Modelli di scheduling su macchine parallele

1. Considerazioni generali

- Il processamento di job su macchine parallele è importante sia dal punto di vista *teorico* che *pratico*.
  Dal punto di vista *teorico* questo caso è una *generalizzazione dello scheduling su macchina singola*, mentre dal punto di vista *pratico* è rilevante in quanto l'esistenza di risorse parallele in comune si verifica nella *realtà*.

- I principali *obiettivi* sono: *makespan*, *tempo di completamento totale*, *massima lateness*.
  Il *makespan* nel caso di macchina singola e' rilevante solo se i tempi di set-up dipendono dalla sequenza mentre nel caso di macchine parallele la *minimizzazione* del *makespan* in ogni caso assicura un bilanciamento del carico tra le macchine.

- Inoltre la *preempzione* che nel caso di maccchina singola è significativa solo in presenza di "*ready time*", acquista *importanza* in questo caso *anche con job simultanei*.

- Infine, anche se nella maggioranza dei casi questi modelli danno luogo a schedule ottime che sono non-delay, nel caso di minimizzazione del "total completion time" senza preempzione e con macchine parallele non-correlate la schedula ottima non è necessariamente non-delay

Modelli di scheduling su macchine parallele

1. Considerazioni generali *(continua)*

- Lo scheduling su macchine parallele è un processo in due passi:
  a) *allocazione dei job alle macchine*;
  b) *sequenziamento dei job su ciascuna macchina*.
- Se l'obiettivo è il *makespan* *solo* il passo *a)* è *significativo*.
- Questo modello ha un notevole interesse pratico perché la *minimizzazione* di C_{max} ha un effetto di *bilanciamento* del carico tra le varie *macchine*.
- Questo problema è *NP-hard* come si può facilmente osservare nel caso più semplice $P2||C_{max}$ che è riconducibile al noto problema del *PARTITION* che è un classico problema *NP-completo* (in senso ordinario).

  Dimostrazione.

  Per dimostrare la complessità di $P2||C_{max}$ occorre richiamare la *definizione* del *PARTITION* problem:

  Dato un insieme di interi positivi $a_1, a_2, \ldots, a_t$ e $b = ½ \sum_{i=1}^{t} a_i$, esistono due sottoinsiemi disgiunti S_1, S_2 tali che $\sum_{i \in S_j} a_i = b,\ j = 1, 2$?

Modelli di scheduling su macchine parallele

1. Considerazioni generali (continua)

E' evidente che il *PARTITION* si riduce al $P2||C_{max}$ ponendo:

$$n = t; \qquad p_j = a_j; \quad z = ½ \sum_{i=1}^{t} a_i = b$$

E' immediato verificare che esiste una schedula con un valore di $C_{max} \leq ½ \sum_{i=1}^{t} a_i$ se e solo se esiste una soluzione per il *PARTITION*.
Quindi il problema $P2||C_{max}$ è almeno tanto complesso quanto il *PARTITION*.

- Di conseguenza molti algoritmi euristici sono stati sviluppati per il modello generale.

Modelli di scheduling su macchine parallele

2. Il modello $Pm||C_{max}$

- La regola *Longest Processing Time* first (*LPT*)

  Assegna al tempo $t = 0$ gli m job più lunghi alle m macchine. Successivamente, quando una macchina si libera assegna ad essa il job non schedulato con il più grande tempo di processamento.

  Osservazione

  Questa euristica cerca di assegnare i job più corti alla fine della schedula dove essi possono essere usati per bilanciare i carichi tra le macchine.

- Per questa euristica esiste un'approssimazione garantita data dal seguente

  Teorema

  In un problema $Pm||C_{max}$ se indichiamo con $C_{max}(\text{LPT})$ il *makespan* della *schedula LPT* e con $C_{max}(\text{OPT})$ il *makespan* della *schedula ottima* vale la seguente relazione

$$\frac{C_{max}(\text{LPT})}{C_{max}(\text{OPT})} \leq \frac{4}{3} - \frac{1}{3m}$$

Modelli di scheduling su macchine parallele

2. Il modello $Pm||C_{\max}$ (continua)

Dimostrazione (per contraddizione).

