Balancing chains in assembly lines with worker-dependent task times

ALIO/EURO 2021-2022

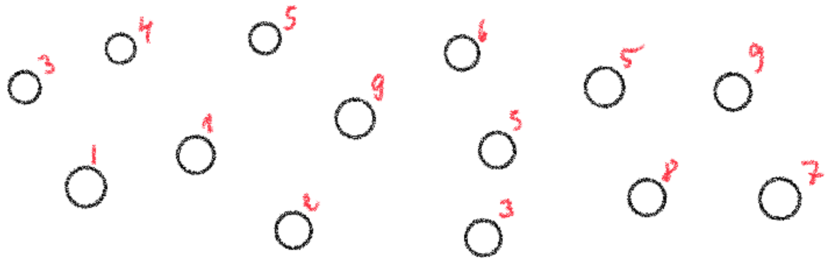
Jordi Pereira Gude (Universidad Adolfo Ibáñez, Chile) and
Marcus Ritt (Universidade Federal do Rio Grande do Sul, Brasil)

April 2022

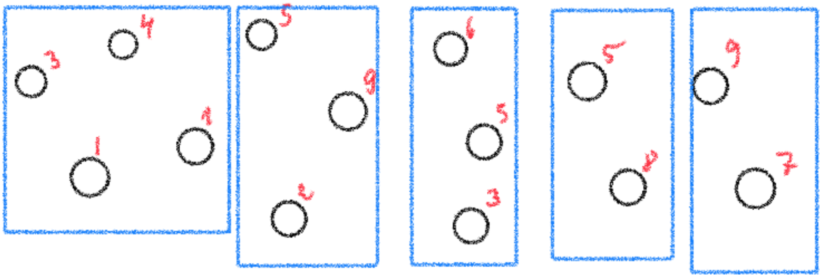
Outline

Outline

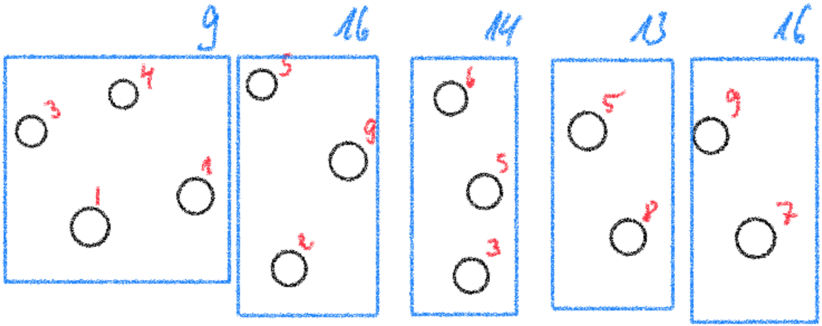
Production line balancing



Production line balancing

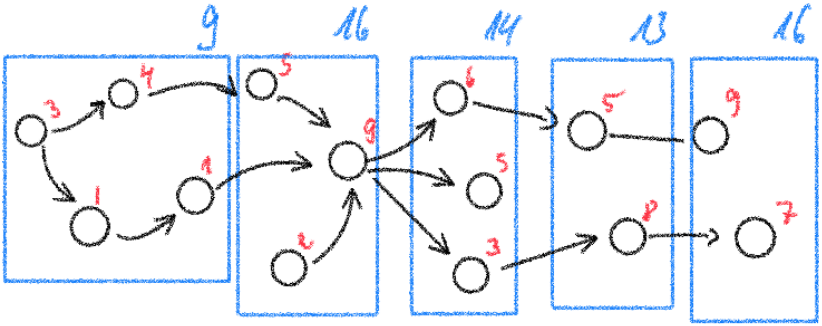


Production line balancing



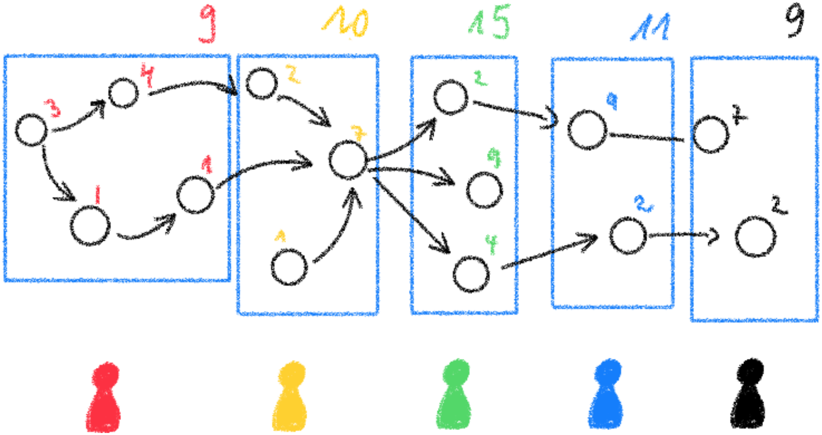
Maximum station time: *cycle time*
$C = \max\{9, 16, 14, 13, 16\} = 16.$

Assembly lines: simple assembly line balancing



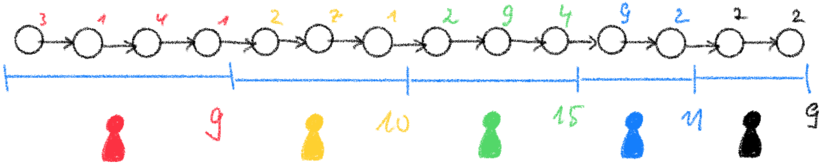
Simple assembly line balancing problem (SALBP, Salveson, 1955).

Assembly lines: worker assignment and line balancing



Assembly line worker assignment and balancing problem (ALWABP, Miralles et. al 2005).

Workload assignment problem (WAP)



Battarra et al., *The Calzedonia workload assignment problem*, J. Oper. Res. Soc. (2021).

Outline

Workload assignment problem (WAP)

Problem data:

- Tasks $T = [n]$, workers $W = [m]$, times t_{ij} for $j \in W, i \in T$.

Feasible assignment:

