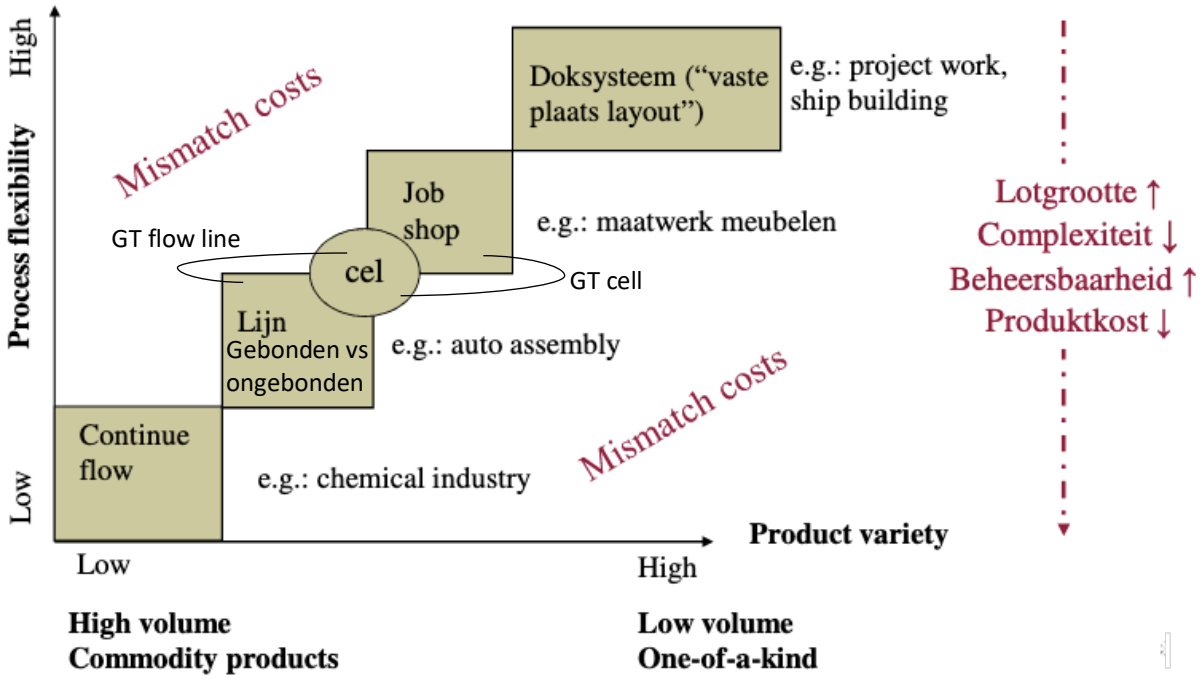
Organisatie van het productieproces

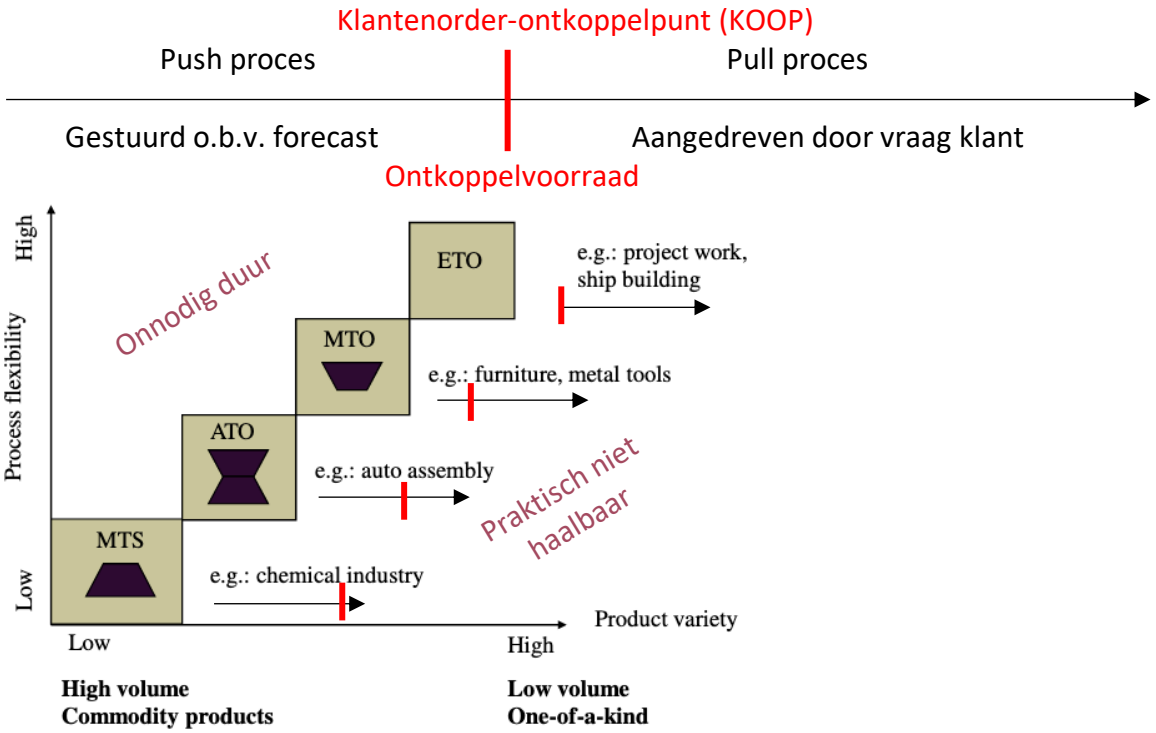
Deel I: Facilities design

= OPTIMALISATIE VAN KOSTEN

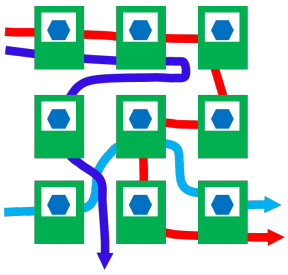
Types lay-outs



Push/pull view – STRATEGISCH



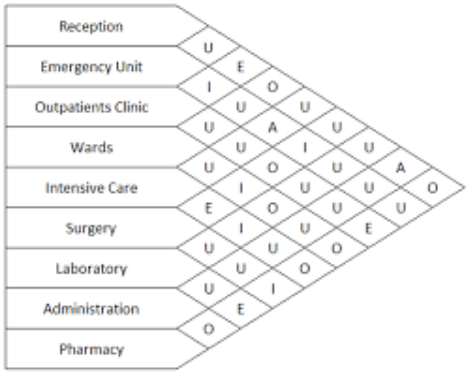
Functionele lay-out/proces lay-out/job shop