Supponiamo che esistano uno o più contro-esempi per i quali la precedente relazione vale col segno $>$ e consideriamo quello con il più piccolo numero di job. Siano essi n. Questo contro-esempio ha la seguente proprietà: in una *schedula LPT* il *job più corto è l'ultimo ad essere processato e l'ultimo ad essere completato*.

Senz'altro la regola *LPT* assicura che il job più corto e' l'ultimo ad essere processato.

Se non fosse anche l'ultimo ad essere completato, allora l'eliminazione di questo job darebbe luogo ad un contro-esempio con un numero più piccolo di job.
Infatti $C_{\max}(\text{LPT})$ resterebbe lo stesso mentre $C_{\max}(\text{OPT})$ potrebbe restare lo stesso o decrescere.

Pertanto nel contro-esempio di cardinalità minima lo "*starting time*" del *job più corto* in una *schedula LPT* è $C_{\max}(\text{LPT}) - p_n$.

Poiché almeno fino a questo istante tutte le macchine sono occupate ne segue che

$$C_{\max}(\text{LPT}) - p_n \leq \frac{1}{m}\sum_{j=1}^{n-1} p_j$$

Modelli di scheduling su macchine parallele

2. Il modello $Pm||C_{\max}$ *(continua)*

Ove il segno $=$ si ha quando schedulando i primi $n - 1$ job secondo la regola *LPT* si verifica che il carico (tempo) di lavoro sulla generica macchina i (somma dei tempi di processa mento dei job allocati ad i) è esattamente lo stesso per tutte le macchine.

Ora si ricava facilmente che:

$$C_{\max}(\mathrm{LPT}) \le p_n + \frac{1}{m}\sum_{j=1}^{n-1} p_j = p_n\left(1-\frac{1}{m}\right) + \frac{1}{m}\sum_{j=1}^{n} p_j$$

Poiché per definizione

$$C_{\max}(\mathrm{OPT}) \ge \frac{1}{m}\sum_{j=1}^{n} p_j$$

per il contro-esempio devono valere le seguenti relazioni:

$$\frac{4}{3}-\frac{1}{3m} < \frac{C_{\max}(\mathrm{LPT})}{C_{\max}(\mathrm{OPT})} \le \frac{p_n\left(1-\frac{1}{m}\right) + \frac{1}{m}\sum_{j=1}^{n} p_j}{C_{\max}(\mathrm{OPT})} =$$

$$= \frac{p_n\left(1-\frac{1}{m}\right)}{C_{\max}(\mathrm{OPT})} + \frac{\frac{1}{m}\sum_{j=1}^{n} p_j}{C_{\max}(\mathrm{OPT})} \le \frac{p_n\left(1-\frac{1}{m}\right)}{C_{\max}(\mathrm{OPT})} + 1$$

Modelli di scheduling su macchine parallele

2. Il modello $Pm||C_{\max}$ (continua)

Pertanto

$$\frac{4}{3}-\frac{1}{3m}<\frac{p_n(1-1/m)}{C_{\max}(\mathrm{OPT})}+1 \Rightarrow C_{\max}(\mathrm{OPT})<3p_n$$

Poiché l'ultima relazione è una disuguaglianza stretta, questo implica che per il contro-esempio in esame la schedula ottima può avere al più due job su ogni macchina.

Poiché è facile dimostrare che se una schedula ottima presenta al più due job su ciascuna macchina, allora la schedula LPT è ottima ed il rapporto dei due makespan è pari ad 1, ne risulta una contraddizione con l'assunto di partenza che per $m \geq 2$ impone che sia