- Worker permutation $\pi \in \mathfrak{S}_m$
- Sequence $S = (s_1, s_2, \dots, s_{m+1})$ such that $s_1 = 1 \le s_2 \le \dots \le s_{m+1} = n+1$
- For $j \in [m]$, worker π_j executes tasks $[s_j, s_{j+1})$.

Minimize the *cycle time*:

- $C = \min_{j \in W} \sum_{i \in [s_j, s_{j+1}[} t_{i\pi_j} = \min_{j \in W} t_{[s_j, s_{j+1})\pi_j}$.

💡 WAP is NP-hard.

Outline

Known worker permutation

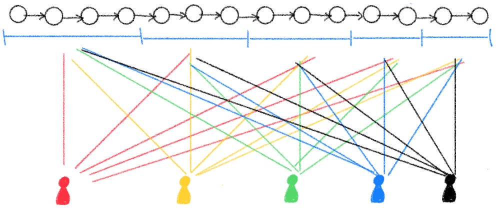
- Worker permutation $\pi \in \mathfrak{S}_m$ known
- Minimal cycle time $C(i, k)$ for tasks $1, \ldots, i$ with workers $\pi_1, \ldots, \pi_k$
- Solved by the dynamic program

$$C(i, k) = \min_{1 \leq j < i} \left\{ \max \left\{ C(j, k-1), t_{(j,i],\pi_k} \right\} \right\} . \qquad (1)$$

- with base cases $C(0, k) = 0,\ C(i, 0) = \infty,\ i > 0$
- in time $O(nm \log n)$ and space $\mathrm{O}(nm)$.

Known intervals

- Sequence $S = (s_1, s_2, \ldots, s_{m+1})$ defining intervals $[s_j, s_{j+1}[$ known
- Solved by a minimum bottleneck perfect matching of intervals to workers
  - For interval $j \in [m]$ and worker $j \in W$ has weight $t_{[s_j, s_{j+1}[, j}$.
- Time $O(m^{5/2} \log m)$ via binary search over $O(m^2)$ candidate times using the algorithm of Hopcroft & Karp (1973).



