



# Job shop scheduling by simulated annealing

Thomas Bamelis & Michiel Jonckheere

KU Leuven Kulak

Academiejaar 2017-2018

# Overzicht

Inleiding

De algemene vorm van simulated annealing

Waarom niet simpeler

SA toegepast op job shop scheduling

Besluit

# Overzicht

Inleiding

De algemene vorm van simulated annealing

Waarom niet simpeler

SA toegepast op job shop scheduling

Besluit

# Inleiding

## *Simulated annealing*

Een manier om dichtbij optimale oplossing te geven voor een combinatorisch optimalisatie probleem.

Gebaseerd op het afkoelen van metalen.

# Inleiding

## *Simulated annealing*

Een manier om dichtbij optimale oplossing te geven voor een combinatorisch optimalisatie probleem.

Gebaseerd op het afkoelen van metalen.

# Overzicht

Inleiding

De algemene vorm van simulated annealing

Waarom niet simpeler

SA toegepast op job shop scheduling

Besluit

# Gegevens

## 3 benodigdheden voor SA:

1.  $\mathcal{R}$  : Een verzameling configuraties / combinaties
2.  $C : \mathcal{R} \rightarrow \mathbb{R}$ : Een functie die de 'kost' van een configuratie weergeeft
3.  $\mathcal{N} : \mathcal{R} \rightarrow 2^{\mathcal{R}}$  : Een functie die de 'neighborhood' van een configuratie weergeeft, met  $\mathcal{N}(i) \subseteq \mathcal{R}$

Een simpele transitie is nodig waaruit indien toegepast op een configuratie  $i$ , alle  $\mathcal{N}(i)$  hieruit kunnen volgen

# Algoritme SA

▷ Iteratief algoritme waarin  $i \in \mathcal{R}$  gegeven is bij de initialisatie.

→ Er wordt een neighbor  $j \in \mathcal{N}(i)$  genomen

→ De kans dat we met  $j$  verdergaan is  $\min\{1, e^{\frac{-(C(j)-C(i))}{c}}\}$  met  $c$  een getal die daalt tijdens de uitvoering (cfr. *temperatuur*).

Kans daalt als:

- $C(j) - C(i)$  groot of dus  $j$  veel zwaarder
- $c$  kleiner wordt of dus tijd vordert

Kans 1 als  $C(j) \leq C(i)$



# Algoritme SA

▷ Iteratief algoritme waarin  $i \in \mathcal{R}$  gegeven is bij de initialisatie.

→ Er wordt een neighbor  $j \in \mathcal{N}(i)$  genomen

→ De kans dat we met  $j$  verdergaan is  $\min\{1, e^{\frac{-(C(j)-C(i))}{c}}\}$  met  $c$  een getal die daalt tijdens de uitvoering (cfr. *temperatuur*).

Kans daalt als:

- $C(j) - C(i)$  groot of dus  $j$  veel zwaarder
- $c$  kleiner wordt of dus tijd vordert

Kans 1 als  $C(j) \leq C(i)$

# Algoritme SA

▷ Iteratief algoritme waarin  $i \in \mathcal{R}$  gegeven is bij de initialisatie.

→ Er wordt een neighbor  $j \in \mathcal{N}(i)$  genomen

→ De kans dat we met  $j$  verdergaan is  $\min\{1, e^{\frac{-(C(j)-C(i))}{c}}\}$  met  $c$  een getal die daalt tijdens de uitvoering (*cfr. temperatuur*).

Kans daalt als:

- $C(j) - C(i)$  groot of dus  $j$  veel zwaarder
- $c$  kleiner wordt of dus tijd vordert

Kans 1 als  $C(j) \leq C(i)$

# Algoritme SA

▷ Iteratief algoritme waarin  $i \in \mathcal{R}$  gegeven is bij de initialisatie.

→ Er wordt een neighbor  $j \in \mathcal{N}(i)$  genomen

→ De kans dat we met  $j$  verdergaan is  $\min\{1, e^{\frac{-(C(j)-C(i))}{c}}\}$  met  $c$  een getal die daalt tijdens de uitvoering (cfr. *temperatuur*).

Kans daalt als:

- $C(j) - C(i)$  groot of dus  $j$  veel zwaarder
- $c$  kleiner wordt of dus tijd vordert

Kans 1 als  $C(j) \leq C(i)$

# Algoritme SA

▷ Iteratief algoritme waarin  $i \in \mathcal{R}$  gegeven is bij de initialisatie.

→ Er wordt een neighbor  $j \in \mathcal{N}(i)$  genomen

→ De kans dat we met  $j$  verdergaan is  $\min\{1, e^{\frac{-(C(j)-C(i))}{c}}\}$  met  $c$  een getal die daalt tijdens de uitvoering (*cfr. temperatuur*).