$$\frac{C_{\max}(\mathrm{LPT})}{C_{\max}(\mathrm{OPT})}>1$$

Pertanto deve valere la relazione riportata nell'enunciato del teorema. c.v.d

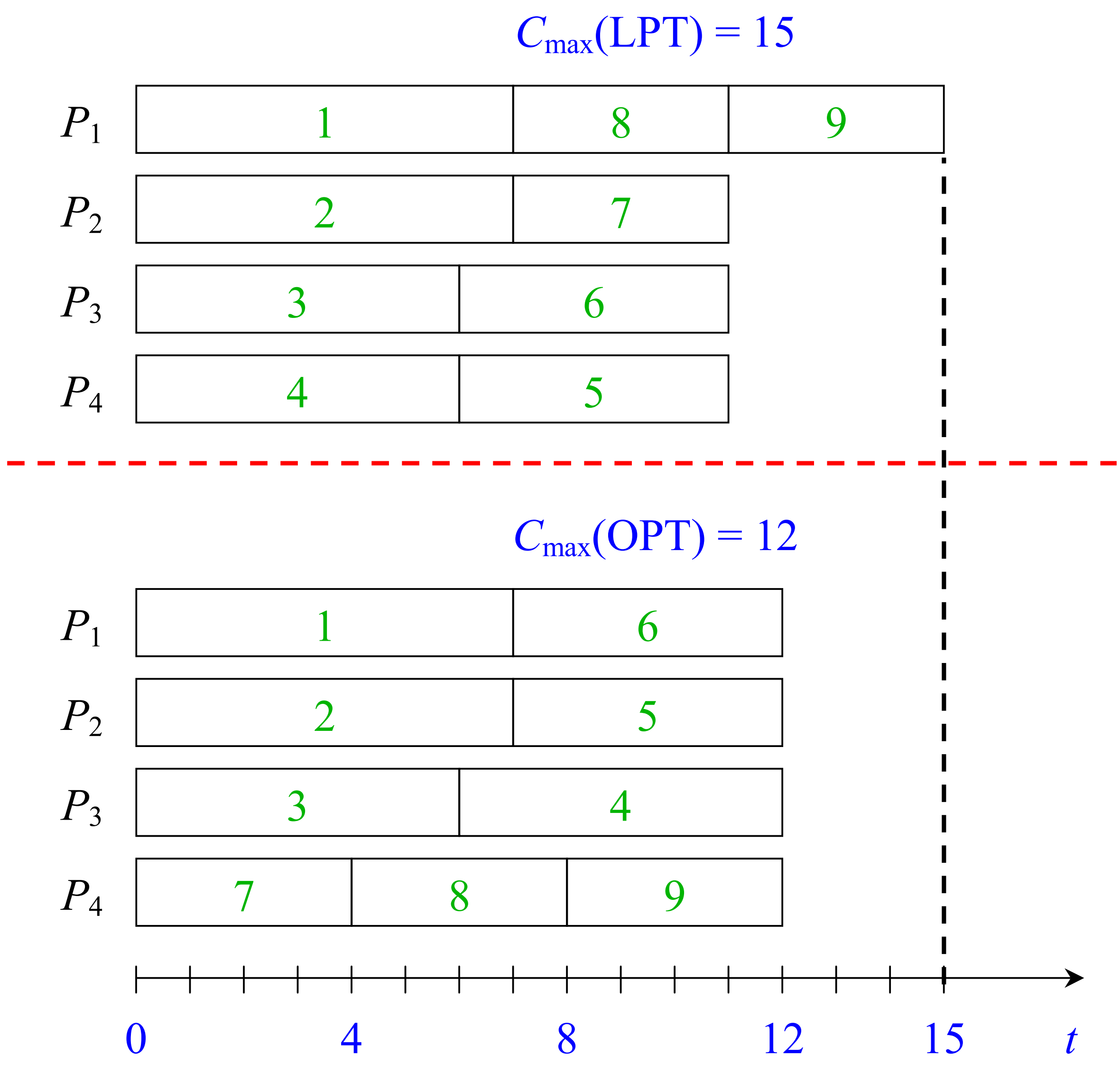
Modelli di scheduling su macchine parallele

2. Il modello $Pm||C_{\max}$ (continua)

- Consideriamo un'istanza con $m = 4$ macchine e la seguente configurazione dei job

job	1	2	3	4	5	6	7	8	9
p_j	7	7	6	6	5	5	4	4	4

- Le schedule LPT e OPT sono le seguenti:



Modelli di scheduling su macchine parallele

2. Il modello $Pm||C_{max}$ (continua)

- Dalle schedule si ricava che

$$\frac{C_{max}(\text{LPT})}{C_{max}(\text{OPT})} = \frac{15}{12} = \frac{4}{3} - \frac{1}{3m}; \qquad m = 4$$

  Ossia la particolare istanza esaminata rappresenta il caso peggiore (worst-case) quando si hanno 4 macchine parallele (l'approsimazione è del 25%).

