Sequenziamento a minimo costo di commutazione in macchine o celle con costo lineare e posizione “home”

(In generale il metodo di ottimizzazione presentato in questo file trova la seq. a costo minimo per un insieme di lavori, anche complessi, che utilizzano una stessa risorsa, anche capace di eseguire in parallelo le operazioni di due lavori diversi)

Sequenziamento su una macchina
minimo costo di commutazione

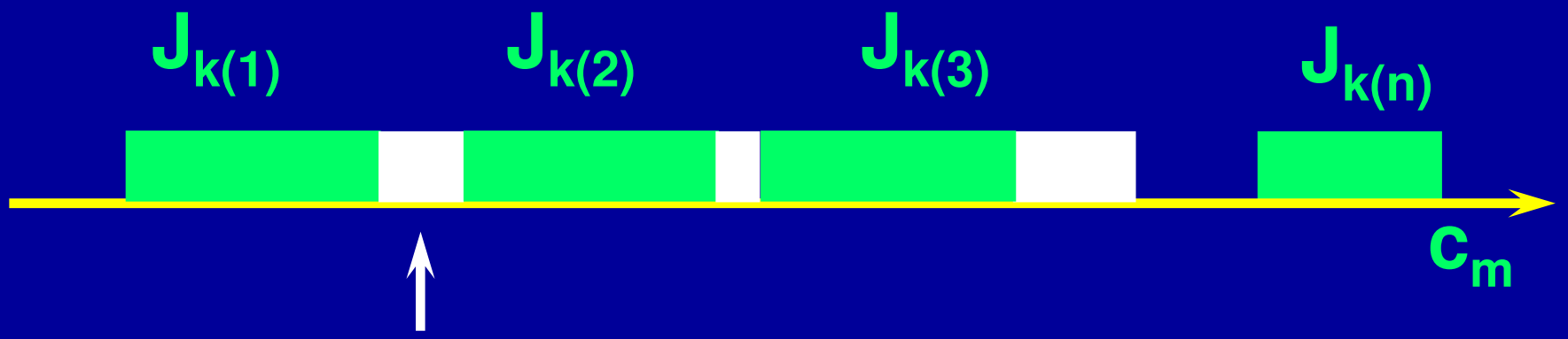
Il costo di commutazione può essere il tempo (allora si minimizza il tempo di completamento totale) o altro

Sequenziamento su una macchina minimo costo di commutazione

Cambio di utensile:

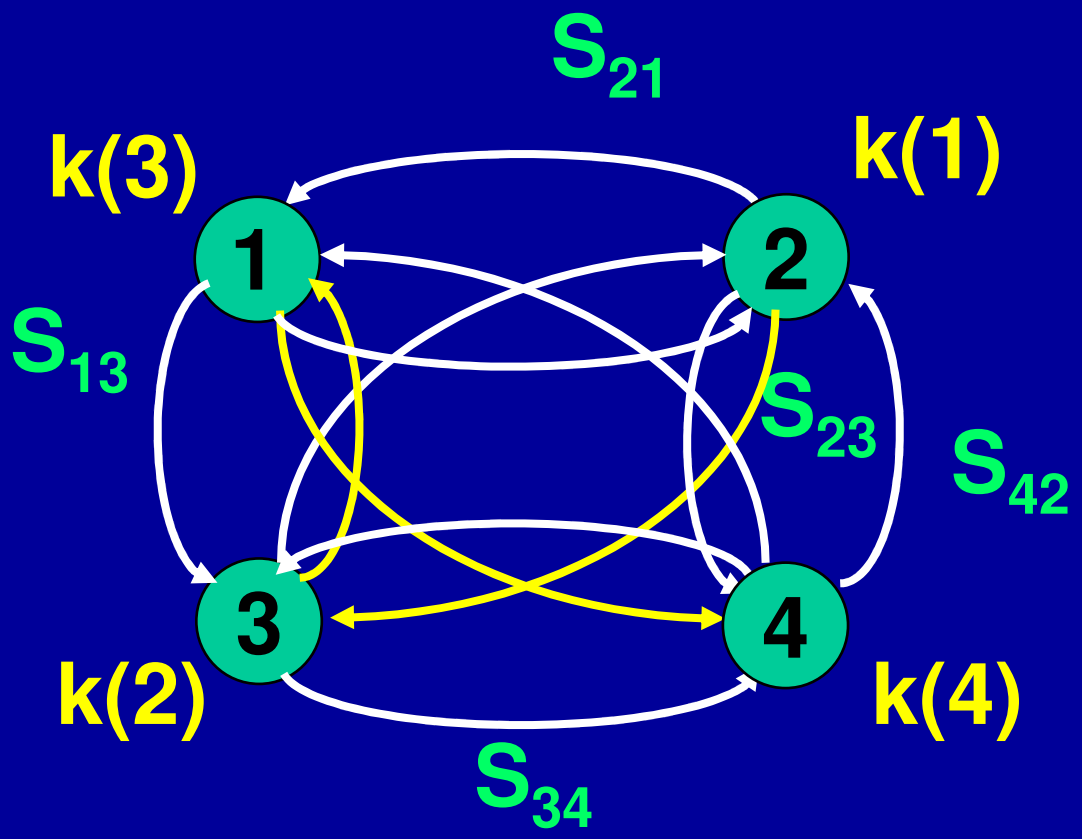
- il tempo di commutazione spesso è fisso

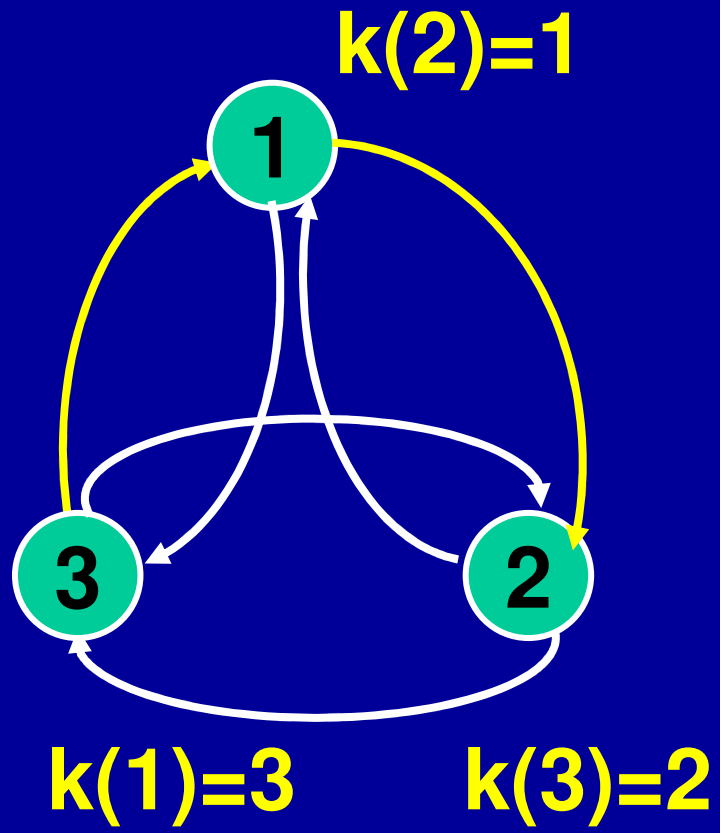
- si ha un costo se gli utensili di due lavori successivi sono diversi



tempi di commutazione

Comunque i tempi di processamento dei singoli lavori non hanno rilevanza, quindi si può rappresentare il problema con un grafo che ha per nodi i lavori e per archi le commutazioni, pesate con il loro costo di commutazione da J_i a J_j : s_{ij}





Il problema è trovare un cammino di costo minimo che includa tutti i nodi una sola volta (cammino hamiltoniano minimo): noto problema difficile

