

Maximizing stability of assembly line balancing schedules under uncertain task duration

O. Battaïa¹, A. Dolgui², E. Gurevsky², A. Pirogov²
A. Rasamimanana², A. Rossi³, R. Shibusaki⁴

¹KEDGE Business School, Bordeaux, France

²LS2N, Nantes Université, Nantes, France

³LAMSADE, Paris Dauphine – PSL, Paris, France

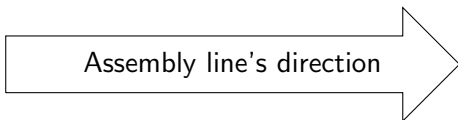
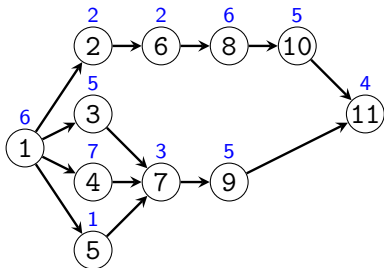
⁴MIS, Université de Picardie Jules Verne, Amiens, France

April 3, 2024



Introduction

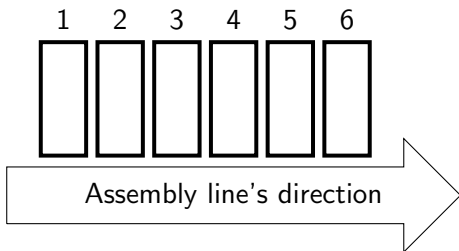
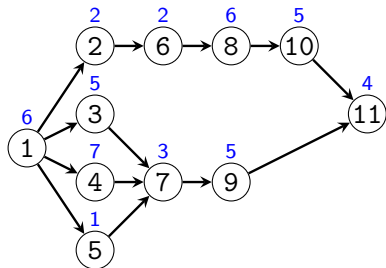
- A set V of n tasks with a duration $t_j \geq 0$ for all $j \in V$
- An acyclic precedence graph $G = (V, A)$
- A set of m workstations, with a cycle time C



- Example: $m = 6$ workstations and $C = 10.5$ units of time.

Introduction

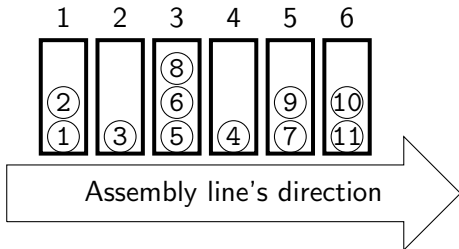
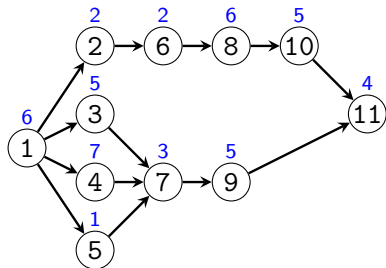
- A set V of n tasks with a duration $t_j \geq 0$ for all $j \in V$
- An acyclic precedence graph $G = (V, A)$
- A set of m workstations, with a cycle time C



- Example: $m = 6$ workstations and $C = 10.5$ units of time.

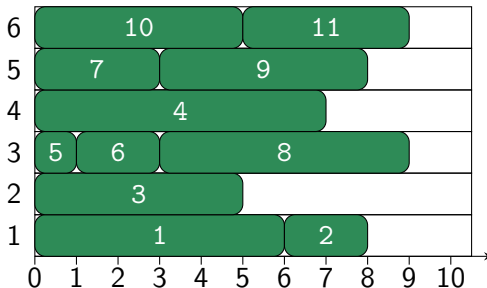
Introduction

- A set V of n tasks with a duration $t_j \geq 0$ for all $j \in V$
- An acyclic precedence graph $G = (V, A)$
- A set of m workstations, with a cycle time C



- Example: $m = 6$ workstations and $C = 10.5$ units of time.

- We can represent this feasible solution using a Gantt chart



- This solution is feasible, because
 - Each task is processed by a workstation
 - Precedence constraints are satisfied
 - The cycle time constraint is satisfied


For a given set of tasks and precedence constraints, there are two classical variants of SALBP:

SALBP-1

SALBP-1 is to minimize the number of workstations, for a given cycle time (close to the bin packing problem)

SALBP-2

SALBP-2 is to minimize the cycle time for a given number of workstations (close to $P_m | prec | C_{\max}$)

 A. Scholl, Balancing and Sequencing of Assembly Lines, second ed., Physica-Verlag Heidelberg, 1999.

Task processing times are not exactly known at the preliminary design stage of the line and only their nominal (or estimated) values are available. This may be caused by the following practical factors:

- For manual assembly lines, the performance of operators implementing tasks, depends on their work rate, skill level, fatigue and motivation
- Product specifications as well as workstation characteristics may be changed during the line life cycle. Such changes may be motivated by a customer demand or updating the market of materials
- Various delays and micro-stoppages when tasks are executed.