- Esistono altre euristiche per il problema $Pm||C_{max}$ che sono più sofisticate della regola LPT e che conseguentemente danno luogo a bound migliori.

- **Modelli di scheduling su macchine parallele**

3. *Il modello* $Pm|\mathrm{pmtn}|C_{\max}$

- Consideriamo il problema di processare un insieme di job su m *macchine parallele* nell'ipotesi che sia *ammessa* la *preempzione* e la funzione obbiettivo sia il *makespan*.

- La *preempzione semplifica* l'analisi del *problema*

(i) Consideriamo la seguente *formulazione lineare*

$$\min C_{\max}$$

$$
\begin{array}{lll}
\text{(a)} & \sum_{i=1}^{m} x_{ij} \geq p_j, & j = 1, \ldots, n \\
\text{(b)} & \sum_{i=1}^{m} x_{ij} \leq C_{\max}, & j = 1, \ldots, n \\
\text{(c)} & \sum_{j=1}^{n} x_{ij} \leq C_{\max}, & i = 1, \ldots, m \\
\text{(d)} & x_{ij} \geq 0, & i = 1, \ldots, m; j = 1, \ldots, n
\end{array}
$$

ove x_{ij} è l'ammontare del tempo totale speso dal job j sulla macchina i.

I vincoli (a) impongono che il job j sia processato per un tempo (almeno) pari a quello necessario per la sua esecuzione.

I vincoli (b) impongono la condizione che il tempo di processamento di ogni job non può superare il makespan.

I vincoli (c) impongono che il tempo di lavoro di ciascuna macchina non superi il valore del makespan.

Modelli di scheduling su macchine parallele

3. Il modello $Pm|\text{pmtn}|C_{\max}$ (continua)

- Poiché $C_{\max}$ è una variabile di decisione e non un elemento del vettore delle risorse del problema di PL i vincoli (b) e (c) si possono riscrivere come

$$C_{\max} - \sum_{i=1}^{m} x_{ij} \geq 0, j = 1, \ldots, n$$
$$C_{\max} - \sum_{j=1}^{n} x_{ij} \geq 0, i = 1, \ldots, m$$

- Pertanto questa formulazione è del tipo

$$\min \bar{c}\,\bar{x}$$
$$A\,\bar{x} \geq \bar{b}$$
$$\bar{x} \geq \bar{0}$$

dove il vettore $\bar{c}$ e la matrice A contengono solo elementi $0, 1$ ed il vettore $\bar{b}$ è formato da n tempi di processamento ed $(m + n)$ zeri.

Osservazione

Questo modello PL si può risolvere in tempo polinomiale ma la sua soluzione non dà in realtà una schedula perché specifica solo l'ammontare del tempo speso da ogni job j sulla macchina i.
Ovviamente questa informazione può essere usata per costruire facilmente una schedula ammissibile.

Modelli di scheduling su macchine parallele

3. Il modello $Pm|\text{pmtn}|C_{\max}$ *(continua)*

(ii) Un altro algoritmo per questo problema si basa sul seguente risultato per il limite inferiore sui valori che può assumere $C_{\max}$.

Sia il job 1 quello con il più grande tempo di processamento.

Lemma

Per il problema $Pm|\text{pmtn}|C_{\max}$ si ha che

$$C_{\max} \geq \max\left[\, p_1,\ \frac{1}{m}\sum_{j=1}^{n} p_j \right] = C^*_{\max}$$

Dimostrazione

Poiché il job 1 è quello con più grande tempo di processamento la precedente relazione ne deriva immediatamente.

Modelli di scheduling su macchine parallele

3. Il modello $Pm|\text{pmtn}|C_{\max}$ (continua)

Il seguente algoritmo permette di determinare una soluzione ammissibile di valore pari a $C^*_{\max}$ e quindi ottima.

Algoritmo

Passo 0: *a)* Se $p_1 \leq 1/m \sum p_j$ vai al passo 1. *b)* Altrimenti schedula al tempo 0 il job 1 sulla macchina 1.