Kans daalt als:

- $C(j) - C(i)$  groot of dus  $j$  veel zwaarder
- $c$  kleiner wordt of dus tijd vordert

Kans 1 als  $C(j) \leq C(i)$

## c en aantal iteraties

Kies voor c begin waarde  $\chi_0$ , eindwaard  $\epsilon_s$  en  $\delta$

$$c_{k+1} = \frac{c_k}{1 + [c_k \cdot \ln(1 + \delta) / 3 \cdot \sigma_k]}$$

met  $\sigma_k$  de standaard afwijking van de kost van de configuraties van de k'de iteratie

→  $\delta$  klein gekozen  $\Rightarrow$  trage 'kwalitatieve' afname c

We kiezen Lengte L als bovengrens van het aantal mogelijke elementen om te bereiken uit een transitie vanuit i:

$$L_k = \max_{i \in \mathcal{R}} \{|\mathcal{N}(i)|\}$$

## c en aantal iteraties

Kies voor c begin waarde  $\chi_0$ , eindwaarde  $\epsilon_s$  en  $\delta$

$$c_{k+1} = \frac{c_k}{1 + [c_k \cdot \ln(1 + \delta) / 3 \cdot \sigma_k]}$$

met  $\sigma_k$  de standaard afwijking van de kost van de configuraties van de k'de iteratie

→  $\delta$  klein gekozen  $\Rightarrow$  trage 'kwalitatieve' afname c

We kiezen Lengte L als bovengrens van het aantal mogelijke elementen om te bereiken uit een transitie vanuit i:

$$L_k = \max_{i \in \mathcal{R}} \{|\mathcal{N}(i)|\}$$

## c en aantal iteraties

Kies voor c begin waarde  $\chi_0$ , eindwaarde  $\epsilon_s$  en  $\delta$

$$c_{k+1} = \frac{c_k}{1 + [c_k \cdot \ln(1 + \delta) / 3 \cdot \sigma_k]}$$

met  $\sigma_k$  de standaard afwijking van de kost van de configuraties van de k'de iteratie

→  $\delta$  klein gekozen  $\Rightarrow$  trage 'kwalitatieve' afname c

We kiezen Lengte L als bovengrens van het aantal mogelijke elementen om te bereiken uit een transitie vanuit i:

$$L_k = \max_{i \in \mathcal{R}} \{|\mathcal{N}(i)|\}$$

# Kwaliteit algoritme

- ▷ Indien  $\delta$  kleiner dan 0,1 gekozen wordt is de eindconfiguratie binnen 2% van het globaal minimum.

Tijdscomplexiteit =  $\theta(\tau L \ln(|\mathcal{R}|))$  met  $\tau$  de tijd om een nieuwe transitie door te voeren.



# Overzicht

Inleiding

De algemene vorm van simulated annealing

Waarom niet simpeler

SA toegepast op job shop scheduling

Besluit

# Waarom niet simpeler?

Wat indien men met  $j$  verdergaat als  $C(j) \geq C(i)$ ?

Stel  $i \in \mathcal{R}$  en  $\forall j \in \mathcal{N}(i) : C(j) \geq C(i)$   
( $i$  heeft kleinste kost van al zijn neighbors)  
 $\Rightarrow$  je zit vast in  $\mathcal{N}(i)$

SA 'verplicht' om te veranderen in begin en zo niet vast te zitten

# Waarom niet simpeler?

Wat indien men met  $j$  verdergaat als  $C(j) \geq C(i)$ ?

Stel  $i \in \mathcal{R}$  en  $\forall j \in \mathcal{N}(i) : C(j) \geq C(i)$   
( $i$  heeft kleinste kost van al zijn neighbors)  
 $\Rightarrow$  je zit vast in  $\mathcal{N}(i)$

SA 'verplicht' om te veranderen in begin en zo niet vast te zitten

# Waarom niet simpeler?

Wat indien men met  $j$  verdergaat als  $C(j) \geq C(i)$ ?

Stel  $i \in \mathcal{R}$  en  $\forall j \in \mathcal{N}(i) : C(j) \geq C(i)$   
( $i$  heeft kleinste kost van al zijn neighbors)  
 $\Rightarrow$  je zit vast in  $\mathcal{N}(i)$

SA 'verplicht' om te veranderen in begin en zo niet vast te zitten

# Overzicht

Inleiding

De algemene vorm van simulated annealing

Waarom niet simpeler

SA toegepast op job shop scheduling

Besluit

# Configuraties

We geven hier de gegevens die SA nodig heeft door voor ons probleem.

