Job shop scheduling by simulated annealing

Thomas Bamelis & Michiel Jonckheere

KU Leuven Kulak

Academiejaar 2017-2018

Overzicht

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

KU LEUVEN kulak

Gegevens

3 benodigdheden voor SA:

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

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

Algoritme SA

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

→ Er wordt een neighbor $j \in \mathscr{N}(i)$ 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 $C(j) \leqslant C(i)$

Algoritme SA

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

$\rightarrow$ Er wordt een neighbor $j \in \mathscr{N}(i)$ genomen

$\rightarrow$ 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 $C(j) \leqslant C(i)$

Algoritme SA

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

$\rightarrow$ Er wordt een neighbor $j \in \mathscr{N}(i)$ genomen

$\rightarrow$ 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 $C(j) \leqslant C(i)$

Algoritme SA

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

$\rightarrow$ Er wordt een neighbor $j \in \mathscr{N}(i)$ genomen

$\rightarrow$ 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 $C(j) \leqslant C(i)$

Algoritme SA

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

$\rightarrow$ Er wordt een neighbor $j \in \mathscr{N}(i)$ genomen

$\rightarrow$ 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 $C(j) \leqslant C(i)$

c en aantal iteraties

Kies voor c begin waarde χ_0, eindwaard ϵ_s en δ

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

met σ_k de standaard afwijking van de kost van de configuraties van de k'de iteratie

$\rightarrow \delta$ 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:

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

c en aantal iteraties

Kies voor c begin waarde χ_0, eindwaard ϵ_s en δ

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

met σ_k de standaard afwijking van de kost van de configuraties van de k'de iteratie

$\rightarrow \delta$ 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:

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

c en aantal iteraties

Kies voor c begin waarde χ_0, eindwaard ϵ_s en δ

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

met σ_k de standaard afwijking van de kost van de configuraties van de k'de iteratie

$\rightarrow \delta$ 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:

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

Kwaliteit algoritme

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

Tijdscomplexiteit $= \theta(\tau L \ln(|\mathscr{R}|))$ met τ 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 $C(j) \leq C(i)$?

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

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

Waarom niet simpeler?

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

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

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

Waarom niet simpeler?

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

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

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

Overzicht

Configuraties

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

Configuraties:

De configuratie i wordt weergegeven door Π_{i} met $\Pi_{\text{i}} = \{\pi_{\text{i}1}, ..., \pi_{\text{im}}\}$ met π_{ik} de volgorde waarin de operaties op machine k worden uitgevoerd.

π_{ik} moet m_{k} operaties bevatten $\Rightarrow \#\mathscr{R} = \prod_{k=1}^{m} m_{\text{k}}!$ met m aantal machines

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

Configuraties

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

Configuraties:

De configuratie *i* wordt weergegeven door Π_i met $\Pi_i = \{\pi_{i1}, ..., \pi_{im}\}$ met π_{ik} de volgorde waarin de operaties op machine k worden uitgevoerd.

π_{ik} moet m_k operaties bevatten $\Rightarrow \#\mathscr{R} = \prod_{k=1}^{m} m_k!$ 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.

Configuraties

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

Configuraties:

De configuratie *i* wordt weergegeven door Π_i met $\Pi_i = \{\pi_{i1}, ..., \pi_{im}\}$ met π_{ik} de volgorde waarin de operaties op machine k worden uitgevoerd.

π_{ik} moet m_k operaties bevatten $\Rightarrow \#\mathscr{R} = \prod_{k=1}^{m} m_k!$ 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.

Kost functie

We definiëren volgende twee grafen:

1. $D_i = (V, A \cup E_i)$, met $E_i = \{(v,w) | \{v,w\} \in E$ en $\pi_{ik}(v) = w$, voor enkele $k \in (M)\}$
2. $\bar{D}_i = (V, A \cup \bar{E}_i)$, met $\bar{E}_i = \{(v,w) | \{v,w\} \in E$ en $\pi_{ik}^{l}(v) = w$, voor enkele $k \in (M), 1 \leq l \leq m_k - 1\}$

$\bar{D}_i$ is een disjuncte graaf met bogen uit E, waarvan de oriëntatie bepaald wordt door Π_i.

D_i is $\bar{D}_i$ waarvan de bogen uit $\bar{E}_i$ die opeenvolgende operaties zijn van dezelfde machine.

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 0 naar $N + 1$ te zoeken in D_i $\rightarrow$ m.b.v. *labeling algoritme*
$\rightarrow$ kortste pad algoritme met gewicht boog = -1

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

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 0 naar $N+1$ te zoeken in D_i $\rightarrow$ m.b.v. *labeling algoritme*

$\rightarrow$ kortste pad algoritme met gewicht boog = -1

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

$\Rightarrow O(N)$

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 0 naar $N + 1$ te zoeken in D_i $\rightarrow$ m.b.v. *labeling algoritme*
$\rightarrow$ kortste pad algoritme met gewicht boog = -1

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

Neighborhood Structuur

Keuze transitie:

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.

$\Rightarrow (u, v)$ en (w, x) wordt (u, w) en (v, x)

Enkele transities dus zeker al uitsluiten:

1. De kost verlaagt niet na de transitie
2. De transitie resulteert in een cyclic graph

We besluiten dus: $\mathcal{N}(i) < \Sigma_{k=1}^{m}(m_k - 1) = N - m$

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 en vermijd de valkuil van de simpelere methode.