Outline

Full problem

- Keep track of the *available workers* W'
- Minimal cycle time $C(i, W')$ for tasks $1, \dots, i$ using workers $W' \subseteq W$

$$C(i, W') = \min_{1 \leq j < i, w \in W'} \{\max\{C(j, W' \setminus \{w\}), t_{(j,i]w}\}\} \qquad (2)$$

- Exponential time $O(nm2^m \log n)$ and space $O(n2^m)$.
- Consequence: WAP is fixed-parameter tractable for m

Worker relaxation

- Keep track only of the *number of workers*
- Minimal cycle time $C(i, k)$ for tasks $1, \ldots, i$ using k workers

$$C(i, k) = \min_{1 \le j < i} \left\{ \max \left\{ C(j, k-1), \min_{w \in W} t_{(j,i]w} \right\} \right\} \tag{3}$$

- Time $O(m^2 n \log n)$ and space $O(nm)$.

Partial worker relaxation

- Keep track of *selected workers* $V \subseteq W$ and the *number of remaining workers*, where $\overline{W} = W \setminus V$
- Minimal cycle time $C(i, k, V)$ for tasks $1, \ldots, i$ using k workers from $\overline{W}$ and workers V

$$C(i, k, V) = \min_{1 \le j < i} \begin{cases} \min_{w \in \overline{W}} \left\{ \max\{C(j, k-1, V), t_{(j,i],w}\} \right\} \\ \min_{w \in V} \left\{ \max\{C(j, k, V \setminus \{w\}), t_{(j,i],w}\} \right\} \end{cases} \tag{4}$$

- Exponential time $O(nmm_2 2^{m_1} \log n)$ and space $O(nm_2 2^{m_1})$ for $m_1 = |V|$, $m_2 = m - m_1$.

Partial worker relaxation (feasibility version)

- Maximum number of tasks $S(V, k)$ solvable with workers $V \subseteq W$ and k workers from $\overline{W}$,
  - Pre-compute maximum number of tasks $s(i, w)$ that worker $w \in W$ can perform when starting from task $i + 1$.

$$S(V, k) = \max \begin{cases} \max_{w \in V} \{S(V \setminus w, k) + s(S(V \setminus w, k), w)\} \\ \max_{w \in \overline{W}} \{S(V, k-1) + s(S(V, k-1), w) \end{cases} \tag{5}$$

- Exponential time $O(m 2^{m_1} \log n)$ and space $O(m_2 2^{m_1})$ for $m_1 = |V|$, $m_2 = m - m_1$.

Proposed solution: Successive worker relaxation

- Apply an *upper bound search* for the optimal cycle time
- For each bound on the cycle time C:
  - Solve (5) repeatedly with increasing V
    - Add in each iteration *most frequent repeated* worker to V
    - Break ties by *largest interval*, worker index
  - If workers are repeated: apply bottleneck matching for feasible solution

Outline

Methodology

- Set I (Battarra et al. 2021), 200 instances
  - Average performance (AP, single worker type) and mixed performance (MP, 3 different worker types)
  - Operations $o \in [18, 42]$, workers $m \sim U[9, 15]$, batches of identical tasks, 1300–2200 tasks
- Set II, 200 instances
  - Operations $o \in [43, 67]$, workers $m \sim U[16, 22]$, batches of identical tasks, 2300–3200 tasks
- AMD Ryzen 9, 4.2 GHz, 32 GB main memory

Results Set I

Table: Summary for data set I.

m	N	t[s]	Opt.(%)	Mem[MB]
9	20	0.07	100	44.20
10	28	0.10	100	44.44
11	32	0.15	100	44.87
12	25	0.22	100	45.35
13	29	0.33	100	46.04
14	39	0.55	100	47.27
15	27	0.81	100	48.91

- Previously solved by Pereira & Ritt (2022) about *1 order of magnitude slower*

Results Set II

Table: Summary for data set II.

m	N	t[s]	Opt.(%)	Mem[MB]
16	37	1.67	100	53.10
17	27	4.14	100	63.41
18	27	6.94	100	73.82
19	32	23.33	100	115.81
20	26	48.98	100	197.40
21	26	114.59	100	308.54
22	25	267.95	100	611.79

- All instances solved in about 5 min
- Instances with up to 27 workers reachable

Thanks for the attention!