## *Configuraties:*

De configuratie  $i$  wordt weergegeven door  $\Pi_i$  met

$\Pi_i = \{\pi_{i1}, \dots, \pi_{im}\}$  met  $\pi_{ik}$  de volgorde waarin de operaties op machine  $k$  worden uitgevoerd.

$\pi_{ik}$  moet  $m_k$  operaties bevatten  $\Rightarrow \#\mathcal{R} = \prod_{k=1}^m m_k!$  met  $m$  aantal machines

Als operatie  $v$  op machine  $k$  uitvoert, dan is  $\pi_{ik}(v)$  de operatie die na  $v$  op machine  $k$  uitvoert.

$\pi_{ik}^l(v)$  is de  $l$ 'de operatie na  $v$ .

# Configuraties

We geven hier de gegevens die SA nodig heeft door voor ons probleem.

## **Configuraties:**

De configuratie  $i$  wordt weergegeven door  $\Pi_i$  met

$\Pi_i = \{\pi_{i1}, \dots, \pi_{im}\}$  met  $\pi_{ik}$  de volgorde waarin de operaties op machine  $k$  worden uitgevoerd.

$\pi_{ik}$  moet  $m_k$  operaties bevatten  $\Rightarrow \#\mathcal{R} = \prod_{k=1}^m m_k!$  met  $m$  aantal machines

Als operatie  $v$  op machine  $k$  uitvoert, dan is  $\pi_{ik}(v)$  de operatie die na  $v$  op machine  $k$  uitvoert.

$\pi_{ik}^l(v)$  is de  $l$ 'de operatie na  $v$ .

# Configuraties

We geven hier de gegevens die SA nodig heeft door voor ons probleem.

## ***Configuraties:***

De configuratie  $i$  wordt weergegeven door  $\Pi_i$  met

$\Pi_i = \{\pi_{i1}, \dots, \pi_{im}\}$  met  $\pi_{ik}$  de volgorde waarin de operaties op machine  $k$  worden uitgevoerd.

$\pi_{ik}$  moet  $m_k$  operaties bevatten  $\Rightarrow \#\mathcal{R} = \prod_{k=1}^m m_k!$  met  $m$  aantal machines

Als operatie  $v$  op machine  $k$  uitvoert, dan is  $\pi_{ik}(v)$  de operatie die na  $v$  op machine  $k$  uitvoert.

$\pi_{ik}^l(v)$  is de  $l$ 'de operatie na  $v$ .



# Kost functie

Stel  $V$  de operaties en  $A$  de gerichte bogen met  $(v, w) \in A$ ,  $v$  en  $w$  in zelfde job en  $w$  voert direct na  $v$  uit.  
( $n$  jobs,  $m$  machines  $\geq 2$  en  $N$  operaties)

Definiëren twee grafen:

1.  $D_i = (V, A \cup E_i)$ , met  $E_i = \{(v, w) | \{v, w\} \in E \text{ en } \pi_{ik}(v) = w, \text{ voor een } k \in (M)\}$   
 $\rightarrow (v, w)$  in  $E_i$  als  $w$  *direct* na  $v$  uitvoert
2.  $\bar{D}_i = (V, A \cup \bar{E}_i)$ , met  $\bar{E}_i = \{(v, w) | \{v, w\} \in E \text{ en } \pi_{ik}^l(v) = w, \text{ voor een } k \in (M) \text{ en een } 1 \leq l \leq m_k - 1\}$   
 $\rightarrow (v, w)$  in  $\bar{E}_i$  als  $w$  na  $v$  wordt uitgevoerd

# Kost functie

Stel  $V$  de operaties en  $A$  de gerichte bogen met  $(v, w) \in A$ ,  $v$  en  $w$  in zelfde job en  $w$  voert direct na  $v$  uit.  
( $n$  jobs,  $m$  machines  $\geq 2$  en  $N$  operaties)

Definiëren twee grafen:

1.  $D_i = (V, A \cup E_i)$ , met  $E_i = \{(v, w) | \{v, w\} \in E \text{ en } \pi_{ik}(v) = w, \text{ voor een } k \in (M)\}$   
 $\rightarrow (v, w)$  in  $E_i$  als  $w$  *direct* na  $v$  uitvoert
2.  $\bar{D}_i = (V, A \cup \bar{E}_i)$ , met  $\bar{E}_i = \{(v, w) | \{v, w\} \in E \text{ en } \pi_{ik}^l(v) = w, \text{ voor een } k \in (M) \text{ en een } 1 \leq l \leq m_k - 1\}$   
 $\rightarrow (v, w)$  in  $\bar{E}_i$  als  $w$  na  $v$  wordt uitgevoerd

# Kost functie

Het langste pad van beide grafen zijn even lang. (Overige paden zijn korter of afkorting van een pad)

⇒ Kost van configuratie  $i$  vinden we door het langste pad van beginknoop/bron 0 naar eindknoop/put  $N + 1$  te zoeken in  $D_i$

→ *labeling algoritme*

→ kortste pad algoritme met gewicht boog = -1

Dit heeft tijdscomplexiteit  $\Theta(\#bogen)$

$|E_i|$  = operaties - # beginknopen machines =  $N - m$

$|A|$  = operaties - # beginknopen jobs =  $N + n$

Want iedere knoop heeft precies 1 toekomstige behalve 1e.