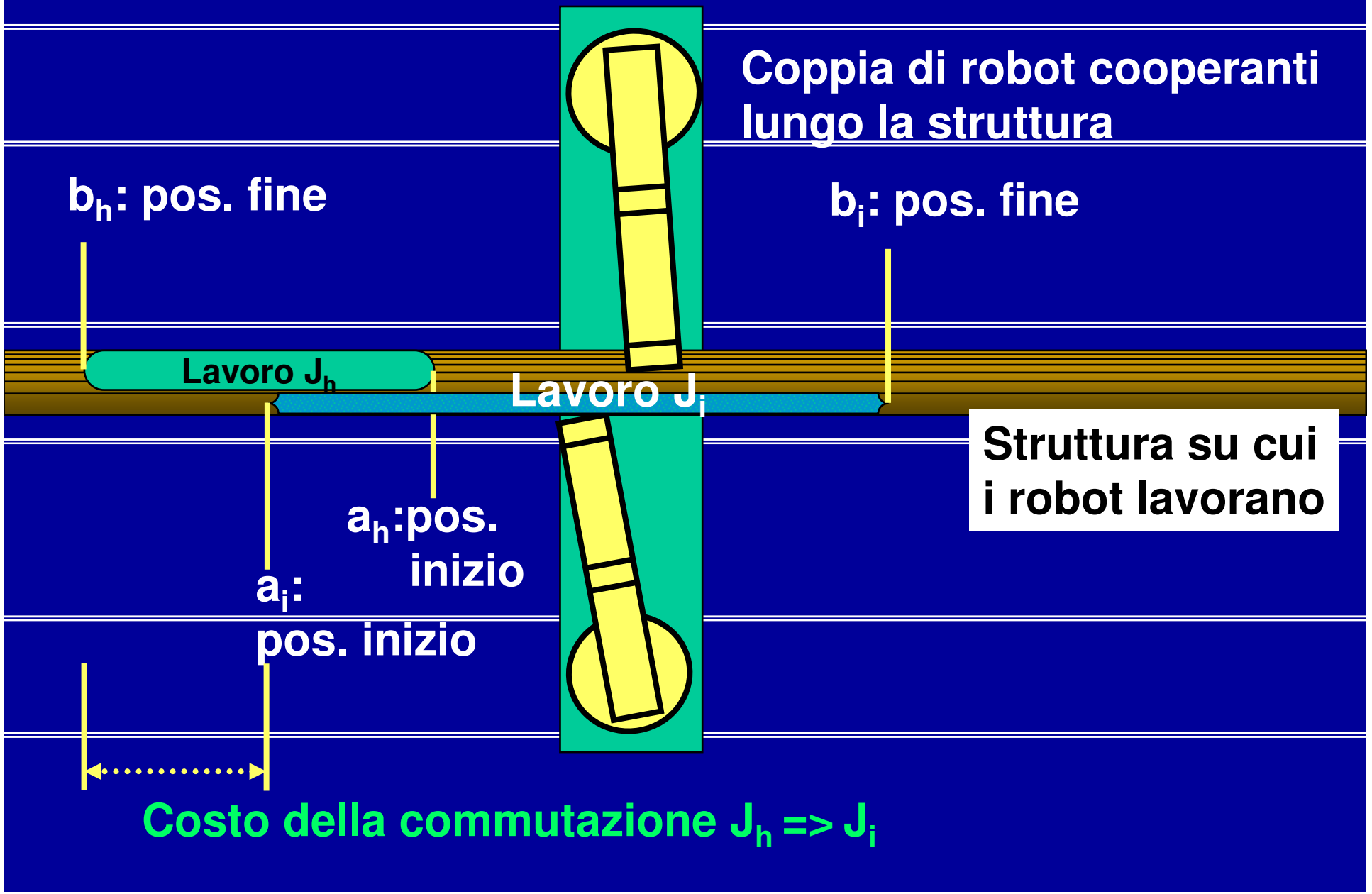
Sequenziamento su una macchina o cella con minimo costo di commutazione

Gilmore e Gomory (1964) : Costo "proporzionale" al cambio di stato della macchina o cella, necessario per commutare.

Esempi:

- **forno: stato: temperatura**
- **cella robotizzata: stato: posizione dei robot**

Costo “proporzionale” al cambio di stato della cella



Sequenziamento su una macchina
minimo costo di commutazione

La soluzione è “facile” se, associati ad ogni operazione due stati della macchina (Gilmore e Gomory 1964):

a_k stato necessario per iniziare l’operazione

b_k stato finale dopo l’operazione k

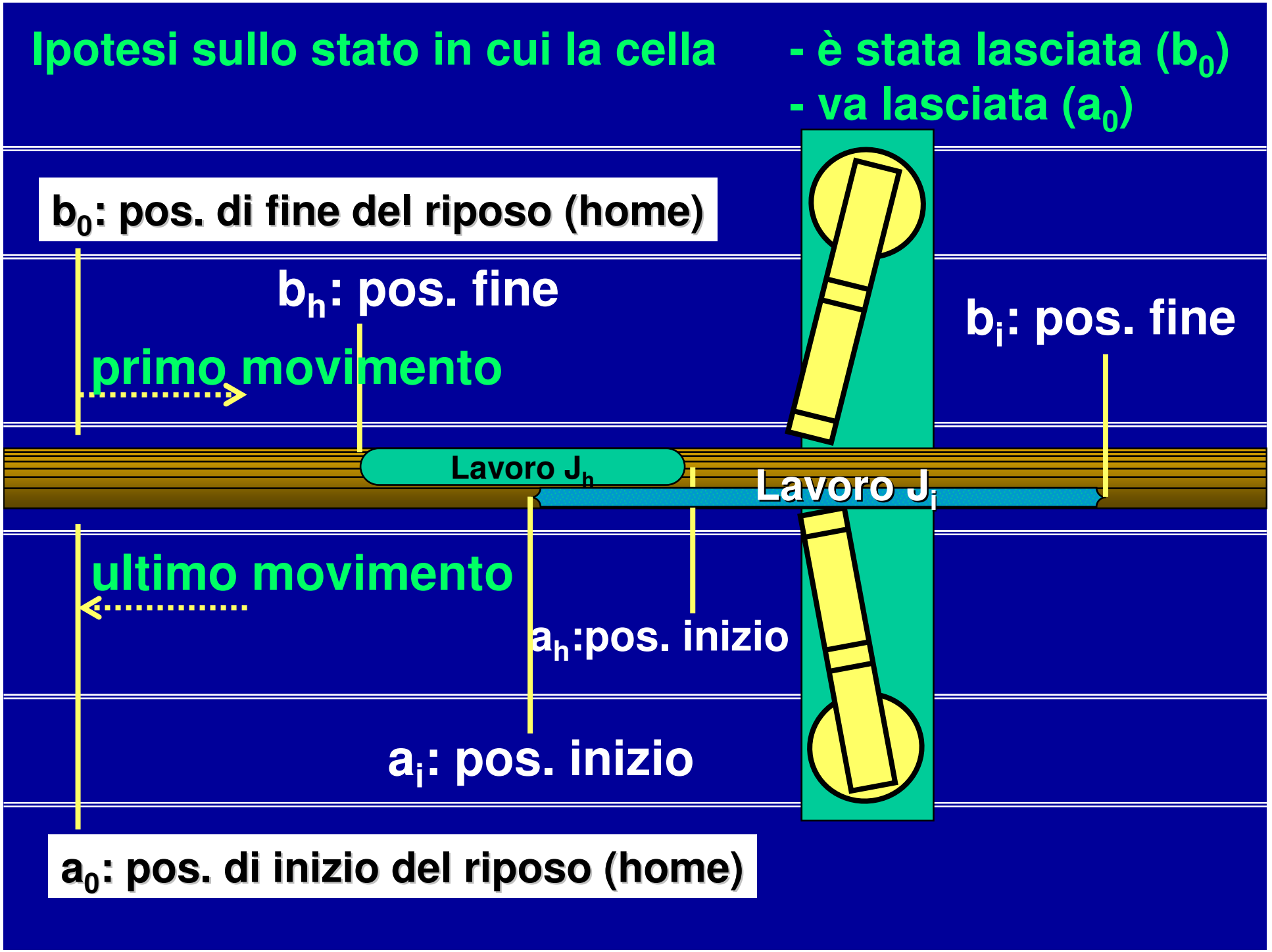
si ha (da op. k a op j):