- Machines gegroepeerd volgens type bewerking
- Elk product volgt eigen routing

Data

- **Kwantitatieve** flow data
  - From-to matrix
  - Trip between matrix
- **Kwalitatieve** flow data
  - Closeness ratings
    A: Absolutely necessary > E: Especially important > I: Important > O: Ordinary > U: Unimportant > X: Not desirable
    → Vb. numerieke waarden toekennen: A = 6, E = 5, …
    MAAR kan leiden tot verkeerde conclusies!
  - Weergave: relationship chart



Methoden

- A.h.v. kwantitatieve data: **kwantitatieve** methoden
  Optimaal = minimale flow f x afstand d (x kost q)
  - Euclidische afstand
  - Rechthoekige afstand (Manhattan)
  - O.b.v. zwaartepunten → benaderend!!

  $$(x,y) = \frac{\sum x/y\text{-coördinaat zwaartepunt vierkantje}}{\#\text{vierkantjes}}$$

  - Lineair toewijzingsprobleem
    Minimaliseert kost a_{ij} tussen nieuwe facilities ↔ bestaande facilities
    Géén interacties tussen nieuwe facilities!!
    Model

    $$\text{Min. } \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij}$$

    $$\text{Odb } \sum_{i=1}^{m} x_{ij} \leq 1 \quad \forall j$$

    $$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i$$

    $$x_{ij} = \begin{cases} 1 \text{ als facility i wordt toegewezen aan locatie j} \\ 0 \text{ anders} \end{cases}$$

Oplossingsmethode

Hongaarse methode (exact)

- Assumptie: m = n
  → m > n: dummy facilities z met $a_{zj} = 0$
  → m < n: infeasible!
- Werkwijze:
  1. Trek kleinste getal in elke rij van $[a_{ij}]$ af van alle getallen in die rij
  2. Zelfde voor kolommen
  3. Teken lijnen die alle nullen afdekken:
     Indien # lijnen = # locaties → opl.: kies in elke rij/kolom één 0;
     anders → stap 4
  4. Trek kleinste getal in $[a_{ij}]$ dat NIET bedekt is door een lijn af van alle niet-afgedekte getallen en tel het op bij alle dubbel doorstreepte getallen

- Kwadratisch toewijzingsprobleem

Minimaliseert kost a_{ij} tussen nieuwe facilities ↔ bestaande facilities **én interactiekost** c_{ijkl} voor tegelijk toewijzen van facility i aan locatie j en k aan l

Model

Min. $\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}x_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} c_{ijkl}\, x_{ij}x_{kl}$

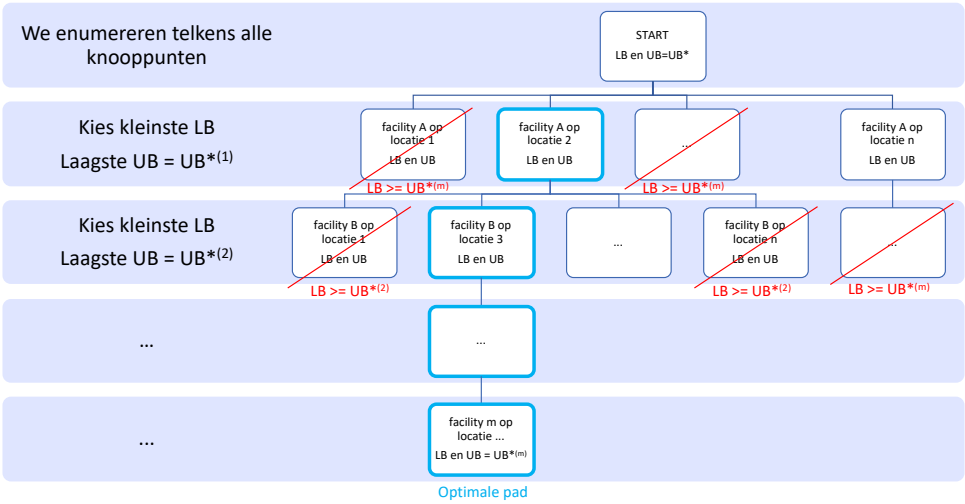
Odb $\sum_{i=1}^{m} x_{ij} \leq 1 \;\; \forall j$

$\sum_{j=1}^{n} x_{ij} = 1 \;\; \forall i$

$x_{ij} = \begin{cases} 1 \text{ als facility i wordt toegewezen aan locatie j} \\ 0 \text{ anders} \end{cases}$

Oplossingsmethoden

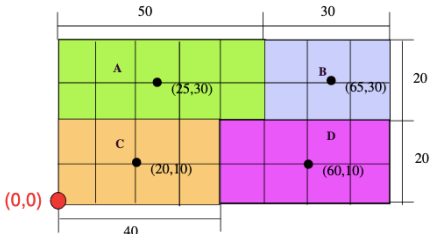
Branch and bound (exact)



- Waarbij LB =
  Interactiekost tussen bestaande facilities
  + kost tussen nieuwe facilities ↔ bestaande facilities } exact
  + **benedengrens** op interactiekost tussen nieuwe facilities
  $[w_{ij}] = [w'_{ij}] + [w''_{ij}]$ met i = te plaatsen facilities en j = nog lege locaties
  → $[f_j] * [d_j] /2$ → grootste flow over kortste afstand
  → rooskleurige inschatting!!
  ➔ Los op via Hongaarse methode!
- En UB = werkelijke kost voor LB oplossing