We vergeten uitknopen 0 en inknopen  $N + 1$

Aantal bogen:  $|A| + |E_i| = (N + n) + (N - m)$

⇒  $O(N)$

# Kost functie

Het langste pad van beide grafen zijn even lang. (Overige paden zijn korter of afkorting van een pad)

⇒ Kost van configuratie  $i$  vinden we door het langste pad van beginknoop/bron 0 naar eindknoop/put  $N + 1$  te zoeken in  $D_i$

→ *labeling algoritme*

→ kortste pad algoritme met gewicht boog = -1

Dit heeft tijdscomplexiteit  $\Theta(\#bogen)$

$|E_i|$  = operaties - # beginknopen machines =  $N - m$

$|A|$  = operaties - # beginknopen jobs =  $N + n$

Want iedere knoop heeft precies 1 toekomstige behalve 1e.

We vergeten uitknopen 0 en inknopen  $N + 1$

Aantal bogen:  $|A| + |E_i| = (N + n) + (N - m)$

⇒  $O(N)$

# Kost functie

Het langste pad van beide grafen zijn even lang. (Overige paden zijn korter of afkorting van een pad)

⇒ Kost van configuratie  $i$  vinden we door het langste pad van beginknoop/bron 0 naar eindknoop/put  $N + 1$  te zoeken in  $D_i$

→ *labeling algoritme*

→ kortste pad algoritme met gewicht boog = -1

Dit heeft tijdscomplexiteit  $\Theta(\#bogen)$

$|E_i|$  = operaties - # beginknopen machines =  $N - m$

$|A|$  = operaties - # beginknopen jobs =  $N + n$

Want iedere knoop heeft precies 1 toekomstige behalve 1e.

We vergeten uitknopen 0 en inknopen  $N + 1$

Aantal bogen:  $|A| + |E_i| = (N + n) + (N - m)$

⇒  $O(N)$

# Neighborhood Structuur

Transitie:

Neem bogen  $v$  en  $w$  zodat

1.  $v$  en  $w$  zijn opeenvolgende operaties op een machine  $k$
2.  $(v, w) \in E_i$  is een kritieke boog

Daarna  $v$  en  $w$  wisselen in uitvoering.

$\Rightarrow (u, v), (v, w), (w, x) \in E_i$  worden  
 $(u, w), (w, v), (v, x) \in E_i$

Geen kringen worden gecreëerd (kan bewezen worden) en omdat  $(v, w) \in$  cruciaal pad zal de transitie effect hebben op de kost.

$\mathcal{N}(i) =$  verzameling grafen bereikbaar uit  $i$  door omwisselen cruciale boog. Cruciaal pad  $\subset E_i$  en  $m_k - 1$  bogen per machine  
 $\Rightarrow |\mathcal{N}(i)| < |E_i| = \sum_{k=1}^m (m_k - 1) = N - m$

# Neighborhood Structuur

Transitie:

Neem bogen  $v$  en  $w$  zodat

1.  $v$  en  $w$  zijn opeenvolgende operaties op een machine  $k$
2.  $(v, w) \in E_i$  is een kritieke boog

Daarna  $v$  en  $w$  wisselen in uitvoering.

$\Rightarrow (u, v), (v, w), (w, x) \in E_i$  worden  
 $(u, w), (w, v), (v, x) \in E_i$

Geen kringen worden gecreëerd (kan bewezen worden) en omdat  $(v, w) \in$  cruciaal pad zal de transitie effect hebben op de kost.

$\mathcal{N}(i)$  = verzameling grafen bereikbaar uit  $i$  door omwisselen cruciale boog. Cruciaal pad  $\subset E_i$  en  $m_k - 1$  bogen per machine  
 $\Rightarrow |\mathcal{N}(i)| < |E_i| = \sum_{k=1}^m (m_k - 1) = N - m$

# Convergentie

Via een bewijs kan aangetoond worden dat voor iedere begin configuratie  $i$  een eindig aantal transities bestaat om een globaal minimale configuratie te bereiken.



# Overzicht

Inleiding

De algemene vorm van simulated annealing

Waarom niet simpeler

SA toegepast op job shop scheduling

Besluit

# Besluit

SA blijft relatief simpel, snel en goed en vermijdt de valkuil van de simpelere methode.