Let $\tilde{V} \subseteq V$ be the given subset of tasks whose duration is *uncertain*. We use the following notations:

- $t = [t_1, t_2, \dots, t_n]^T \in \mathbb{R}_+^n$ nominal task times
- $\Omega = \{\xi \in \mathbb{R}^n, \xi_j = 0, j \in V \setminus \tilde{V}\}$ set of processing time deviation

For a given solution to SALBP, what is the minimum task duration increase that compromises feasibility?

- $F(t)$ set of feasible solutions to SALBP w.r.t vector $t \in \mathbb{R}_+^n$

Since any decrease of task processing time cannot compromise solution feasibility, it is sufficient to consider only non-negative task time deviations, *i.e.*, $\xi_j \geq 0$ for all $j \in V$.

Stability radius of a feasible solution $s \in F(t)$

$$\rho(s, t) = \max\{\epsilon \geq 0, s \in F(t + \xi) \forall \xi \in B(\epsilon)\}$$

Where

Stability ball

$$B(\epsilon) = \{\xi \in \Omega, \|\xi\| \leq \epsilon\}$$

Hence, $\rho(s, t)$ is determined as the value of the radius of the greatest closed ball $B(\cdot)$, called *stability ball*, representing the deviations of the uncertain task nominal processing times, for which s remains feasible.

The norm used in the definition of $B(\epsilon)$ plays an important role on the type of uncertainty that affects the tasks duration.

Stability radius of a solution to SALBP

- l_1 -norm: $B_1(\epsilon) = \{\xi \in \Omega, \sum_{j \in \tilde{V}} |\xi_j| \leq \epsilon\}$


Stability radius using the l_1 -norm

$$\rho_1(s, t) = \max\{\epsilon \geq 0, s \in F(t + \xi) \forall \xi \in B_1(\epsilon)\}$$

- l_∞ -norm: $B_\infty(\epsilon) = \{\xi \in \Omega, \max_{j \in \tilde{V}} |\xi_j| \leq \epsilon\}$

Stability radius using the l_∞ -norm

$$\rho_\infty(s, t) = \max\{\epsilon \geq 0, s \in F(t + \xi) \forall \xi \in B_\infty(\epsilon)\}$$

 Y. Sotskov, A. Dolgui, M.-C. Portmann, *Stability analysis of an optimal balance for an assembly line with fixed cycle time*, European Journal of Operational Research, 168 (3) (2006) 783–797.

Property 1

For any solution $s \in F(t)$ and $\xi \in \Omega$ such that $\xi_j = \rho_\infty(s, t)$ for all $j \in \tilde{V}$, we have $s \in F(t + \xi)$

This means that even when all the uncertain tasks have their nominal duration increased by ρ_∞ , the solution remains feasible.

Property 2

The inequality $\rho_\infty(s, t) \leq \rho_1(s, t)$ holds for any $s \in F(t)$

Indeed, we have $B_1(\epsilon) \subseteq B_\infty(\epsilon)$ for any $\epsilon \geq 0$.

Illustration of the stability radius for SALBP

We assume that all the tasks are uncertain, *i.e.*, $\tilde{V} = V$.

- Measure of the stability radius of this solution in the ℓ_1 -norm

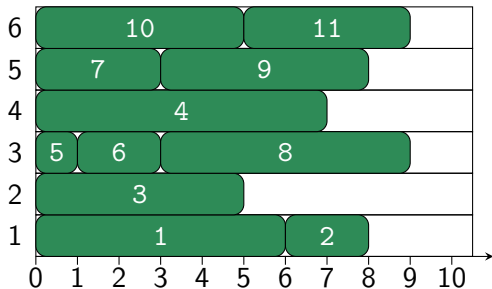
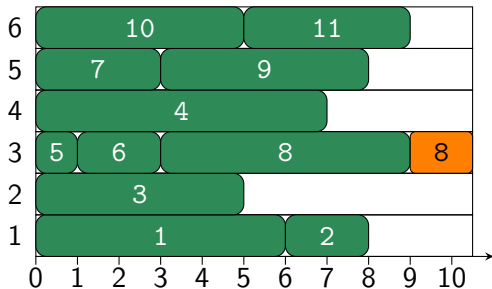


Illustration of the stability radius for SALBP

We assume that all the tasks are uncertain, *i.e.*, $\tilde{V} = V$.

- Measure of the stability radius of this solution in the ℓ_1 -norm



- $\rho_1 = 1.5$

Illustration of the stability radius for SALBP

We assume that all the tasks are uncertain, *i.e.*, $\tilde{V} = V$.

