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

Gegevens

3 benodigdheden voor SA:

1. $\mathscr{R}$: Een verzameling configuraties / combinaties
2. $C : \mathscr{R} \to \mathbb{R}$: Een functie die de 'kost' van een configuratie weergeeft
3. $\mathscr{N} : \mathscr{R} \to 2^{\mathscr{R}}$: 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.

→ 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 $C(j) \leqslant C(i)$

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

Lengte ?van de k'de markovketen?:

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

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

Lengte ?van de k'de markovketen?:

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

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

Lengte ?van de k'de markovketen?:

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

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

KU LEUVEN kulak

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

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 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.

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.

Kost functie

Hier op dezel

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.