$$s_{kj} = | b_k - a_j |$$

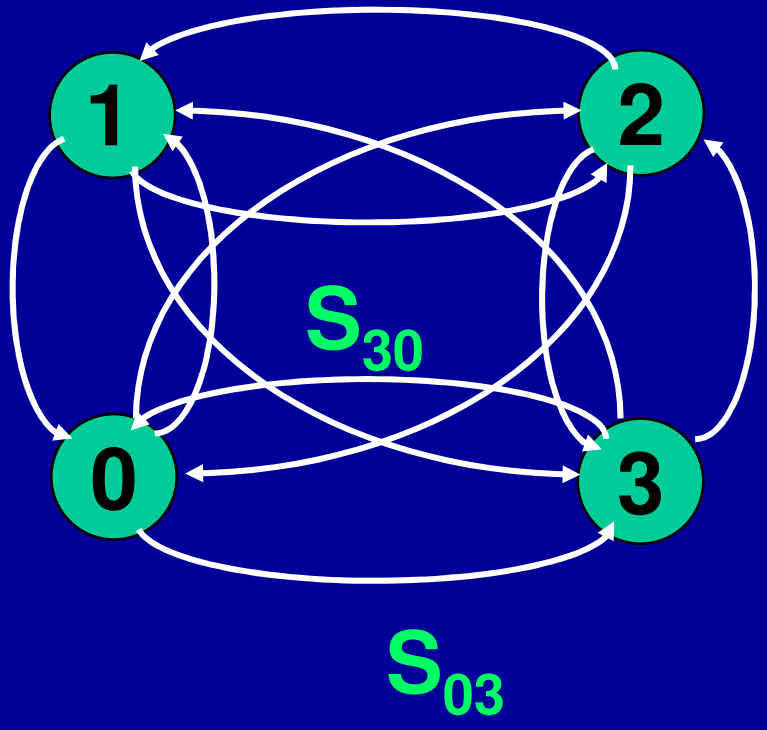
L'algoritmo di Gilmore-Gomory è applicabile nell'ipotesi che siano assegnati:

- stato in cui è stata lasciata la macchina, dopo i lavori: a_0 (inizio del riposo)

- stato in cui deve essere la macchina, prima dei lavori: b_0 (fine del riposo)


Ipotesi sullo stato in cui la cella
- è stata lasciata (b_0)
- va lasciata (a_0)
b_0: pos. di fine del riposo (home)
b_h: pos. fine
primo movimento
b_i: pos. fine
Lavoro J_h
Lavoro J_i
ultimo movimento
a_h:pos. inizio
a_i: pos. inizio
a_0: pos. di inizio del riposo (home)

Nel grafo dei lavori si aggiunge un lavoro J_0 con archi s_{0k} (percorso se k è la prima op.) e s_{k0} (percorso se k è l'ultima op.)