Comparison to Battarra et al. (2021)

Table: Comparison to the BILS of Battarra et al. (2021): best and average relative deviation of 10 runs for 2 mins.

			BILS	
m	N	t[s]	Best	Avg.
9	20	0.07	0.00	0.00
10	28	0.10	0.00	0.02
11	32	0.15	0.00	0.15
12	25	0.22	0.00	0.58
13	29	0.33	0.00	0.75
14	39	0.55	0.02	1.02
15	27	0.81	0.07	2.07

A mathematical model

- Assignment variables $x_{ij} \in \{0, 1\}$.

$$\begin{aligned}
\textbf{minimize} \quad & C && && (6)\\
\textbf{subject to} \quad & C \geq \sum_{i \in T, j \in W} t_{ij} x_{ij}, && \forall j \in W, && (7)\\
& \sum_{j \in [w]} x_{ij} = 1, && \forall i \in T, && (8)\\
& x_{gj} + x_{ij} \leq 1 + x_{hj}, && \forall g, h, i \in T, j \in W, g < h < i, && (9)\\
& x_{ij} \in \{0, 1\}, && \forall i \in T, j \in W, && (10)\\
& C \geq 0. && && (11)
\end{aligned}$$

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
https://doi.org/10.1080/01605682.2020.1755897

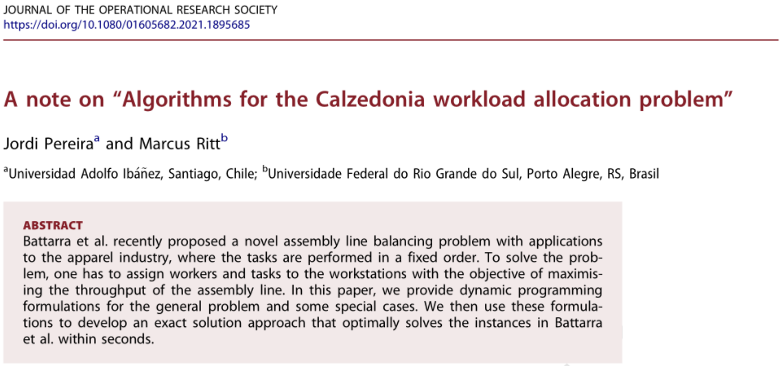
ORIGINAL ARTICLE

Algorithms for the Calzedonia workload allocation problem

Maria Battarra[a], Federico Fraboni[b], Oliver Thomasson[a], Güneş Erdoğan[a*], Gilbert Laporte[c] and Marco Formentini[d]

[a]School of Management, University of Bath, Bath, UK; [b]Calzedonia, Villafranca di Verona, Italy; [c]HEC Montréal, Chemin de la Côte-Sainte-Catherine, Montréal, Canada; [d]Audencia Business School, Nantes, France

ABSTRACT
The Workload Allocation Problem consists of assigning a sequence of $|S|$ operations to workers. The order of these operations is fixed. Each operation consists of a batch of B units, hence a total of $|J|$ jobs have to be performed. Each worker is assigned to an ordered subset of consecutive jobs. Workers have different skills, and therefore jobs take a variable time to process, depending on the assigned worker. The study of this problem is rooted in the operations of Calzedonia. In this paper, we briefly introduce the application before presenting algorithms for solving the problem exactly and heuristically. Our computational results compare the performance of a stand-alone mathematical formulation solved by CPLEX, a sequential exact algorithm, and a metaheuristic, with a simple heuristic implemented in the company.



JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
https://doi.org/10.1080/01605682.2021.1895685

A note on “Algorithms for the Calzedonia workload allocation problem”

Jordi Pereira[a] and Marcus Ritt[b]

[a]Universidad Adolfo Ibáñez, Santiago, Chile; [b]Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil

ABSTRACT
Battarra et al. recently proposed a novel assembly line balancing problem with applications to the apparel industry, where the tasks are performed in a fixed order. To solve the problem, one has to assign workers and tasks to the workstations with the objective of maximising the throughput of the assembly line. In this paper, we provide dynamic programming formulations for the general problem and some special cases. We then use these formulations to develop an exact solution approach that optimally solves the instances in Battarra et al. within seconds.