- Measure of the stability radius of this solution in the ℓ_∞ -norm

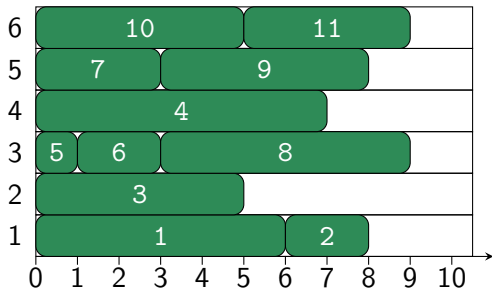
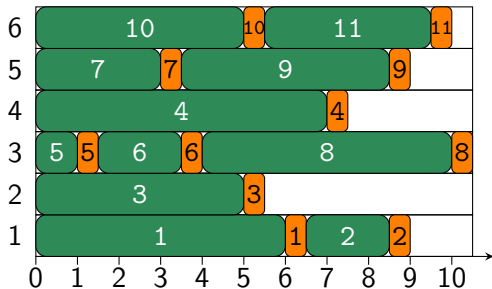


Illustration of the stability radius for SALBP

We assume that all the tasks are uncertain, *i.e.*, $\tilde{V} = V$.

- Measure of the stability radius of this solution in the ℓ_∞ -norm



- $\rho_\infty = 0.5$

Stability radius computation

- V_k^s set of tasks assigned to the workstation k in solution s
- $\widetilde{W}^s = \{k \in W, \widetilde{V}_k^s \neq \emptyset\}$ and $\widetilde{V}_k^s = V_k^s \cap \widetilde{V}$

Computing the stability radius in ℓ_1 -norm for $s \in F(t)$

$$\rho_1(s, t) = \min_{k \in \widetilde{W}^s} \left(C - \sum_{j \in V_k^s} t_j \right)$$

Computing the stability radius in ℓ_∞ -norm for $s \in F(t)$

$$\rho_\infty(s, t) = \min_{k \in \widetilde{W}^s} \frac{C - \sum_{j \in V_k^s} t_j}{|\widetilde{V}_k^s|}$$

Maximizing the stability radius

The problem of building a solution to SALBP that maximizes ρ_1 (resp. ρ_∞), the stability radius in the ℓ_1 -norm (resp. the ℓ_∞ -norm) is denoted by P_1 (resp. P_∞). Both are NP-hard, even when $V = \tilde{V}$ (by reduction to the bin packing problem).

Decision variables

- $\rho_1 \geq 0$ stability radius value, to be maximized (P_1 only)
- $\rho_\infty \geq 0$ stability radius value, to be maximized (P_∞ only)
- $x_{j,k}$ is 1 if task $j \in V$ is allocated workstation $k \in W$, 0 otherwise
- a_k is 1 if workstation $k \in W$ processes at least one uncertain task (P_1 only)
- $\xi_{j,k} \geq 0$ duration increase of task $j \in \tilde{V}$ if allocated to $k \in W$ (P_∞ only)

MILP formulation of P_1

$$\begin{aligned} & \text{Maximize } \rho_1 \\ & \sum_{k \in W} x_{j,k} = 1 \quad \forall j \in V \\ & \sum_{j \in V} t_j x_{j,k} \leq C \quad \forall k \in W \\ & \sum_{q=k}^m x_{i,q} \leq \sum_{q=k}^m x_{j,q} \quad \forall (i,j) \in A \\ & x_{j,k} \leq a_k \quad \forall (j,k) \in \tilde{V} \times W \\ & \rho_1 \leq C(2 - a_k) - \sum_{j \in V} t_j x_{j,k} \quad \forall k \in W \\ & \rho_1 \geq 0 \\ & a_k \in \{0, 1\} \quad \forall k \in W \\ & x_{j,k} \in \{0, 1\} \quad \forall (j,k) \in V \times W \end{aligned}$$


MILP formulation of P_∞

$$\begin{aligned} & \text{Maximize} && \rho_\infty \\ & \sum_{k \in W} x_{j,k} = 1 && \forall j \in V \\ & \xi_{j,k} \leq UB_\infty x_{j,k} && \forall (j,k) \in V \times W \\ & \rho_\infty = \sum_{k \in W} \xi_{j,k} && \forall j \in V \\ & \sum_{j \in V} t_j x_{j,k} + \sum_{j \in \tilde{V}} \xi_{j,k} \leq C && \forall k \in W \\ & \sum_{q=k}^m x_{i,q} \leq \sum_{q=k}^m x_{j,q} && \forall (i,j) \in A \\ & \rho_\infty \geq 0 \\ & \xi_{j,k} \geq 0 && \forall (j,k) \in V \times W \\ & x_{j,k} \in \{0,1\} && \forall (j,k) \in V \times W \end{aligned}$$

Allocation intervals

Precedence constraints induce allocation intervals for the tasks: task $j \in V$ can be allocated to a workstation $k \in Q_j$ where

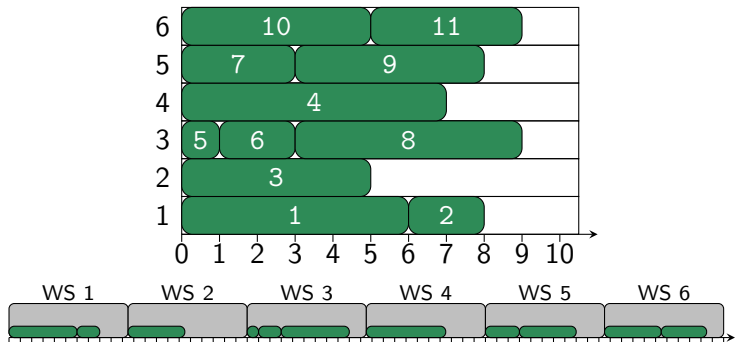
$$Q_j = [l_j, u_j] = \left[\left[\frac{t_j + \sum_{i \in P(j)} t_i}{C} \right], m + 1 - \left[\frac{t_j + \sum_{i \in S(j)} t_i}{C} \right] \right]$$


 J. Patterson, J. Albracht, Assembly-line balancing: zero-one programming with Fibonacci search, *Operations Research* 23 (1) (1975) 166–172.

