheduling problems in which the scheduling criterion can be represented as a special case of (5).

| Scheduling Criterion | Positional Penalties ($\xi_i$) |
|---|---|
| $C_{\max}$ | $1$ |
| $\sum_{j=1}^{n} C_j$ | $n+i-1$ |
| $\sum_{s=1}^{n}\sum_{t=s}^{n} \lvert C_s - C_t\rvert$ | $(i-1)(n-i+1)$ |
| $\sum_{s=1}^{n}\sum_{t=s}^{n} \lvert W_s - W_t\rvert$ | $i(n-i)$ |
| $\alpha\sum_{j=1}^{n} E_j + \beta\sum_{j=1}^{n} T_j + \gamma\sum_{j=1}^{n} d_j$ (1) | $\alpha(i-1)+\gamma n$ for $i \leqslant i^*$<br>$\beta(n-i+1)$ for $i > i^*$ |
| $\alpha\sum_{j=1}^{n} E_j + \beta\sum_{j=1}^{n} T_j + \gamma\sum_{j=1}^{n} d_j$ (2) | $\alpha i+\gamma(n+1)$ for $i \leqslant i^*-1$<br>$\beta(n-i)+\gamma$ for $i \geqslant i^*$ |
| $\alpha\sum_{j=1}^{n} E_j + \beta\sum_{j=1}^{n} T_j + \gamma\sum_{j=1}^{n} d_j$ (3) | $(n-i+1)\min\{\beta,\gamma\}$ |
| $\alpha\sum_{j=1}^{n} E_j + \beta\sum_{j=1}^{n} T_j + \gamma_1 n\underline{d} + \gamma_2 nD$ (4) | $\alpha(i-1)+n\gamma_1$ for $i \leqslant i_1^*$<br>$n\gamma_2$ for $i_1^* < i \leqslant i_2^*$<br>$\beta(n-i+1)$ for $i > i_2^*$ |

(1) $d_j = d$ for $j = 1, \ldots, n$ and $d$ is a decision variable. $i^*$ can be computed in constant time.

(2) $d_j = p_j + slack$ for $j = 1, \ldots, n$ and *slack* is a decision variable. $i^*$ can be computed in constant time.

(3) Each job can be assigned a due date with no restrictions.

(4) The scheduler can assign a common due window $\left[\underline{d}, \overline{d} = \underline{d} + D\right]$ where $D$ is a constant,

for the completion time of each job. $i_1^*$ and $i_2^*$ can be computed in constant time.

# Variants of the Problem

- P1: Find a solution $\boldsymbol{S}$ =($\phi$,**u**) which minimizes:

$$F_1(S) + F_2(S) = \sum_{i=1}^{n} \xi_i \, p_{\phi(i)}(u_{\phi(i)}) + \sum_{j=1}^{n} v_j \, u_j$$

- P2: Find a solution $\boldsymbol{S}$ =($\phi$,**u**) which minimizes

$$F_1(S) = \sum_{i=1}^{n} \xi_i \, p_{\phi(i)}\left(u_{\phi(i)}\right)$$

subject to:

$$F_2(S) = \sum_{j=1}^{n} v_j \, u_j \leq U_v$$

$U_v$ - bound on the total resource allocation cost

# Complexity Results from the Literature

| Problem | $\xi_i$ | Complexity | Reference |
|---|---|---|---|
| P1 | $\xi_i$=1 for i=1,…,$n$ | $O(n)$ | Shabtay and Steiner (2007) |
| P1 | Various special cases where $B_j$=0 for j=1,…,$n$ | O($n$log$n$) | Lee and Lei (2001), Shabtay and Kaspi (2004), Yin *et al*., (2016), Wang and Wang (2017) |
| P1 | arbitrary | O($n^3$) | Yedidsion and Shabtay (2017) |
| P2 | NP-hard | For any $\xi_i$ parameters satisfying the condition that $\xi_l \neq \xi_m$ for any $l \neq m$ | Yedidsion and Shabtay (2017) |