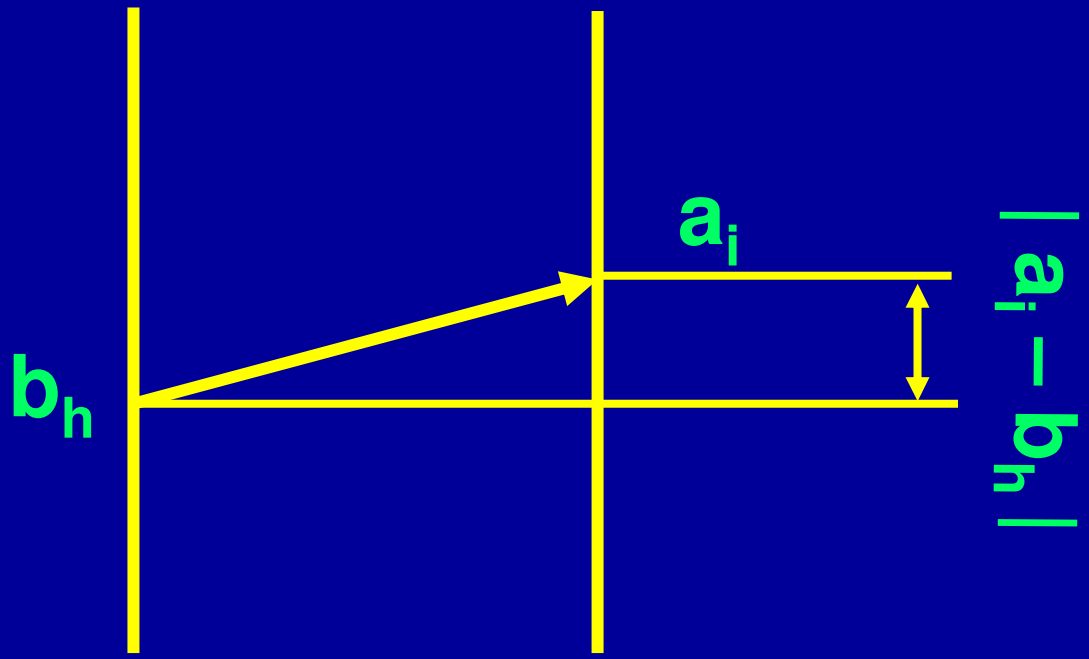


Ciclo hamiltoniano minimo

$S_{kj} = | b_k - a_j |$

$$\min_{S} \sum_{kj \in S} | b_k - a_j |$$

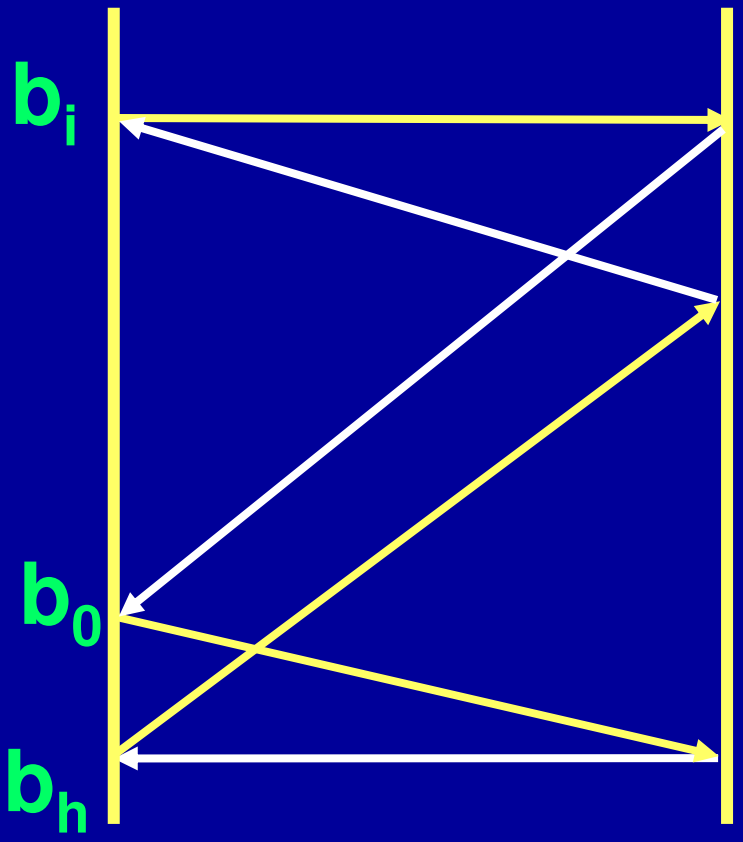
Costo di una commutazione



Costo $J_h \longrightarrow J_i$

$$\underset{S}{\text{Min}} \sum_{jk \,\varepsilon\, S} | b_j - a_k |$$

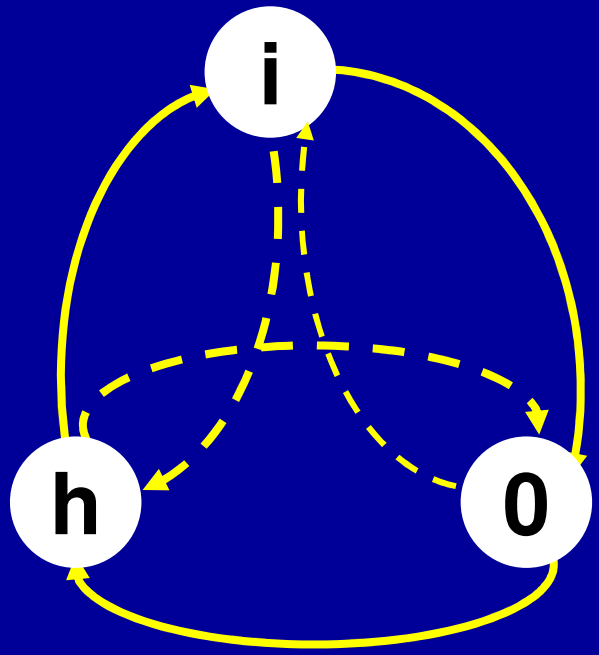
Se, partendo da 0, si punta all'inizio più vicino, si può avere un ciclo o più cicli e/o incroci



$a_0 = a_{\varphi\,(i)}$

$a_i = a_{\varphi\,(h)}$

$a_h = a_{\varphi\,(0)}$

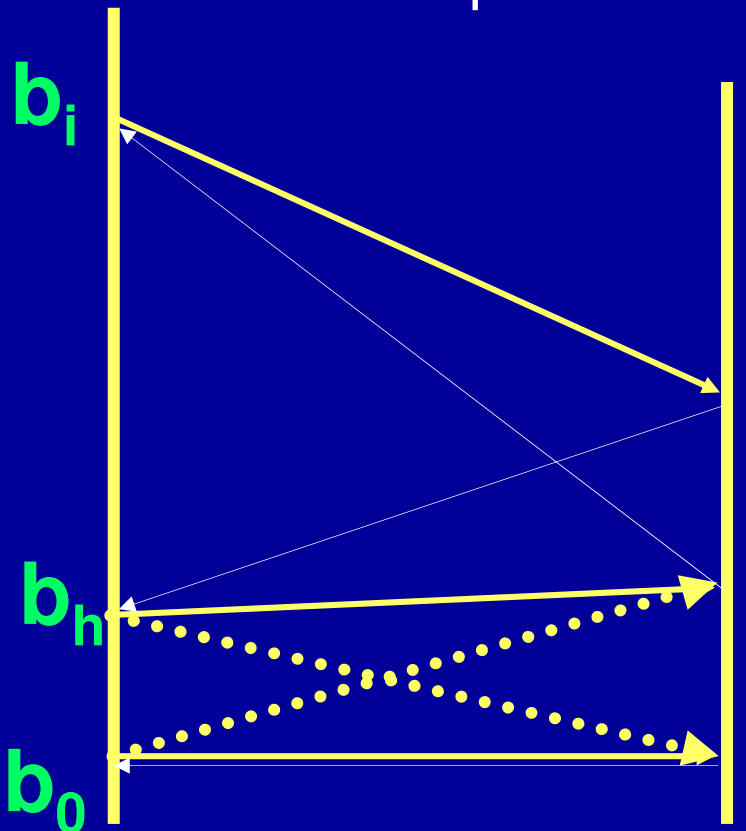


Gialle: commutazioni bianche: operazioni

Una qualsiasi sequenza può essere descritta da un puntatore $\varphi(k)$ che dà l'indice del lavoro successivo al lavoro k nella sequenza

$\varphi^*(k)$: nessuna intersezione

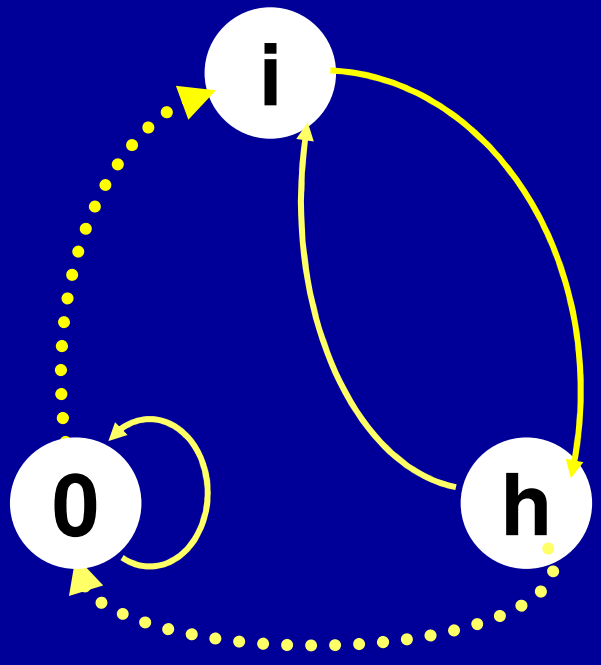
se φ^* non definisce un sol ciclo si hanno più sequenziamenti periodici che non risolvono il problema e bisogna unire i cicli



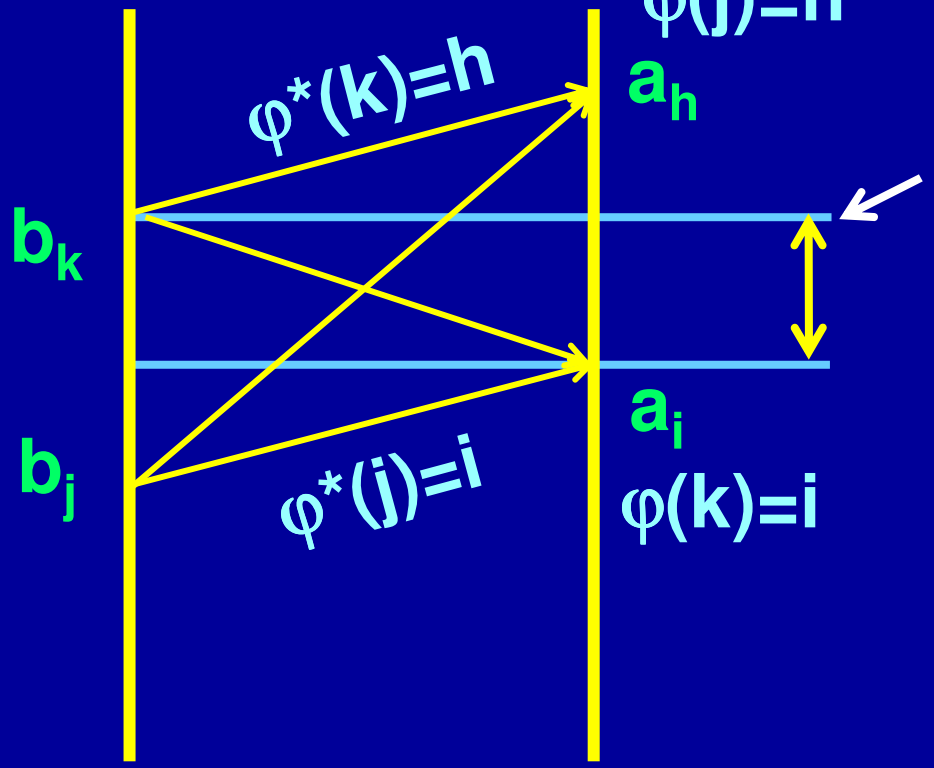
$a_h = a_{\varphi(i)}$

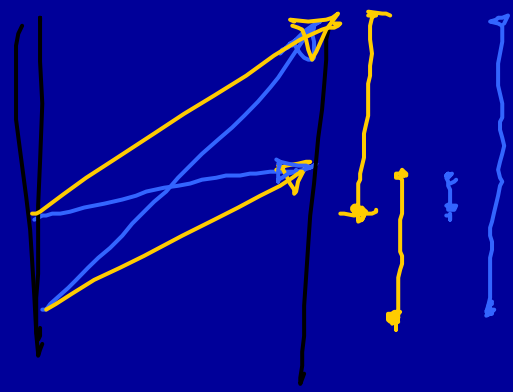
$a_i = a_{\varphi(h)}$

$a_0 = a_{\varphi^*(0)}$

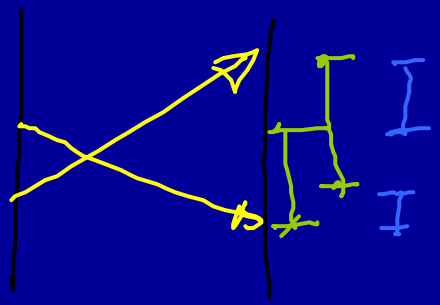


Unire i cicli corrisponde a intersezioni delle frecce di commutazione, con un maggior costo totale di commutazione