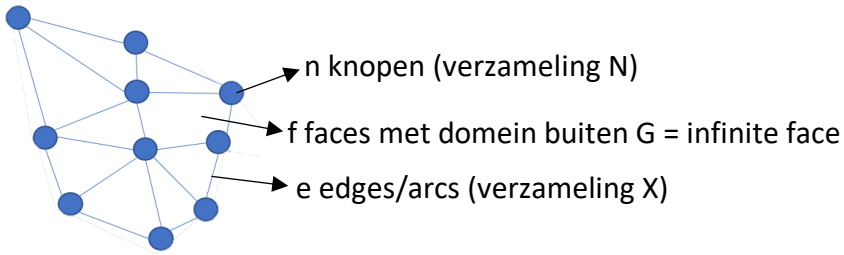
CRAFT (heuristiek)

- Werkwijze:
  1. Startoplossing



  2. Paarsgewijze omwisseling van zwaartepunten i.g.v. zelfde oppervlakte *of* gemeenschappelijke grens
     a. Bereken kost voor omwisseling coördinaten
     b. Omwisseling met laagste kost wordt gekozen
     c. Evt. correctie van kosten!
  3. Herhaal tot geen verbetering meer mogelijk is!
- Nadeel: optimum afhankelijk van startoplossing → lokale optima

- A.h.v. kwantitatieve of kwalitatieve data: **kwalitatieve** methoden
  - Grafentheorie (heuristiek)
    - Graf G(N,X)



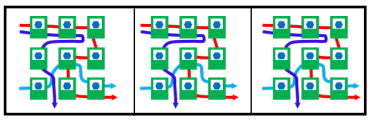
    - **Planaire graf** = graf die getekend kan worden zonder snijdende edges
      Hiervoor geldt: $n - e + f = 2$
    - **Maximaal planaire graf** = graf waarbij geen enkele edge meer kan toegevoegd worden zonder planariteit op te offeren
      Hiervoor geldt: $2e = 3f \Rightarrow e_{max} = 3n - 6$
    - Werkwijze:
      1. Stel maximaal onderspannende boom op
         a. Trek een edge tussen 2 knooppunten zodat gewicht (o.b.v. closeness rating of flow matrix) gemaximaliseerd wordt
         b. Stop wanneer alle knooppunten verbonden zijn, zonder dat er een cylus optreedt! (Voor n knooppunten: n-1 edges)
      2. Greedy algoritme
         a. Voeg edge toe met meeste gewicht
         b. Stop indien e_{max} bereikt is
      3. Stel duale graf op
         a. Plaats een knooppunt in infinite face en voeg nodige edges toe om terug een maximaal planaire graf te bekomen
         b. Plaats een knooppunt in elke face van de planaire graf
         c. Teken bogen tussen de knooppunten zodat ze telkens een gemeenschappelijke grens tussen de faces snijden
      4. Duale graf stelt meerdere feasible lay-outs voor

Group technology/celproductie

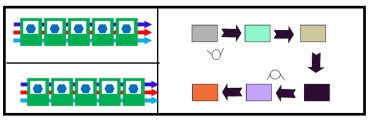
Doel = combinatie van efficiëntie flow shop en flexibiliteit job shop
→ Cross-trained team operatoren!

Job shop ⟶ Flow shop

GT cell (minifabriek, subplant job shop)



GT flow line (JIT cell)
→ Ongebonden lijnen, vaak U-shaped



<u>Cellen en part families identificeren</u>

Startpunt: Part-machine Processing Indicator (**PPI**) **matrix**

$$\begin{array}{c} \\ P1 \\ P2 \\ \dots \end{array} \begin{array}{c} \begin{array}{ccc} M1 & M2 & \dots \end{array} \\ \begin{pmatrix} 1 & 0 & \dots \\ 1 & 1 & \\ \dots & & \end{pmatrix} \end{array}$$

Doel: blok-diagonale matrix

$$\begin{array}{ll} & \begin{array}{cccc} & \text{Cel 1} & & \text{Cel x} \\ M5 & M1 & \dots & M3 \quad M2 \end{array} \\ \begin{array}{l} \text{Product-familie 1} \begin{cases} P1 \\ P3 \end{cases} \\ \dots \\ \text{Product-familie y} \begin{cases} P6 \\ P4 \end{cases} \end{array} & \begin{pmatrix} 1 & 1 & \dots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \dots & & & & \\ 0 & 0 & & 1 & 1 \\ 0 & 0 & & 1 & 0 \end{pmatrix} \end{array}$$

Ideaal: geen material flow tussen cellen
Maar niet altijd haalbaar → bottleneck part/machine

Clustering algoritmes:

- Row-and-columns masking (R&CM) algoritme
  1. Kies een rij die niet bedekt is door een lijn: trek hier horizontaal een lijn door
  2. Zolang er een 1 voorkomt die slechts door één lijn bedekt wordt:
     Kies zo'n 1 en trek een lijn loodrecht op bestaande lijn
  3. Productfamilie gevonden!
- Rank-order clustering (ROC) algoritme
  Stel # machines = m en # parts = n
  1. Ken **binair gewicht** $BG_j = 2^{m-j}$ toe aan elke kolom j in de PPI matrix
  2. Bepaal **decimaal equivalent** $DE_i = \Sigma_j BG_j a_{ij}$ voor elke rij i
  3. Herschik rijen zodat DE_i's in dalende volgorde staan
     Indien reeds in dalende volgorde → stop; anders → stap 4
  4. Ken binair gewicht toe aan elke rij in de PPI matrix
  5. Bepaald decimaal equivalent voor elke kolom
  6. Herschik kolommen zodat DE_j's in dalende volgorde staan
     Indien reeds in dalende volgorde → stop; anders → stap 1

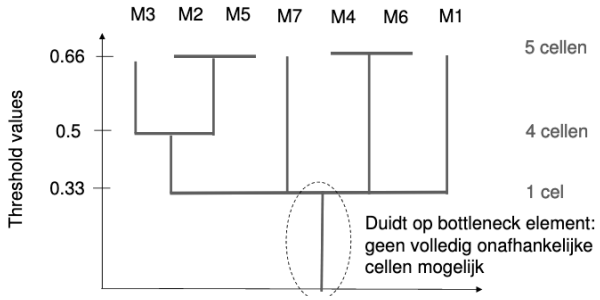
➔ Beide geven *géén oplossing* indien ideale blok-diagonale matrix onmogelijk is!