# Relevant Results from the Literature

- Given $\phi$, P2 reduces to the following convex programming problem:

$$\text{Min } c(\mathbf{u}) = \sum_{i=1}^{n} \xi_i \, p_{\phi(i)}\left(u_{\phi(i)}\right) = \sum_{i=1}^{n} \xi_i \left( B_{\phi(i)} + \left(\frac{w_{\phi(i)}}{u_{\phi(i)}}\right)^k \right)$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_{\phi(i)} \, u_{\phi(i)} \le U_v$$

# Relevant Results from the Literature

- Using KKT, Yedidsion and Shabtay (2017) showed that the optimal resource allocation strategy as a function of $\phi$ is:

$$u^*_{\phi(i)} = \frac{(\xi_i)^{\frac{1}{k+1}} \left(w_{\phi(i)}\right)^{\frac{k}{k+1}}}{\left(v_{\phi(i)}\right)^{\frac{1}{k+1}} \sum_{i=1}^{n} (\xi_i)^{\frac{1}{k+1}} \eta_{\phi(i)}} U_v \text{ for } i = 1, \ldots, n. \tag{1}$$

- By inserting (1) into the objective value, they obtain that the minimum scheduling cost for a given $\phi$ is given by:

# Relevant Results from the Literature

$$c(\phi, \mathbf{u}^*(\phi)) = c_1(\phi) + (U_v)^{-k}(c_2(\phi))^{k+1}, \quad (2)$$

Where

$$c_1(\phi) = \sum_{i=1}^{n} \xi_i B_{\phi(i)} \quad (3)$$

and

$$c_2(\phi) = \sum_{i=1}^{n} (\xi_i)^{\frac{1}{k+1}} \eta_{\phi(i)}. \quad (4)$$

and $\eta_j = (w_j v_j)^{k/(k+1)}$ for $j$=1,…,$n$

# Relevant Results from the Literature

- Therefore, they conclude that P2 reduces to a sequencing problem of finding $\phi$ minimizing (2).

- Unfortunately, Yedidsion and Shabtay (2017) proved that this problem is NP-hard for any $\xi_i$ parameters satisfying the condition that $\xi_l \neq \xi_m$ for any $l \neq m$.

# Gaps

- The only method exists in the literature for solving P2 is the approximation algorithm by Yedidsion and Shabtay (2017).

- However, this algorithm wasn't tested against any other algorithm or against the value of a tight lower bound.

- Our aim is to help closing this gap in the literature.

# Heuristic Algorithms

- **H1**: A simple Sorting Algorithm ($O(n \log n)$).Sorting Algorithm
- **H2**: Heuristic which is based on the Agent Insertion Method (as the one used by Nawza *et al.* (1983)) ($O(n^3)$ time).
- **H3**: The approximation algorithm of Yedidsion and Shabtay (2017). It is based on solving a series of P1 problems ($O(n^{3+p})$ time, where $n^p$ is the number of P1 problems solved). H3 Algorithm

# Heuristic Algorithms

- **H4**: A Simulated Annealing (SA) algorithm.
- **H5**: Genetic Algorithm (GA).
- **H6:** Selects the best permutation out of 50, 000 randomly generated permutations.

- **We also used the solution obtained by H3 to construct a lower bound (LB) on the objective value**.

# Experimental Study

- Algorithms $H_i$ for $i$ = 1, 2, 3, 4, 5 were implemented in C++ and run on an Intel(R) Core ™ i7-8650U CPU @ 1.90 GHz 16.0 GB RAM platform.
- The programming platform consisted of the 'Visual Studio' software.

# Experimental Study

- For each out of several combinations of $k$ and $n$, we randomly generated a set of 50 numerical instances.
- For each instance, we draw the parameters ($w_j$, $v_j$ and $B_j$) from a discrete uniform distribution ranging between 1 and 20.
- We draw the value of $U_v$ from a discrete uniform distribution ranging between $0.5n10^{2-1/k}$ and $1.5n10^{2-1/k}$.

# Experimental Study

- For each set of problems, we compute:
  - The average relative gap ($avg\delta^i$) of the value of the solution obtained by heuristic **H*i*** (*i*=1,…,6) from the lower bound value.
  - The maximal relative gap (max$\delta^i$) of the value of the solution obtained by heuristic **H*i*** (*i*=1,…,6) from the lower bound value.
  - The average running time ($r.t$) of each heuristic (sec) except from **H1** and **H6**, in which the running time was negligible.