Hence, we set $x_{j,k} = 0$ for all $j \in V, \forall k \in W \setminus Q_j$ in P_1 and P_∞ .

Tightening the allocation intervals

SALBP can be relaxed to a single-machine scheduling problem, as proposed in Scholl & Becker (2006), where the machine has a working time of $m \times C$.



 A. Scholl & C. Becker (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168 (3), 666–693.

Tightening the allocation intervals

Notations

For all task $i \in V$, we denote:

- τ_i a lower bound on the earliest starting time of task i
- ζ_i an upper bound on the latest completion time of task i

Step 1 – initial value for τ_i

We solve $1|r_j|C_{\max}$ on the tasks $j \in P(i)$ to compute τ_i , where r_j is set to the earliest starting time of j , so τ_i are computed in topological order of the tasks i .

Step 2 – first improvement of τ_i

If $\tau_i > C \cdot \lfloor \frac{\tau_i}{C} \rfloor$, then $l_i = 1 + \lfloor \frac{\tau_i}{C} \rfloor$, and at least one direct predecessor of i is allocated to l_i , so τ_i can be improved to $\max(\tau_i, C \cdot \lfloor \frac{\tau_i}{C} \rfloor + \min_{j \in \bar{P}(i)} t_j)$

Step 3 – second improvement of τ_i


Let $q = \lfloor \frac{\tau_i}{C} \rfloor$: if task i starts at time τ_i , then it is processed by workstation $q + 1$. Now, if $\lceil \frac{1}{C}(\tau_i + t_i) \rceil > q + 1$, then i cannot be allocated to $q + 1$ because SALBP is not preemptive. Hence we improve τ_i to $(q + 1) \cdot C$.

Improving ζ_i

We perform the same steps to improve ζ_i , by considering the reversed precedence graph.

Addition of precedence constraints (Fleszar & Hindi (2003))

Let $i \in V$, if there exists $j \in V \setminus P(i)$ such that $\zeta_j - t_j < \tau_i + t_i$, then task j must complete before task i can start. Hence arc (j, i) can be added to the precedence graph.

 Fleszar & Hindi (2003). An enumerative heuristic and reduction methods for the assembly line balancing problem. *European Journal of Operational Research*, 145 (3), 606–620.

$UB_1^1 = C - \max_{j \in \tilde{V}} t_j$ is a natural upper bound on ρ_1 . We introduce

another one, denoted by UB_1^2 , by allowing preemption. In the best case, all the certain tasks are processed by the same workstations.

The maximum number of workstations that process certain tasks

for C units of time is $\chi = \left\lfloor \frac{\sum_{j \in V \setminus \tilde{V}} t_j}{C} \right\rfloor$.

The remaining $m - \chi$ workstations should process a load of

$\sum_{j \in V \setminus \tilde{V}} t_j - C \cdot \chi + \sum_{j \in \tilde{V}} t_j$ units of time.

The remaining $m - \chi$ workstations should process a load of $\sum_{j \in V \setminus \tilde{V}} t_j - C \cdot \chi + \sum_{j \in \tilde{V}} t_j$ units of time. Two cases may occur:

- Either the $m - \chi$ workstations share the same load which is

$$\frac{\sum_{j \in V} t_j - C \cdot \chi}{m - \chi}$$

- Or a single workstation processes all the remaining certain tasks, letting $m - \chi - 1$ workstations for processing the uncertain tasks with a common load, hence

$$UB_1^2 = C - \min \left(\frac{\sum_{j \in V} t_j - C \cdot \chi}{m - \chi}, \frac{\sum_{j \in \tilde{V}} t_j}{m - \chi - 1} \right)$$

The MILP formulation of P_∞ requires an upper bound on ρ_∞ :

$$\xi_{j,k} \leq UB_\infty x_{j,k} \quad \forall (j,k) \in V \times W$$

UB_∞ can of course be set to $\min(UB_1^1, UB_1^2)$, but the tightest it is, the better.

Bisection method to improve UB_∞

Input

- UB_∞ , any upper bound on ρ_∞ , like $\min(UB_1^1, UB_1^2)$
- LB_∞ , any lower bound on ρ_∞ (0, or on found by a heuristic)

Output: UB_∞ , a (possibly better) upper bound on ρ_∞ .

The bisection method is based on the allocation intervals.