- Similariteitscoëfficiënten (SC) algoritme

$$s_{ij} = \frac{\#\text{ parts dat zowel machine i als j nodig heeft}}{\#\text{ parts dat ofwel i ofwel j ofwel allebei nodig heeft}}$$

1. Beschouw elke machine als aparte cel
   a. Bereken s_{ij} voor elk koppel machines
   b. Machines waarvoor $s_{ij} \geq$ drempelwaarde zijn een cel
2. Kies lagere drempelwaarde

Dendogram = visuele weergave van gevonden <u>machinecellen</u>:



- Mathematisch programmering (MP) of p-median algoritme

  <u>IP model</u>

  Min. $\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$ met d_{ij} = # NIET-gemeenschappelijke machines van parts i en j

  Odb $\sum_{i=1}^{n} x_{ij} = 1 \quad \forall i$

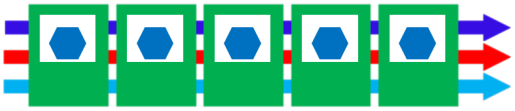
  $\sum_{j=1}^{n} x_{jj} = P$ = # gewenste families (op voorhand vastgelegd)

  $x_{ij} \leq x_{jj} \quad \forall i, j$

  $$x_{ij} = \begin{cases} 1 \text{ als part i wordt toegewezen aan familie j} \\ 0 \text{ anders} \end{cases}$$

Lijn/product lay-out/flow shop

- Gebonden vs ongebonden lijn
- Continu lopende band vs intermitterende lijn
- Strikte **takttijd** (veronderstel deterministisch)
- Single-model vs multi-model vs mixed-model
- Assembly line balancing vs line sequencing
  - Verdeel werk over stations zodat:
    - Volgordebeperkingen voldaan
    - Bewerkingstijd station $\leq$ takttijd
  - Optimaliseer vb. leegloop, # stations, …



<u>Single-model deterministic line balancing</u>

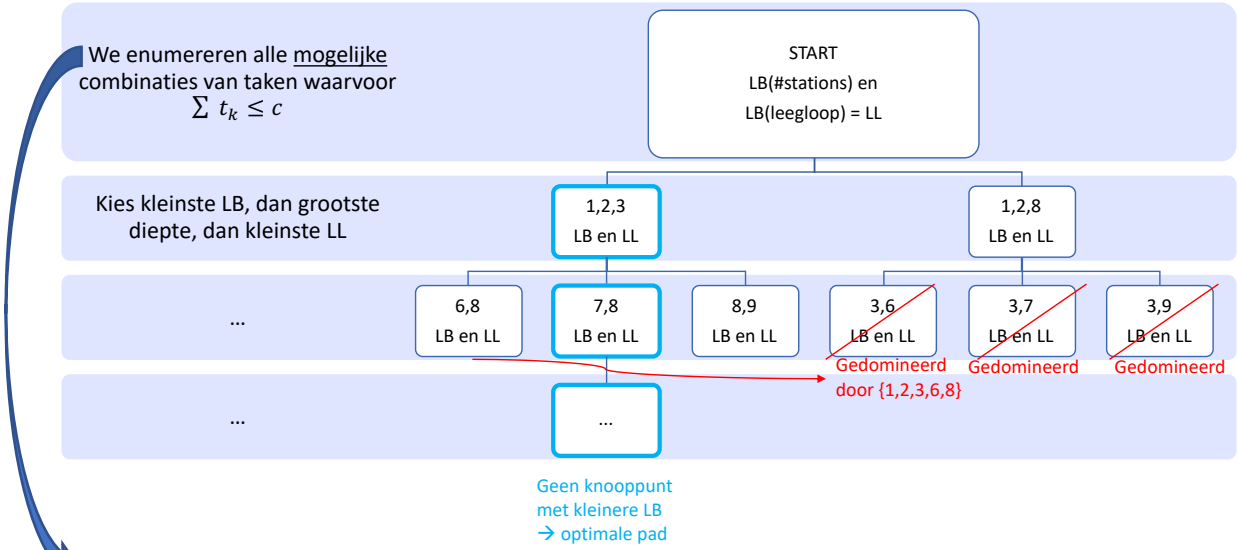
- Assemblage lijn voor **1 product**, waarvoor K taken uitgevoerd moeten worden met tijd per taak t_k deterministisch en gekend
- Stel: N eenheden maken in tijdsduur T
  - Takttijd/cyclustijd c = T/N
  - Productietempo = 1/c
  - Totale tijdsduur toegewezen taken aan station i = p_i
  - Leegloop station i = c – p_i
- **Gegeven c**, minimaliseer # stations n = minimaliseer totale leegloop:

  Min $\sum_{i=1}^{n}(c - p_i) = \min\,(nc - \sum_{i=1}^{n} p_i) = \min\,(nc - \sum_{k=1}^{K} t_k)$

  → **Singly Assembly Line Balancing type 1** (SALB-I)

Oplossingsmethoden SALB-I

Branch and bound (exact)



Voor lineaire lijnen: mogelijk = alle voorgangers zijn toegewezen
Voor **U-shaped lines**: mogelijk = alle voorgangers OF volgers zijn toegewezen

Ranked Positional Weight (heuristiek)

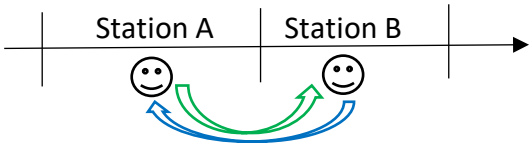
1. Elke taak x krijgt score $n(x) = t_x + t_{\text{alle volgers}}$
   Station nummer i = 1
2. Zolang nog taken kunnen toegevoegd worden aan station i:
   Wijs taak met hoogste n(x) toe aan i *indien* dit binnen de cyclustijd en volgordebeperkingen past
3. i=i+1

Hoffmann (heuristiek)