# Results - total completion time objective ($\xi_i = n - i + 1$):

| *n* | *k* | H1 | | *H2* | | | *H3 (p=1)* | | |
|---|---|---|---|---|---|---|---|---|---|
| | | *avg* $\delta$ | *max* $\delta$ | *avg* $\delta$ | *max* $\delta$ | *r.t* | *avg* $\delta$ | *max* $\delta$ | *r.t* |
| 50 | 0.5 | 1.9287 | 3.6936 | 1.4568 | 2.8736 | 0.002 | 0 | 0 | 1.399 |
| 50 | 0.75 | 2.6438 | 20.345 | 2.2923 | 20.125 | 0.002 | 0.6248 | 17.105 | 1.373 |
| 50 | 1 | 1.4230 | 3.6101 | 1.2919 | 3.5772 | 0.002 | 0.0993 | 3.3115 | 1.268 |
| 100 | 0.5 | 2.4580 | 14.758 | 2.1808 | 14.364 | 0.015 | 0.4772 | 12.595 | 50.01 |
| 100 | 0.75 | 2.2495 | 19.725 | 2.0427 | 19.358 | 0.015 | 0.3513 | 17.564 | 37.84 |
| 100 | 1 | 1.6648 | 14.914 | 1.5836 | 14.855 | 0.015 | 0.5082 | 12.971 | 43.79 |
| 150 | 0.5 | 2.5458 | 23.634 | 2.3552 | 23.383 | 0.050 | 0.4232 | 21.158 | 378.8 |
| 150 | 0.75 | 3.4535 | 38.305 | 3.2280 | 37.886 | 0.051 | 1.4935 | 33.744 | 419.0 |
| 150 | 1 | 2.2054 | 27.374 | 2.1256 | 27.332 | 0.050 | 0.9784 | 25.385 | 375.6 |

# Results - total completion time objective ($\xi_i = n - i + 1$):

| *n* | *k* | *H4* | | | *H5* | | | *H6* | |
|---|---|---|---|---|---|---|---|---|---|
| | | *avg δ* | *max δ* | *r.t* | *avg δ* | *max δ* | *r.t* | *avg δ* | *max δ* |
| 50 | 0.5 | 0.0333 | 0.4291 | 0.124 | 0.4297 | 2.0360 | 0.153 | 9.2411 | 11.718 |
| 50 | 1 | 0.6689 | 17.151 | 0.132 | 0.8655 | 17.368 | 0.364 | 11.041 | 30.824 |
| 50 | 2 | 0.1478 | 3.3614 | 0.132 | 0.3186 | 3.6101 | 0.360 | 11.026 | 14.103 |
| 100 | 0.5 | 0.4974 | 12.597 | 0.236 | 1.0846 | 13.325 | 0.436 | 14.490 | 29.217 |
| 100 | 1 | 0.3656 | 17.564 | 0.236 | 0.8793 | 18.451 | 0.912 | 15.536 | 35.284 |
| 100 | 2 | 0.5201 | 12.983 | 0.236 | 0.9671 | 13.461 | 0.904 | 17.207 | 29.487 |
| 150 | 0.5 | 0.4348 | 21.231 | 0.348 | 1.2101 | 22.199 | 0.841 | 16.019 | 42.168 |
| 150 | 1 | 1.5039 | 33.762 | 0.348 | 2.2095 | 34.675 | 1.704 | 19.271 | 57.314 |
| 150 | 2 | 0.9891 | 25.386 | 0.349 | 1.5491 | 26.164 | 1.681 | 20.150 | 46.103 |

# Conclusions

- The PV1-based heuristic (H3) outperforms all other heuristics. It provides a solution that has an average gap of less than 0.44% relative to the lower bound value. This result provides strong evidence for (i) the high quality of the PV1-based heuristic and (ii) the tightness of our lower bound, which seems to match the optimal value in most cases.
- The main disadvantage of the PV1- based heuristic (H3) is its running time, which may become an obstacle when trying to solve instances of over 200 jobs.

# Conclusions

- The two meta-heuristics possess the advantage of a short running time combined with a high quality solution. It seems that for large instances, SA outperforms GA, as the former is able to provide better solutions in a shorter computation time.