Bisection method to improve UB_∞

Set $L \leftarrow LB_\infty$, $U \leftarrow UB_\infty$ and $\varepsilon \leftarrow 10^{-3}$

while $U - L > \varepsilon$ **do**

 set $\Delta \leftarrow \frac{1}{2}(L + U)$

for each $j \in \tilde{V}$ **do**

 set $t_j \leftarrow t_j + \Delta$

for each $j \in V$ **do**

 compute the allocation interval Q_j

if $l_j \leq u_j$ for all $j \in V$ **then**

 set $L \leftarrow \Delta$

else

 set $U \leftarrow \Delta$

for each $j \in \tilde{V}$ **do**


 restore $t_j \leftarrow t_j - \Delta$

return $UB_\infty \leftarrow U$

Starting from the MILP formulation of P_∞ , we perform a Dantzig-Wolfe decomposition:

Reformulation and Column generation

- Extreme points (Columns) are workstation patterns
- Initialize columns with a feasible solution (heuristically)
- Pricing problem:
 - Define a task assignment maximizing the sum of reduced costs
 - Decomposition: one pricing problem per workstation

 R.S. Shibasaki, E. Gurevsky, A. Rossi, *A new upper bound based on Dantzig-Wolfe decomposition to maximize the stability radius of a simple assembly line under uncertainty*, European Journal of Operational Research, 313 (3), pp.1015–1030, 2024.

- Let S denote the bounded set of integer points defined by the constraints

$$\begin{aligned} \xi_{j,k} &\leq UB_\infty x_{j,k} && \forall (j, k) \in V \times W \\ \sum_{j \in V} t_j x_{j,k} + \sum_{j \in \tilde{V}} \xi_{j,k} &\leq C && \forall k \in W \\ \xi_{j,k} &\geq 0 && \forall (j, k) \in V \times W \\ x_{j,k} &\in \{0, 1\} && \forall (j, k) \in V \times W \end{aligned}$$

- S can be decomposed by workstation such that $S = S_1 \times \dots \times S_m$
- Since x is binary, S coincides with B , which is the set of extreme points of the convex hull of S , denoted by $\text{conv}(S)$

A *pattern* is a subset of tasks that can be allocated to a given workstation. A solution to SALBP is an element of $B = B_1 \times \cdots \times B_m$ where each task appears in exactly one pattern B_k .

- $\lambda_k^{(r)}$ is related to the extreme point $r \in B_k$, assuming value 1 if r is selected for the workstation $k \in W$, and value 0 otherwise
- $\bar{x}_{j,k}^{(r)}$ corresponds to variable $x_{j,k}$ at the point $r \in B$
- $\bar{\xi}_{j,k}^{(r)}$ corresponds to variable $\xi_{j,k}$ at the point $r \in B$

Maximize ρ_∞

$$\sum_{r \in B_k} \lambda_k^{(r)} = 1, \quad \forall k \in W \quad [\mu_k]$$

$$\sum_{k \in Q_j} \sum_{r \in B_k} \bar{x}_{j,k}^{(r)} \cdot \lambda_k^{(r)} = 1, \quad \forall j \in V \quad [\kappa_j]$$

$$\rho_\infty \leq \sum_{k \in Q_j} \sum_{r \in B_k} \bar{\xi}_{j,k}^{(r)} \cdot \lambda_k^{(r)}, \quad \forall j \in V \quad [\omega_j]$$

$$\sum_{q=k}^{u_i} \sum_{r \in B_q} \bar{x}_{i,q}^{(r)} \cdot \lambda_q^{(r)} \leq \sum_{q=k}^{u_j} \sum_{r \in B_q} \bar{x}_{j,q}^{(r)} \cdot \lambda_q^{(r)}, \quad \forall (i,j) \in A, \forall k \in Q_i \cap Q_j \quad [\pi_{i,j,k}]$$

$$\rho_\infty \geq 0$$

$$\lambda_k^{(r)} \geq 0, \quad \forall r \in B_k, \forall k \in W$$

Valid Inequality for the master problem

- Let $\tilde{B}_k \subseteq B_k$ be the subset of uncertain patterns of workstation $k \in W$

The following inequality is valid for the DW reformulated problem and increases the lower bound value.

Valid inequality for the master problem

$$\rho_\infty \leq \sum_{r \in \tilde{B}_k} (\rho_k^{(r)} - UB_\infty) \cdot \lambda_k^{(r)} + UB_\infty, \quad \forall k \in W \quad [\gamma_k]$$