1. **Forward balance**: genereer voor elk station taaktoewijzing met kleinste leegloop, beginnend vooraan in het netwerk
2. **Reverse balance**: zelfde procedure, beginnend achteraan in het netwerk
3. Kies oplossing met kleinste # stations

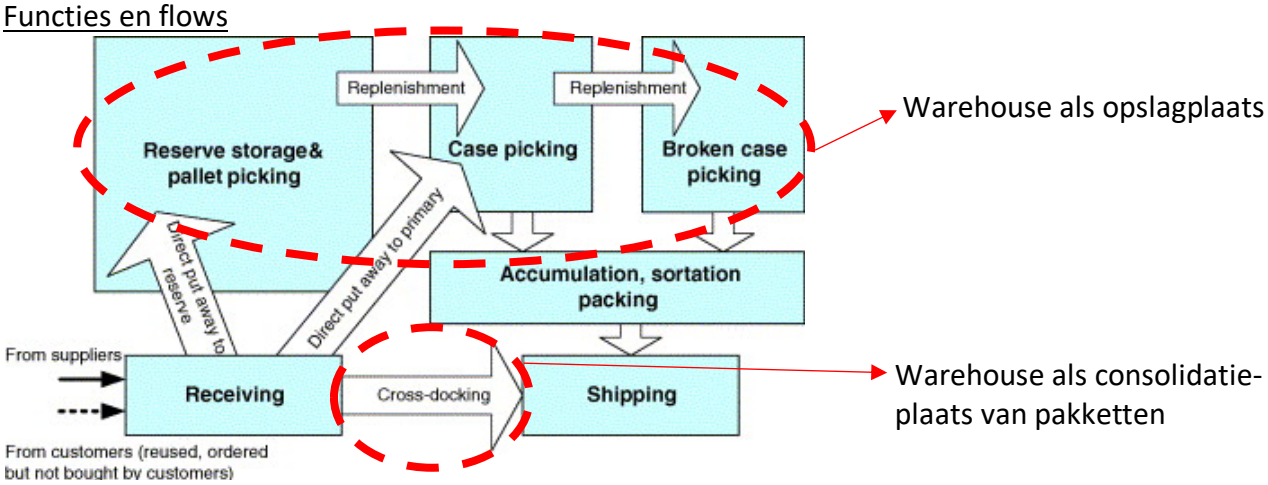
Mixed model line balancing

- Assemblage lijn voor **meerdere producten/types/modellen**, géén batches, met duurtaak taak k voor model j t_{kj}
- Stel: N_j eenheden maken van model j *per schift* met duurtijd T
  - Stel een gecombineerd volgordediagram op
  - **Taakgroep** $\widehat{t_k} = \sum_{j=1}^{J} N_j\, t_{jk}$
  - Totale work content toegewezen aan station i = T_i
- **Gegeven T**, minimaliseer # stations n = minimaliseer totale leegloop:
  Min $\sum_{i=1}^{n}(T - T_i) = \min\,(nT - \sum_{k=1}^{K} \widehat{t_k})$
  → Zelfde methoden als voor SALB-I toepassen!
- Nadeel: per model kan t_{kj} verschillen ➔ problemen takttijd!
  Oplossingen:
  - Taken in parallel uitvoeren: extra arbeiders nodig = **utility workers**
  - **Equipment-paced lines**: open to the left vs open to the right → sequencing!



Warehousing

Functies en flows



Daarnaast ook: after-sales, herstellen, testen, … én individuele orders (vooral e-commerce)!

Manuale order picking systemen

→ KPI: minimaliseer ophaaltijd van bestellingen!

- **Picker-to-parts**
  - Low level: manueel picken
  - High level: man-on-board Automated Storage and Retreival System (AS/RS)
- **Parts-to-picker**
  - Unit load AS/RS
  - Mini-load AS/RS
  - KIVA robot

Design van order picking systemen

→ Inspelen op setup/picking/zoek tijd

- Paper picklists
- Technologie: pick-by-voice, pick-by-light, pick-by-HMD, pick-by-vision

→ Inspelen op travel time van de picker ~ afgelegde afstand

- **Lay-out design**: # gangen, dwarsgangen, # blocks, plaats van depot, …
- **Opslagtoewijzing**
  - Random
  - Toegewezen
  - Full-turnover: laagste COI = m^3/# trips → dichtste bij depot
  - Class-based: random opslag binnen een klasse
- **Batching** = pick-by article i.p.v. pick-by-order
  - Sort-while-pick vs pick-and-sort
  - Proximity batching vs time-window batching
- **Zoning**
  - Progressive assembly/pick-and-pass
  - Parallel/synchronised picking: vb. wave picking
- **Routing**
  - S-shape, return, mid-point, largest gap, combined (heuristieken)
  - TSP (exact)

Deel II: Dynamisch gedrag van productiesystemen

= INZICHT IN OPERATIONELE PERFORMANTIE

Focus op lage **Work-In-Process** (WIP), hoge **Throughput** (TH) en lage **Cycle Time** (CT)
➔ Trade-off bij variabiliteit: hoge TH leidt tot WIP → ∞ en CT → ∞!

Eigenschappen van een systeem:

- Theoretische capaciteit vs effectieve capaciteit $r_e = \frac{m}{t_e}$ met m = # machines in parallel en t_e = gemiddelde *effectieve* procestijd → incl. storingen etc.
- Aankomstritme $r_a = \frac{1}{t_a}$
- Bezettingsgraad $u = \frac{r_a}{r_e}$ < 100% indien variabiliteit!!
- Bottleneck ritme r_b = praktische bovengrens op TH
- Raw process time $T_0 = \Sigma\, t_e$ = praktische ondergrens op gemiddelde CT
- Kritieke WIP $W_0 = r_b * T_0$ = ideale WIP indien geen variabiliteit

Variability basics

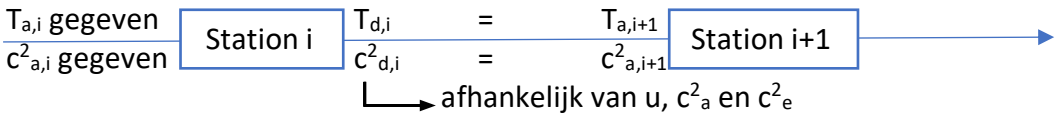
→ Wat zijn de oorzaken van variabiliteit? Hoe kunnen we dit kwantificeren?