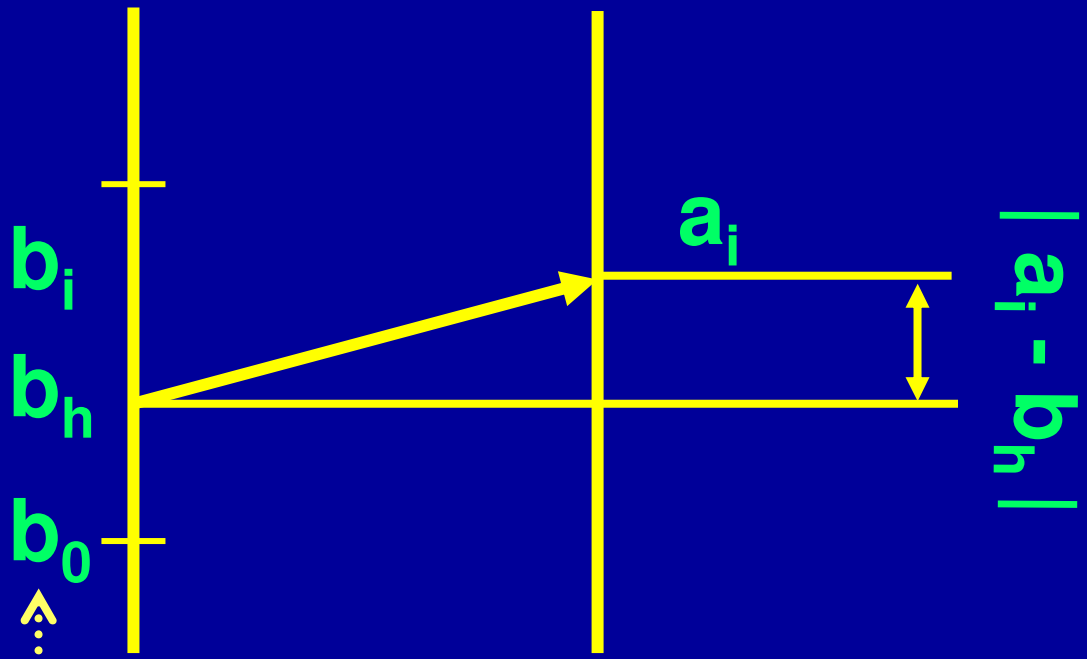


Costo di un incrocio:
il doppio della intersezione tra
il segmento staccato dalle b e
quello staccato dalle a
(intersezione nulla: costo nullo)



ALGORITMO

Sequenziamento su una macchina minimo costo di commutazione



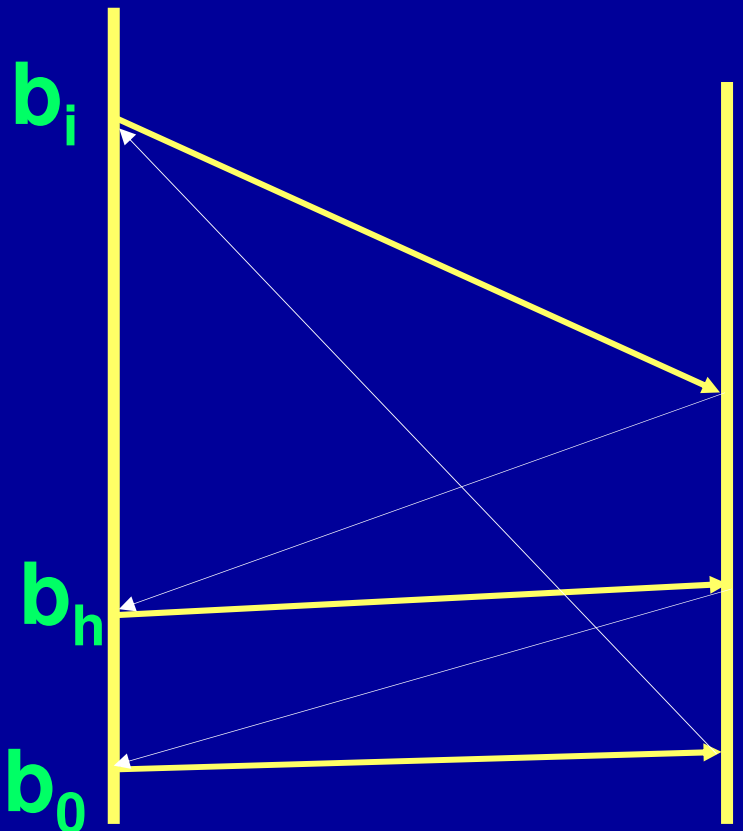
Costo $J_h \rightarrow J_i$

Si riordinano gl'indici secondo le b crescenti : $b_0 \leq b_1 \leq \ldots \leq b_n$

$\{ i = \varphi^*(j) \}$:

a_i : è il (j+1)-esimo più piccolo degli a

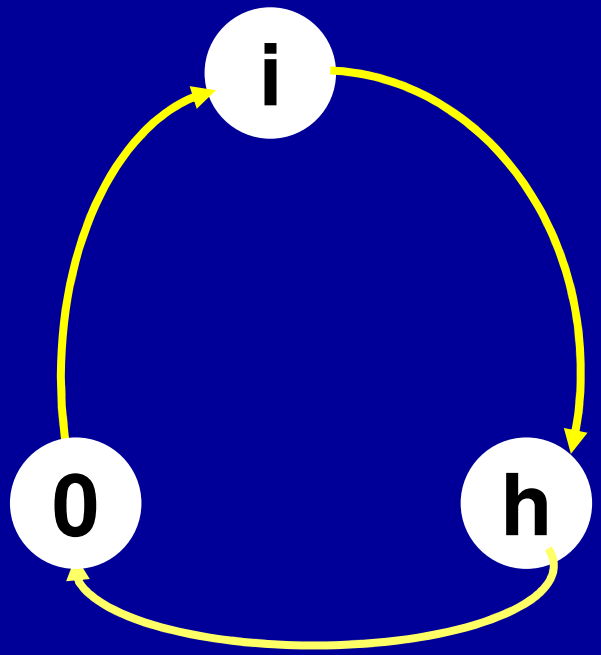
Se si ha un sol ciclo si ottiene $\min_{j,i} \Sigma \mid b_j - a_i \mid : \{i = \varphi^*(j)\}$