- Since at most one pattern is assigned per workstation, these inequalities require that ρ_∞ is less than or equal to the local stability radius of the uncertain pattern assigned to workstation k . If no uncertain pattern has been assigned, then the inequality becomes loose ($\rho_\infty \leq UB_\infty$)

$$\text{Maximize } \sum_{j \in V_k} \omega_j \cdot \xi_j^k + \gamma_k \cdot \rho_k - rc_j^k \cdot x_j^k$$

$$\xi_{j,k} \leq UB_\infty \cdot x_{j,k}, \quad \forall j \in V_k$$

$$\sum_{j \in V_k} t_j \cdot x_{j,k} + \sum_{j \in \tilde{V}_k} \xi_{j,k} \leq C$$

$$\xi_{j,k} \leq \rho_k, \quad \forall j \in V_k$$

$$\xi_{j,k} \geq \rho_k - UB_\infty \cdot (1 - x_{j,k}), \quad \forall j \in \tilde{V}_k$$

$$\xi_{j,k} \geq 0, x_{j,k} \in \{0, 1\} \quad \forall j \in V_k$$

$$\rho_k \geq 0$$


$$\text{Where } rc_j^k = \kappa_j + \sum_{\substack{i \in V_k \\ (j,i) \in A \\ l_i \leq k \leq u_j}} \sum_{q=l_i}^k \pi_{j,i,q} - \sum_{\substack{i \in V_k \\ (i,j) \in A \\ l_j \leq k \leq u_i}} \sum_{q=l_j}^{\min(k, u_i)} \pi_{i,j,q}$$

As in Peeters & Degraeve (2006), we reinforce the precedence relations by adding the following valid inequalities to each pricing subproblem:

Valid inequality for the pricing subproblem

$$x_{i,k} + x_{l,k} \leq x_{j,k} + 1 \quad \forall (i,j) \in A, \forall l \in S(j) \cap V_k$$

Finally, if PS_k has a feasible solution, whose objective value is p_k and $p_k - \mu_k - UB_\infty \cdot \gamma_k > 0$, then this solution is added as a new column to the master problem.

 M. Peeters & Z. Degraeve (2006). An linear programming based lower bound for the simple assembly line balancing problem. *European Journal of Operational Research*, 168 (3), 716–731.

We use a multi-start, greedy heuristic, that considers the workstations in order, and tries to allocate a task j such that:

Heuristic

- All the predecessors of j have been scheduled
- The cycle time constraint is satisfied
- The selection of j does not cause the stability radius to be less than LB , the current best lower bound
- The current workstation belongs to the allocation interval Q_j

If a feasible solution is obtained, LB is updated, otherwise, a new attempt is made, and the heuristic algorithm stops after a predefined number of trials.

We also run this heuristic backward by reversing the precedence graph.

Numerical results for problem P_{∞}

Group	PA			NAI			
	Fixed Vars		CG Time (s)	Fixed Vars		CG Time (s)	+Arcs
	No LB	With LB		No LB	With LB		
BN_2.BM	379	385	300.0	387	393	300.0	0.0
BN_2.PB	210	217	300.0	211	218	300.0	0.0
BN_2.PM	974	974	189.7	1052	1052	181.1	0.0
BN_6.BM	1258	1271	300.0	1260	1273	300.0	0.0
BN_6.PB	746	773	278.0	746	773	278.3	0.0
BN_6.PM	2933	2934	77.6	2977	2978	76.5	0.0
CH_2.BM	386	390	300.0	395	399	300.0	0.0
CH_2.PB	208	214	300.0	209	215	300.0	0.0
CH_2.PM	968	968	192.6	1052	1052	182.5	0.0
CH_6.BM	1272	1280	282.8	1281	1288	281.9	0.3
CH_6.PB	728	753	273.6	728	753	271.7	0.0
CH_6.PM	2975	297	96.2	3054	3055	90.1	0.0
MX_2.BM	393	398	300.0	398	403	300.0	0.0
MX_2.PB	210	217	300.0	210	217	300.0	0.0
MX_2.PM	979	979	184.8	1039	1039	175.7	0.0
MX_6.BM	1271	1284	288.2	1273	1286	287.5	0.1
MX_6.PB	756	780	284.2	756	780	283.3	0.0
MX_6.PM	2940	2941	88.6	2978	2979	83.1	0.0
MX_9.BM	1955	1996	4.2	1983	2022	4.6	0.6
MX_9.PB	1154	1198	0.8	116	1205	0.9	1.2
MX_9.PM	4468	4483	40.7	4627	4648	33.4	0.0
Overall Average	1293	1305	208.7	1323	1335	206.2	0.1

Table: Average number of binary variables pre-fixed to zero and the impact of assignment intervals on CG

Comparison with Compact Formulation

- CPLEX: UB obtained after 300s ($UB_{compact}$)
- Dantzig-Wolfe: UB obtained after 300s (UB_{DW})

$$\text{DEV UB} = (UB_{DW} - UB_{compact}) / UB_{DW}$$

Group	DEV UB (%)	Time (s)
BN_2_PM	-65.34	181
BN_6_PM	-64.42	77
CH_2_PM	-66.84	182
CH_6_PM	-65.36	90
MX_2_PM	-70.48	176
MX_6_PM	-63.05	83
MX_9_PM	-52.57	33
Overall Average	-9.25	298

Table: UB improvement and computational times resulting from DW over compact formulation

Limitations of the stability radius

- With the stability radius in the ℓ_1 -norm, the whole extra-duration may affect a single task
- With the stability radius in the ℓ_∞ -norm, the maximum duration increase is the same for very short and very long tasks

The **stability factor** has been introduced Gurevsky *et. al.* to circumvent these drawbacks. It is the maximum **rate** of increment of the nominal processing time applied for any uncertain task without compromising the feasibility of the corresponding line configuration.