Proces variability

- Effectieve procestijd T_e is *random* → gemiddeld t_e en standaarddeviatie σ_e
- Variatiecoëfficiënt $c_e = \frac{\sigma_e}{t_e}$ (dimensieloos!!)
- Preemptive outages
  - Kan op *eender welk moment* storen
  - Time to failure is random (m_f, σ_f, c_f) → machine beschikbaar
  - Time to repair is random (m_r, σ_r, c_r) → machine onbeschikbaar

  ➔ Availability factor A
- Nonpreemptive outages
  - Kan enkel *tussen jobs* in storen
  - Tijd dat een station buiten werking is T_S is random (t_S, σ_S, c_S)
- Quality problems, rework
  - Na productie op een station: part is defect met kans p
  - Ofwel 1 keer rework nodig → nonpreemptive outage met $N_S = 1/p$
  - Ofwel na rework terug kans op defect p

Flow variability



↳ afhankelijk van u, c^2_a en c^2_e

➔ Variabiliteit schuift mee door in de lijn!!

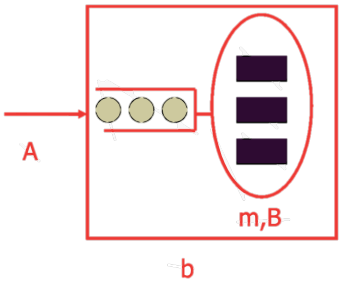
Variability interactions

→ Wat is de impact van variabiliteit op operationele performantie?

$CT_{q,i}$ = gemiddelde wachttijd aan station i *in steady state* (stabiel systeem)
TH = r_a
$CT_i = CT_{q,i} + t_e$ = gemiddelde flow time station i → voor een lijn: $CT = \sum_{i=1}^{N} (CT_{qi} + t_{e,i})$
$WIP_i = TH \times CT_i$ = gemiddelde WIP station i → voor een lijn: WIP = TH x CT (Wet van Little)

Queueing models

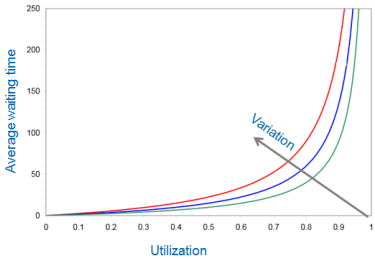
Kendall's notation: A/B/m(/b)



- M/M/1 en M/M/m
  - Zowel aankomst- als procestijd exponentieel verdeeld: $c^2_a = c^2_e = 1$
    - t_a ~ exponentieel(β) → r_a ~ Poisson(1/β)
    - Som van onafhankelijke t_a ~ Poisson(β) = t_a^{tot} ~ Poisson(β^{tot})
    - Som van *vele* onafhankelijk t_a (eender welke verdeling) = t_a^{tot} ~ Poisson
    - Splitsen van Poisson verdeeld proces volgens gelijke kansen geeft Poisson verdeelde deelprocessen
  - Max. aantal in systeem b = ∞
  - Bezettingsgraad u = $\frac{r_a}{r_e} = \frac{r_a t_e}{m}$!!!
- G/G/1 en G/G/m
  - Geen gegeven over verdeling aankomst- en procestijd

Algemene wetten

- Utilization law: bezettingsgraad u ↑ ⇒ WIP ↑↑ en CT ↑↑



  → **Trade-off** indien variabiliteit: hoge bezettingsgraad ↔ lage CT
- Capacity law: $r_a < r_{bottleneck}$ want anders WIP → ∞
  - Realiteit:
    - WIP beperkt door plaats → TH daalt
    - Reactie "fire-fighting" = reactief: herhaaldelijk **tijdelijke** ingrepen
- Lead time law: Lead time L neemt toe in gemiddelde **en standaarddeviatie** van CT!!
  - Service level = Pr(CT ≤ L)
- Variability law: Variabiliteit ↑ ⇒ WIP ↓ en CT ↓
  - Oplossingen:
    - Variabiliteit ↓ door investeren
    - Bezettingsgraad ↓ door extra capaciteit
    - Klanten afschermen van proces door investeren in voorraad
    - Anders: lange wachttijden en backlogs

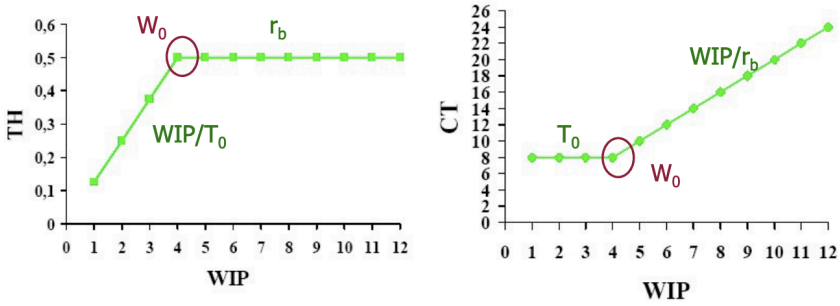
    } Buffering mechanismen
- Buffering law: variabiliteit impliceert buffering (**voorraad, capaciteit en/of tijd**)
  - "pay me now or pay me later"

Basic factory dynamics

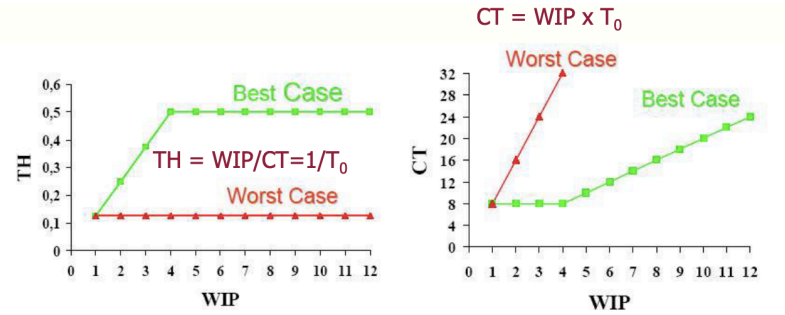
→ Wanneer is een proces "goed genoeg" en hoe kunnen we dit verbeteren?