Passo 1: Processa i (restanti) job su una macchina in modo da generare una qualsiasi sequenza. Il makespan che ne deriva è pari alla somma dei tempi di processamento ed è $\leq m\ C^*_{\max}$ (caso *a*) [$\leq (m-1)\ C^*_{\max}$ (caso *b*)].

Passo 2: Caso *a*): Partiziona la schedula precedente in m parti come segue: $[0, C^*_{\max}]$, $[C^*_{\max}, 2C^*_{\max}]$, …, $[(m-1)C^*_{\max}, mC^*_{\max}]$.
Caso *b*): Partiziona la schedula precedente in $m-1$ parti come segue: $[0, C^*_{\max}]$, $[C^*_{\max}, 2C^*_{\max}]$, …, $[(m-2)C^*_{\max}, (m-1)C^*_{\max}]$.

Passo 3: Considera la sequenza del *primo intervallo* come la *schedula della macchina* 1 (caso *a*) [2 (caso *b*)], la sequenza del *secondo intervallo* come la *schedula della macchina* 2 (caso *a*) [3 (caso *b*)] e così via.

Modelli di scheduling su macchine parallele

3. Il modello $Pm|\text{pmtn}|C_{\max}$ (continua)

Osservazione

La *schedula* fornita è ovviamente *ammissibile*. Infatti, poiché è ammessa la *preempzione*, parte di un job può comparire alla fine della schedula della macchina i e parte all'inizio della schedula della macchina $i+1$, e siccome ogni $p_j < C^*_{\max}$ non c'è alcun job j che risulta contemporaneamente in esecuzione su due macchine. Pertanto, una tale schedula è ammissibile.
Inoltre, poiché tale schedula ha $C_{\max} = C^*_{\max}$ questa è anche *ottima*.

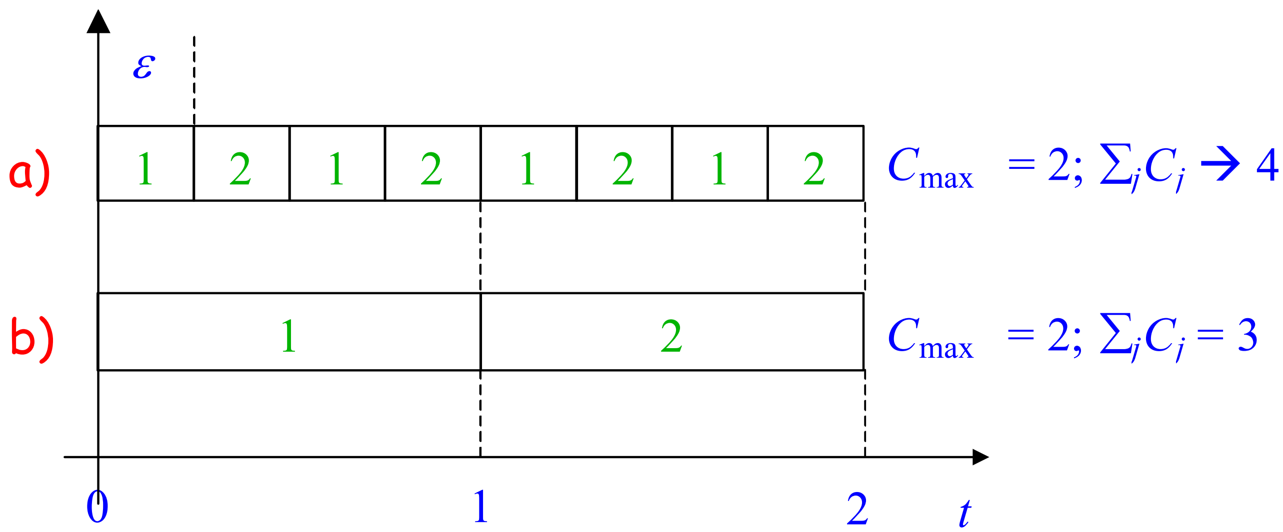
(iii) Un'altra regola per il problema in esame è:
Longest Remaining Processing Time first (LRPT)
La schedula risultante è la *versione preemptiva* della schedula *LPT*.
Questa regola è più di interesse teorico che pratico. Infatti, il numero di preempzioni necessarie nel caso deterministico può essere infinito.