 E. Gurevsky, A. Rasamimanana, A. Pirogov, A. Dolgui, A. Rossi, Stability factor for robust balancing of simple assembly lines under uncertainty, *Discrete Applied Mathematics*, 318, pp.113–132, 2022.

Illustration of the stability factor for SALBP

We assume that all the tasks are uncertain, *i.e.*, $\tilde{V} = V$.

- Measure of the stability factor of this solution

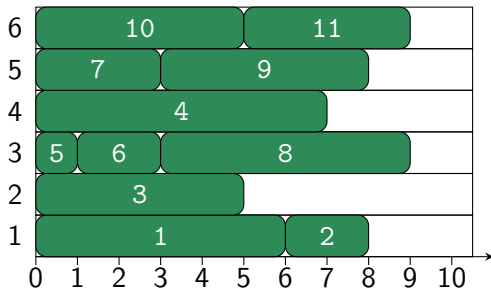
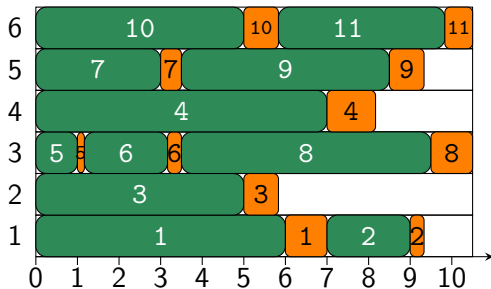


Illustration of the stability factor for SALBP

We assume that all the tasks are uncertain, *i.e.*, $\tilde{V} = V$.

- Measure of the stability factor of this solution



- $f = \frac{1}{6} \approx 0.167$

Stability factor of a feasible solution $s \in F(t)$

$$f(s, t) = \max\{\epsilon \geq 0, s \in F(t + \xi) \forall \xi \in H(\epsilon, t)\}$$

Where

Stability hyperrectangle

$$H(\epsilon, t) = \{\xi \in \Omega, \xi_j \leq \epsilon \cdot t_j, \forall j \in \tilde{V}\}$$

Hence, $f(s, t)$ is defined as the proportionality factor defining the greatest closed n -dimensional hyperrectangle $H(\cdot)$, called *stability hyperrectangle*, representing the deviations of the uncertain task nominal processing times, for which s remains feasible.

Computing the stability factor for $s \in F(t)$

$$f(s, t) = \min_{k \in \tilde{W}^s} \frac{C - \sum_{j \in V_k^s} t_j}{\sum_{j \in \tilde{V}_k^s} t_j}$$

The MILP model of P_f is very similar to the one of P_∞ , but

$$\sum_{j \in V} t_j x_{j,k} + \sum_{j \in \tilde{V}} \xi_{j,k} \leq C \quad \forall k \in W$$

becomes

$$\sum_{j \in V} t_j x_{j,k} + \sum_{j \in \tilde{V}} t_j \cdot \xi_{j,k} \leq C \quad \forall k \in W$$

Now, $\xi_{j,k}$ is a factor, not a time.

Maximizing the stability factor – upper bounds on f

UB_1^a – First upper bound on f

$$UB_1^a = \frac{C - \lambda_1^a}{\lambda_1^a}$$

where $\lambda_1^a = \max_{j \in \tilde{V}} t_j$.

Moreover, the total amount of work that the workstations can carry out is upper-bounded by $m \cdot C$. This can be written as $(1 + f) \cdot \sum_{j \in \tilde{V}} t_j + \sum_{j \in V \setminus \tilde{V}} t_j \leq m \cdot C$

UB_1^b – Second upper bound on f

$$UB_1^b = \frac{m \cdot C - \sum_{j \in V \setminus \tilde{V}} t_j}{\sum_{j \in \tilde{V}} t_j - 1}$$

UB_1^c and UB_1^d are based on the pigeonhole principle. There exists at least one workstation that processes at least $\gamma_1^c = \left\lceil \frac{|\tilde{V}|}{m} \right\rceil$ uncertain tasks. Let π be a permutation of the set \tilde{V} such that $t_{\pi_1} \leq \dots \leq t_{\pi_{|\tilde{V}|}}$. It can then be deduced that the total processing time due to uncertain tasks in this workstation is at least λ_1^c where

$$\lambda_1^c = \sum_{q=1}^{\gamma_1^c} t_{\pi_q}$$

UB_1^c – Third upper bound on f

$$UB_1^c = \frac{C - \lambda_1^c}{\lambda_1^c}$$

Maximizing the stability factor – upper bounds on f

At most $\left\lfloor \sum_{j \in V \setminus \tilde{V}} t_j / C \right\rfloor$ workstations have a load equal to C due to certain tasks only, which leaves at least $m - \left\lfloor \sum_{j \in V \setminus \tilde{V}} t_j / C \right\rfloor$ workstations to process uncertain tasks, and by the pigeonhole principle, there exists a workstation that processes at least $\gamma_1^d = \left\lceil |\tilde{V}| / \left(m - \left\lfloor \sum_{j \in V \setminus \tilde{V}} t_j / C \right\rfloor \right) \right\rceil \geq \gamma_1^c$ uncertain tasks. As a result, there exists a workstation that processes at least γ_1^d uncertain tasks whose total load is at least