<u>Gegeven r_b en T_0 (en $W_0 = r_b * T_0$) voor een systeem:</u>

- Best Case Law
  → Voor **deterministische** lijnen!



- Worst Case Law
  - Eerste job duurt 2*WIP h, andere jobs duren 0 h
    → **Deterministisch**, maar wel variabel!
  - Realistisch: process batching ("bad control")



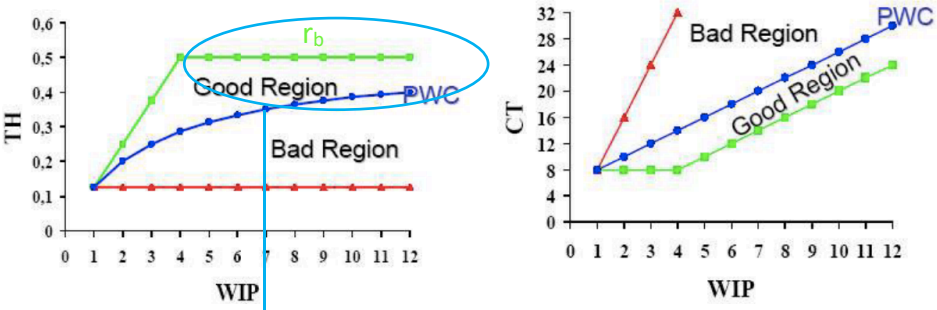
- Practical Worst Case
  Veronderstel:
  - Gebalanceerde lijn
  - 1 machine per station
  - Exponentieel verdeelde procestijden

  } Maximum randomness case (= alle *states* hebben gelijke kans)

  Dan $CT = N * \left[1 + \frac{1-w}{N}\right] * t_e = T_0 + \frac{w-1}{r_b}$

  en $TH = \frac{w}{CT}$

  → Grote WIP nodig om TH dichtbij r_b te krijgen, maar dan CT↑↑

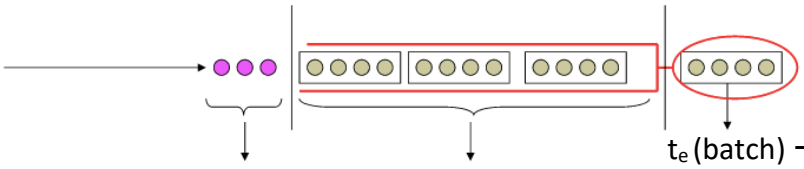


→ Als we TH willen verhogen, dan moeten we r_b verhogen!
Pas dan zal TH verhogen *voor elke WIP*

Batching and buffer sizing

Batching

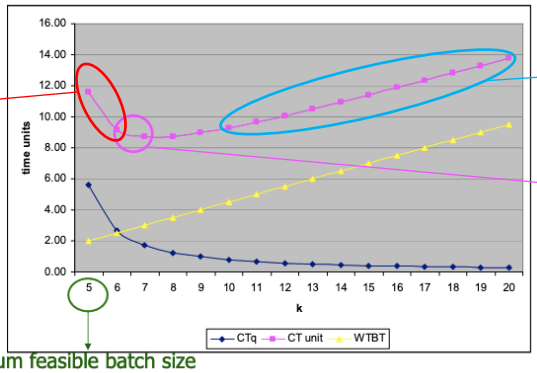
- **Parallel process batching**



WTBT $CT_q(unit) = CT_q(batch)$ t_e (batch) → onafhankelijk van k!!

→ CT(unit) = WTBT + CT_q(batch) + t_e(batch)

Saturatie-effect:
k↓ → u↑ → CT_q↑



Batching-effect:
k↑→ WTBT↑

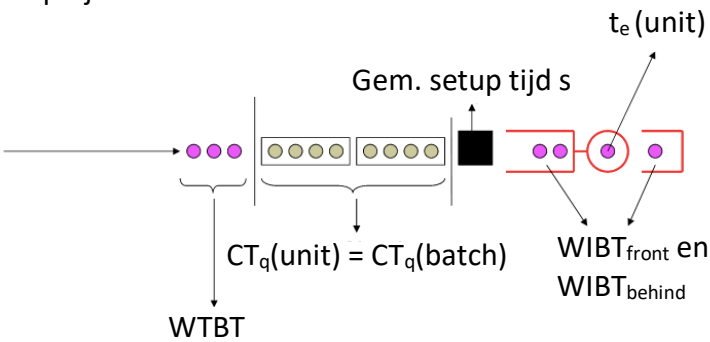
Optimale batchgrootte k minimaliseert CT!!

Minimum feasible batch size

Bezettingsgraad u(batch) = $\frac{r_a(batch)}{r_e(batch)} = \frac{r_a(unit)}{k * r_e(batch)}$ < 100% → k > …

- **Serial process batching**

  ! Setup tijd tussen elke batch !



t_e (unit)

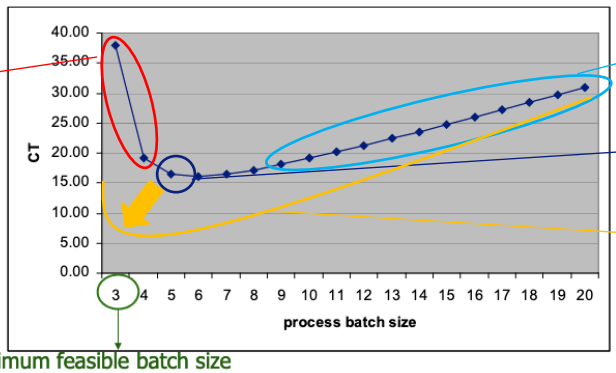
Gem. setup tijd s

$CT_q(unit) = CT_q(batch)$ $WIBT_{front}$ en $WIBT_{behind}$

WTBT

→ CT(unit) = WTBT + CT_q(batch) + $WIBT_{front}$ + t_e(batch) + $WIBT_{behind}$

Saturatie-effect:
k↓ → u↑ → CT_q↑



Batching-effect:
k↑→ WTBT↑ en WIBT↑

Optimale batchgrootte k minimaliseert CT!!