Esempio

Consideriamo il caso $n = 2$, $p_j = 1$, per $j = 1, 2$ e $m = 1$.
Con la strategia *LRPT* (a) i job ruotano continuamente sulla macchina con periodicità ε. Il *makespan* è 2 ed ovviamente indipendente dalla schedula. Il *completion time totale* $\sum_j C_j \to 4$, per $\varepsilon \to 0$, mentre nel caso *non preemptivo* (b) è pari a 3.

Modelli di scheduling su macchine parallele

3. Il modello $Pm|\mathrm{pmtn}|C_{\max}$ *(continua)*



a) Schedula *preemptiva* secondo la regola LRPT.

b) Schedula *non preemptiva*

Modelli di scheduling su macchine parallele

4. Il modello $Pm||\Sigma C_j$

- Consideriamo m macchine in parallelo ed n job.
- L'obbiettivo è minimizzare il tempo di completamento totale. Anche in tal caso vale il seguente

  Teorema

  La regola *Shortest Processing Time first* (*SPT*) dà una soluzione ottima per $Pm||\Sigma C_j$.

  Dimostrazione

  È analoga a quella del caso di macchina singola.

- Secondo questa regola il job più piccolo è assegnato alla macchina 1 al tempo $t = 0$, il secondo più piccolo alla macchina 2 e così via fino al m-mo job più piccolo.

  A questo punto si ricomincia assegnando il job $(m+1)$-mo più piccolo alla macchina 1, il job $(m+2)$-mo più piccolo alla macchina 2 e così via.

- È facile verificare che la *schedula SPT* corrisponde ad un assegnamento ottimo dei job.
- La regola *SPT non può* però *essere generalizzata* al *caso pesato* come nel caso di macchina singola. Tuttavia la regola *WSPT* è una buona euristica per $Pm||\Sigma w_jC_j$.

  Si può dimostrare che $\dfrac{\sum_j w_j C_j(\text{WSPT})}{\sum_j w_j C_j(\text{OPT})} \leq \dfrac{1}{2}\left(1+\sqrt{2}\right)$

Modelli di scheduling su macchine parallele

4. Il modello $Pm||\Sigma C_j$ (continua)

Osservazione

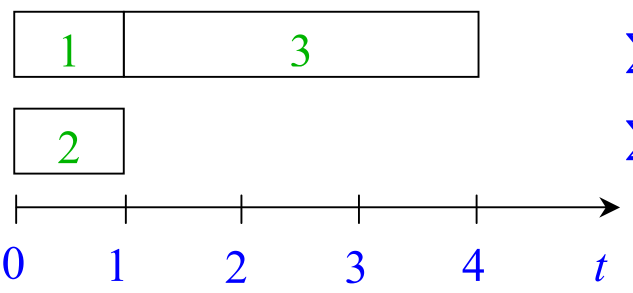
- La regola *SPT* in questo modello non esaurisce tutte le soluzioni ottime.

Esempio

- Consideriamo la seguente istanza con $m = 2$ e $n = 3$:

job	1	2	3
p_j	1	1	3
w_j	1	1	3

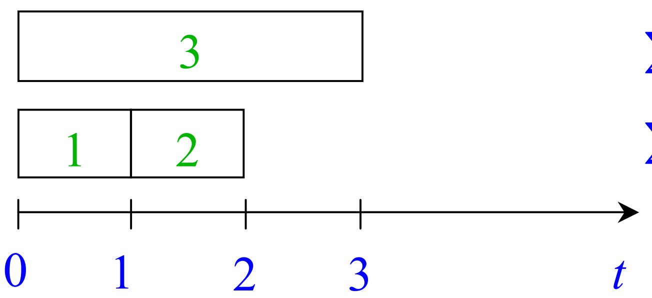
- La regola *SPT* dà luogo alla sequenza:



$\Sigma_j C_j = 6$

$\Sigma_j w_j C_j = 14$

- Consideriamo la sequenza seguente che *non* è *SPT*



$\Sigma_j C_j = 6$

$\Sigma_j w_j C_j = 12$

- Si vede che esistono più sequenze ottime per $\Sigma_j C_j$ e non solo la *SPT*. Inoltre, le due sequenze precedenti sono entrambe *WSPT* ma non sono entrambe ottime.