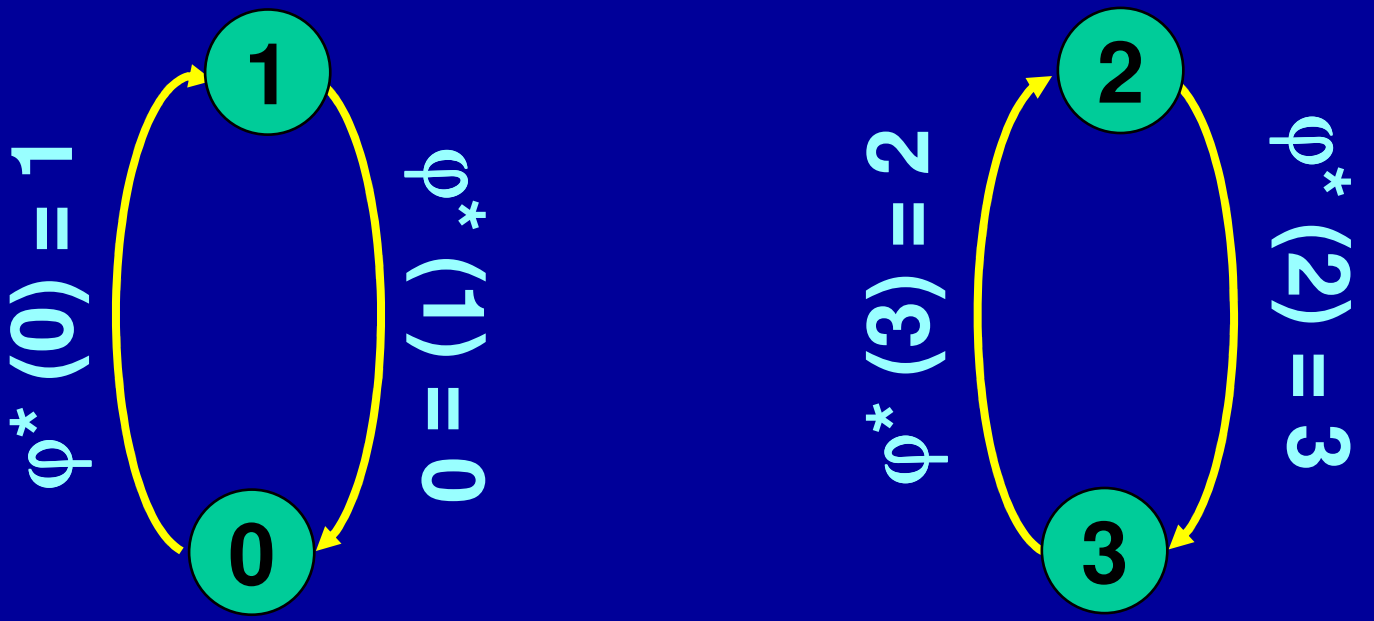
$a_h = a_{\varphi^*(i)}$

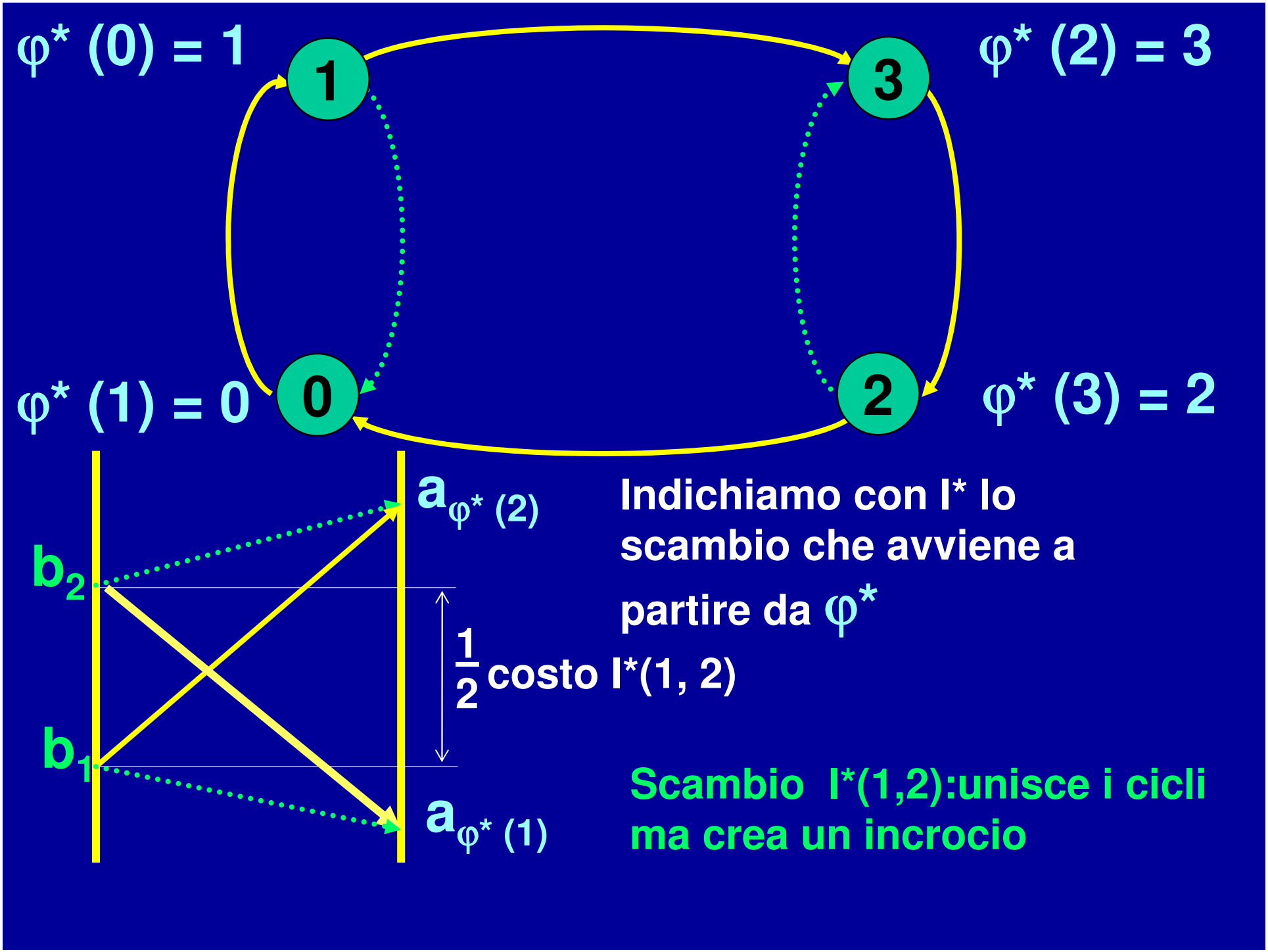
$a_0 = a_{\varphi^*(h)}$

$a_i = a_{\varphi^*(0)}$

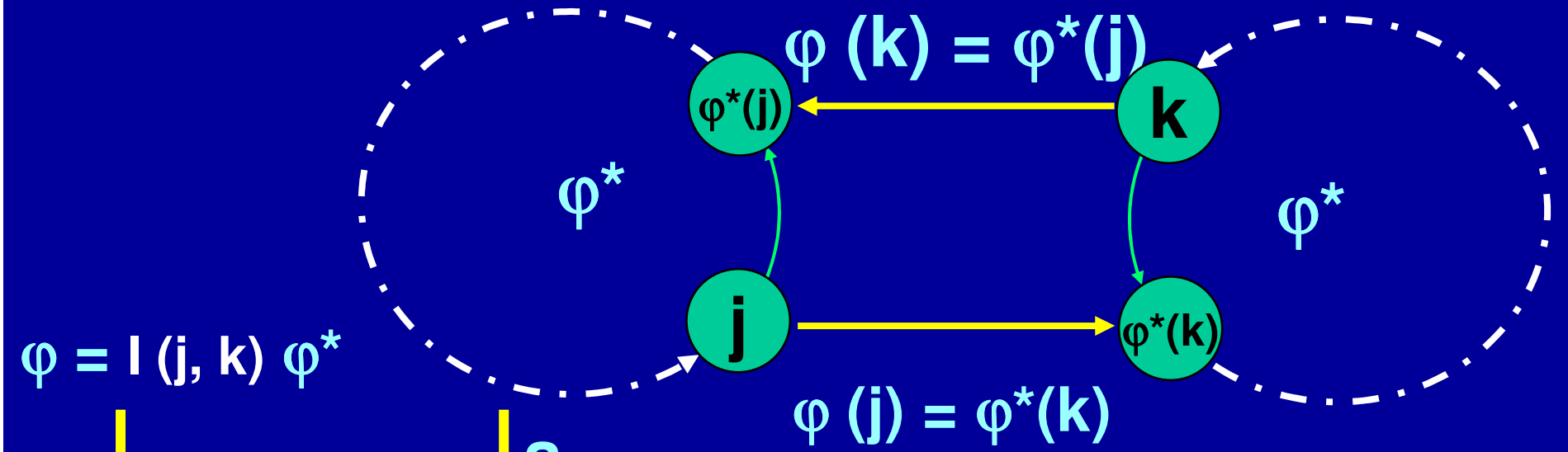


Si può non trovare un solo ciclo

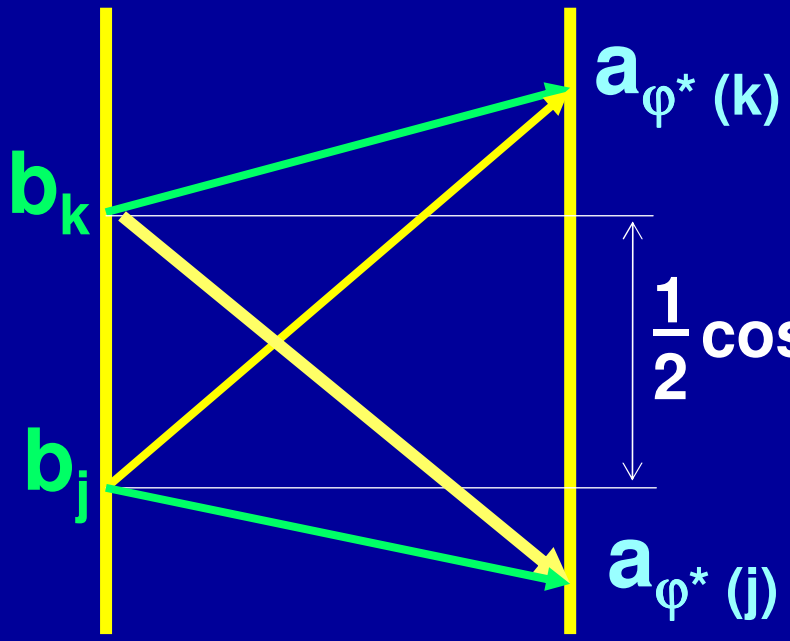



φ* (0) = 1
φ* (2) = 3
φ* (1) = 0
φ* (3) = 2
1
3
0
2
$a_{\varphi^*(2)}$
b_2
b_1
$a_{\varphi^*(1)}$
$\frac{1}{2}$ costo l*(1, 2)
Indichiamo con l* lo scambio che avviene a partire da φ*
Scambio l*(1,2):unisce i cicli ma crea un incrocio

Scambio I*(j,k), con j e k in cicli diversi



$\varphi = I(j, k)\ \varphi^*$