# Conclusions

- The strongest evidence for the effectiveness of our heuristics emerges when we compare them to H6, which selects the best out of 50,000 randomly generated permutations. For example,
  - the average percent relative deviation between H6 and the lower bound value over all instances is 14.9%, while it is 0.46% for SA and 0.95% for GA, both of which also generate about 50,000 permutations during the search process.
  - Even H1 and H2, each of which constructs a single solution, yield an average percent relative deviation that is much smaller than that of H6 (it is 2.16% and 1.94% for H1 and H2, respectively, compared to 14.9% for H6).

# Additional Results and Future Research

- We design two exact algorithms.
  - Based on a branch and bound procedure
  - Based on Integer Convex Programming formulation.
- In future research, we aim to extend the experimental study to include this two algorithms.

# Any questions?

$$c(\phi, \mathbf{u}^*(\phi)) = c_1(\phi) + (U_v)^{-k}(c_2(\phi))^{k+1},$$

# Sorting Algorithm

W.L.O.G we assume that $\xi_1 \le \xi_2 \le \cdots \le \xi_n$

$$c_1(\phi) = \sum_{i=1}^{n} \xi_i B_{\phi(i)} \xrightarrow{\phi_1^*}$$ Order the jobs in a non-increasing order of $B_j$

and

$$c_2(\phi) = \sum_{i=1}^{n} (\xi_i)^{\frac{1}{k+1}} \eta_{\phi(i)}. \xrightarrow{\phi_2^*}$$ Order the jobs in a non-increasing order of $\eta_j$

$$LB = c_1(\phi_1^*) + (U_v)^{-k}\, c_2(\phi_2^*)$$

$$UB_1 = c_1(\phi_1^*) + (U_v)^{-k}\, c_1(\phi_1^*)$$

$$UB_2 = c_2(\phi_2^*) + (U_v)^{-k}\, c_2(\phi_2^*)$$

# H1 Algorithm

**Order the jobs in a non-increasing order of**

$$\alpha \boldsymbol{B_j} + (\boldsymbol{1} - \alpha)\boldsymbol{\eta_j}$$

$$\alpha = \frac{c_1(\phi_1^*)}{c_1(\phi_1^*) + (U_v\ )^{-k}\ c_2(\phi_2^*)}$$

# H3 Algorithm

- P1($\alpha$): Find a solution $\boldsymbol{S}=(\phi,\mathbf{u})$ which minimizes:

  $$\alpha F_1(S) + (1-\alpha)F_2(S) = \alpha \sum_{i=1}^{n} \xi_i \, p_{\phi(i)}(u_{\phi(i)}) + (1-\alpha) \sum_{j=1}^{n} v_j \, u_j$$

- P1($\alpha$) is equivalent to P1 and therefore is solvable in O($n^3$) time.
- For $\alpha$=0, the optimal solution is to set $u_j = 0$. The solution is feasible to P2 but obviously not optimal.
- For $\alpha$=1, the optimal solution is to set $u_j \to \infty$. The solution is not feasible to P2.

# H3 Algorithm

- Starting from $[\underline{\alpha}, \overline{\alpha}] = [0,1]$, we solve a series of P1 problems such that
  - the optimal solution for P1($\underline{\alpha}$) is feasible (but not necessarily optimal) for P2, and
  - the optimal solution for P1($\overline{\alpha}$) is not feasible for P2
- We do so as follow:

# H3 Algorithm

- Compute $\alpha = (\underline{\alpha} + \overline{\alpha})/2$.
  - If the optimal solution for P1($\alpha$) is feasible for P2, update

$$\underline{\alpha} = \alpha$$

Otherwise, update

$$\overline{\alpha} = \alpha$$

If at some point the optimal permutation for P1($\underline{\alpha}$) is identical for P1($\overline{\alpha}$), this permutation is optimal.

# H3 Algorithm

- Yedidsion and Shabtay (2017) proved that P1(α) has an approximation ratio of

$$\rho(k) = 1 + \frac{k}{(k+1)^{\frac{k+1}{k}} - k} + \varepsilon = f(k) + \varepsilon,$$

where

$$p \geq \max\{\lceil g \rceil, 2g + 2 - \log_2((1 + \varepsilon/f(k))^{\frac{k+1}{k}} - 1)\},$$