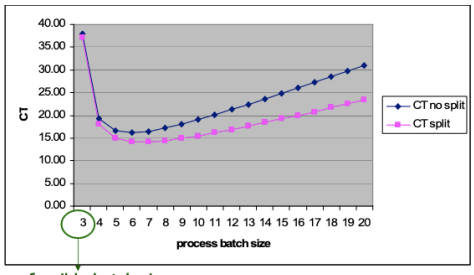
Setup tijd s↓
→ kleinere batchgroottes k mogelijk
+ optimale batchgrootte is kleiner

Minimum feasible batch size

Bezettingsgraad u(batch) = $\frac{r_a(batch)}{r_e(batch)} = r_a(unit)\left[\frac{s}{k} + t_e(unit)\right]$ < 100% → k > …

Merk op: als s → 0, dan k → 1: geen batching meer nodig!

- **Transfer batching**/move batching/lot splitting
  $WIBT_{behind}$ daalt:

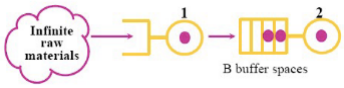


Minimum feasible batch size

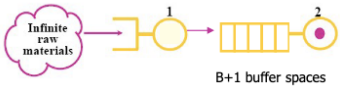
<u>Buffer sizing</u>

Buffers zijn **beperkt** wegens

- Beperkte ruimte



  → **Blocking**: buffer zit vol → station 1 valt stil ➔ verloren TH
  → **Starvation**: buffer is leeg → station 2 valt stil ➔ verloren TH
- Operationeel beleid (Kanban)



  Als alle kaarten in de buffers (aan WIP gehecht) zitten, dan is er geen trigger
  → Zelfde blocking/starvation effecten!!

Oplossing? Variabiliteit ↓↓↓

- Station 2: c_e^2 ↓ en/of c_a^2 ↓ → WIP_q ↓ → minder blocking
- Station 2: c_a^2 ↓ → stabielere inflow → minder starvation

Material flow control

Push/pull view – TACTISCH

PUSH

Planning (vooraf bepaald: exogeen)
o.b.v. due date

stuurt productie aan

Job

Productieproces

➔ WIP ongelimiteerd!
Vb. MRP

PULL

Unit uit WIP
(toestand van het systeem: endogeen)

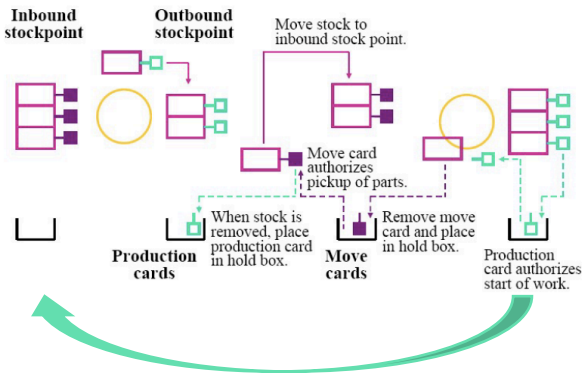
Productieproces

Job

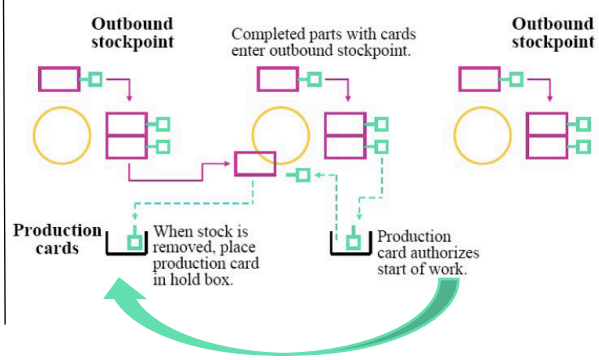
➔ Limiet op WIP!
Vb. Kanban, CONWIP

Kanban

O.b.v. productie- en movekaarten



of enkel productiekaarten



Signaal op het einde van de lijn triggert sequentieel productie vooraan in de lijn!
Dus ook: geen signaal → geen productie (≈ blocking)

Minder kaarten → minder WIP: WIP onder controle (gelimiteerd)

- CT ook onder controle
- TH daalt mee, tenzij variabiliteit ↓
  → JIT is **cruciaal** in combinatie met kanban!!

JIT

- Zero defects/zero quality control
- Zero breakdowns
- Zero setups

} Variabiliteit ↓

- Zero surging = **HEIJUNKA**
  → Stabiele eindassemblage planning
- Zero (excess) lot size/lot size of one

} Stabiele flow

- Zero handling
- Zero lead time

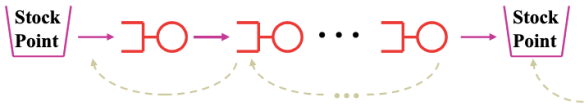
} CT↑ → WIP↑ willen we vermijden!

!! **NIET** WIP = 0 !! Buffers zijn noodzakelijk om TH te beschermen door storende factoren.

CONWIP

= "Constant WIP" o.b.v. kaarten **per lijn** (>< Kanban: per station)
→ Individueel station kan geen "blocking" ondervinden!

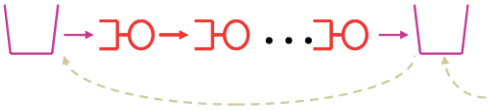
Kanban



- Beperking op WIP **per station**
- Kaarten zijn **product specifiek**
  → Wijziging in productmix/volumes moeilijk te accommoderen
- Werkelijke WIP < # kaarten door starvation
- Bottleneck houdt de hele lijn op

><

CONWIP



- Beperking op WIP **per lijn**
- Kaarten zijn **lijn specifiek**
  → Wijziging in productmix/volumes vereist enkel aanpassing <u>release list</u>
- Werkelijke WIP = # kaarten
  ➔ Hogere CT en hogere TH
- <u>Robuust</u>: WIP verplaatst vanzelf naar de bottleneck