Come detto, unisce i cicli ma crea un incrocio

Intuizione: **Occorre unire i cicli minimizzando il costo degli scambi**

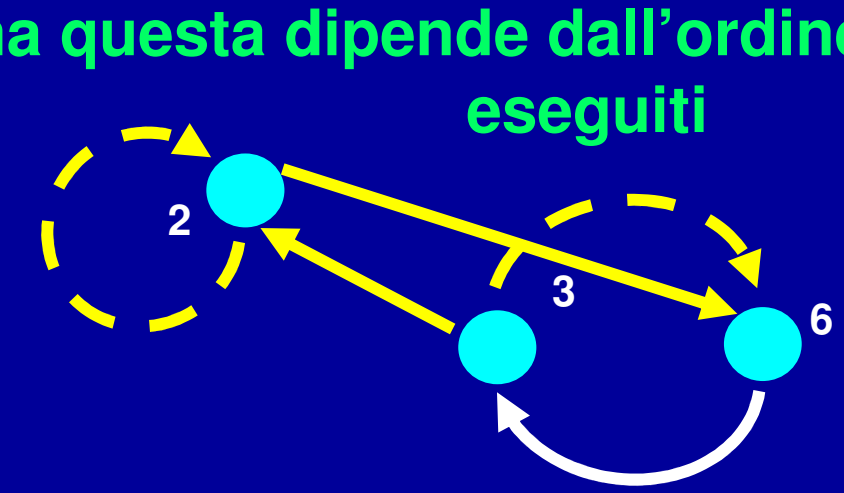
Occorre fare tre considerazioni

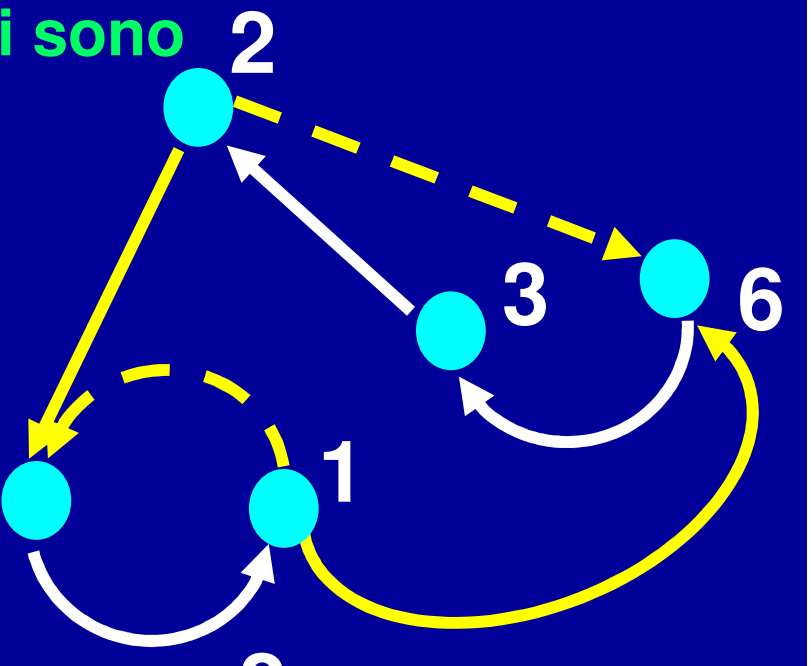
1. **gli scambi non definiscono la sequenza, ma questa dipende dall'ordine in cui sono eseguiti**

2. **basta considerare gli scambi tra i puntatori di due operazioni successive, nell'ordinamento crescente delle b**

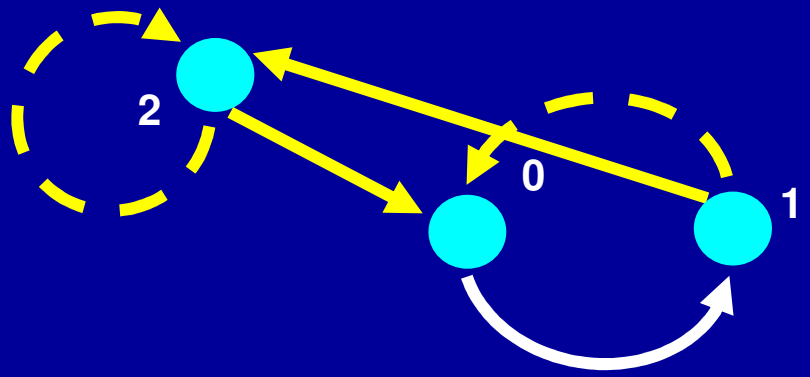
3. **uno scambio può influenzare il costo dei contigui (non gli altri)**