$$\lambda_1^d = \sum_{q=1}^{\gamma_1^d} t_{\pi_q}$$

UB_1^d – Forth upper bound on f

$$UB_1^d = \frac{C - \lambda_1^d}{\lambda_1^d}$$

Maximizing the stability factor vs. the stability radius

Non-Equivalence result #1

Problems P_∞ and P_f are not equivalent

$n = 5$, $m = 2$, $t = [1, 1, 1, 1, 4]$, $\tilde{V} = V$, $C = 8$, no precedence constraints.

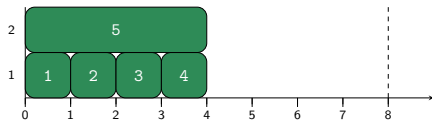


Figure: The solution \mathcal{S}_1 , for which $f(\mathcal{S}_1) = 1$ and $\rho_1(\mathcal{S}_1) = 1$.

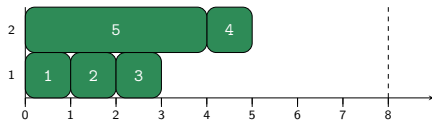


Figure: The solution \mathcal{S}_2 , for which $f(\mathcal{S}_2) = 0.6$ and $\rho_\infty(\mathcal{S}_2) = 1.5$.

Solution \mathcal{S}_1 is optimal for P_f , as $f(\mathcal{S}_1) = 1$ reaches the upper bound UB_1^a . Solution \mathcal{S}_2 has a better stability radius value in the ℓ_∞ -norm

Maximizing the stability factor vs. the stability radius

Non-Equivalence result #2

Problems P_1 and P_f are not equivalent

$n = 5$, $m = 2$, $t = [1, 1, 1, 1, 4]$, $\tilde{V} = \{1, \dots, 4\}$, $C = 8$, no precedence constraints.

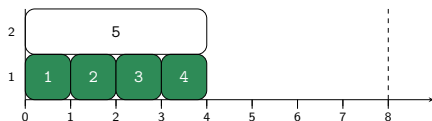


Figure: The solution \mathcal{S}_1 , for which $f(\mathcal{S}_1) = 1$ and $\rho_1(\mathcal{S}_1) = 4$.

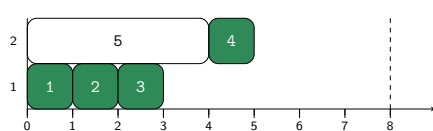


Figure: The solution \mathcal{S}_2 , for which $f(\mathcal{S}_2) = \frac{5}{3}$ and $\rho_1(\mathcal{S}_2) = 3$.

Solution \mathcal{S}_1 is optimal for P_1 , but solution \mathcal{S}_2 has a better stability factor value.

Maximizing the stability factor vs. the stability radius

Equivalence result

Problems P_1 and P_f are equivalent when $\tilde{V} = V$

Let $j^- \in \tilde{V}$ (resp. j^+) be an uncertain task such that $t_{j^-} = \min_{j \in \tilde{V}} t_j$ (resp. $t_{j^+} = \max_{j \in \tilde{V}} t_j$)

Property

If the stability factor of a given feasible solution \mathcal{S} is $f(\mathcal{S})$, then its stability radius in the ℓ_∞ -norm is at least $t_{j^-} \cdot f(\mathcal{S})$.

Corollary

If the stability radius in the ℓ_∞ -norm of a given feasible solution \mathcal{S} is $\rho_\infty(\mathcal{S})$, then its stability factor is at least $\frac{\rho_\infty(\mathcal{S})}{t_{j^+}}$.

We use a set of 25 instances with up to 297 tasks and 60 workstations.

$ \tilde{V} $	#OPT	Avg. CPU	Avg. gap
$\lceil \frac{n}{4} \rceil$	23	50	0.4%
$\lceil \frac{n}{2} \rceil$	18	173	5.2%
$\lceil \frac{3n}{4} \rceil$	15	242	7.4%

- The problem gets more difficult when the number of uncertain tasks increases
- It gets more difficult when the number of precedence constraints decreases

Maximizing stability for SALBP

- We formulated three variants of SALBP under uncertainty
- The problem is challenging even for 50 tasks

In terms of methods

- Use the upper bounds to design Branch-and-bound algorithms
- Extend the Dantzig-Wolfe decomposition into a Branch-and-price algorithm

Other problems

- Extensions to incompatible tasks, capacitated variants
- Extensions to Transfer assembly lines balancing problems

Other models of uncertainty

- Alternative models to handle uncertainty besides stability?