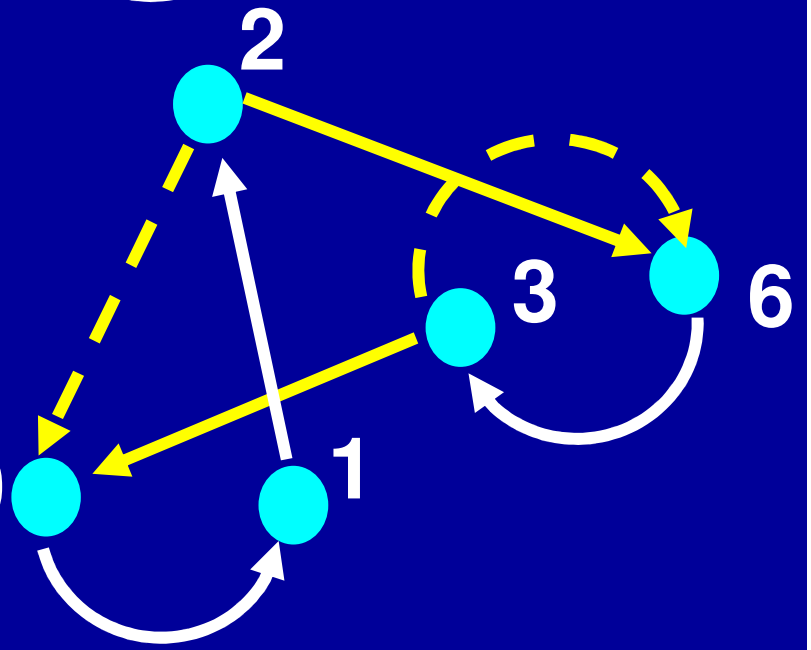
1. gli scambi non definiscono la sequenza, ma questa dipende dall'ordine in cui sono eseguiti





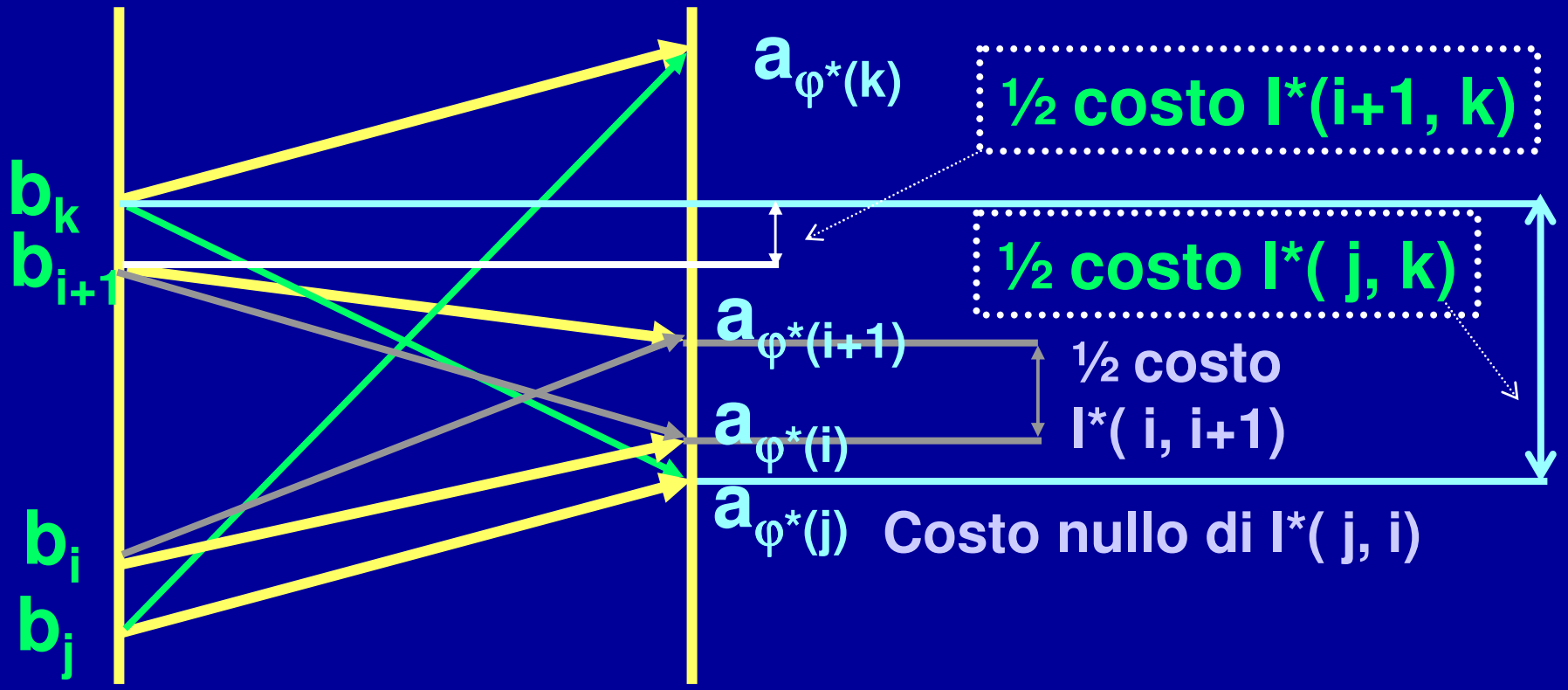
I (1, 2) I (2, 3) φ*





I (2, 3) I (1, 2) φ*

j e k in cicli diversi:



$$l^*(j,k) \geq \sum_{i=j=>k-1} l^*(i,i+1)$$

$$l^*(j,k) \geq \sum_{i=j=>k-1} l^*(i,i+1)$$

2. basta considerare gli scambi tra i puntatori di due operazioni successive, nell'ordinamento crescente delle b

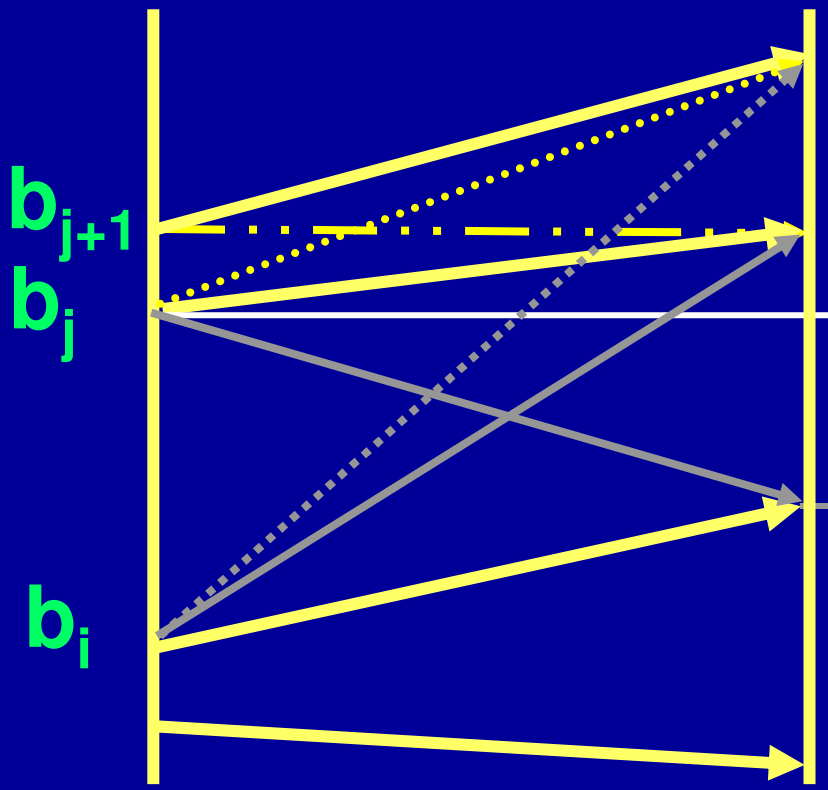
Se i nodi con indice compreso tra j e k, estremi inclusi, appartengono solo a due cicli diversi, conviene unirli scambiando tra due con indici successivi.

Se ci sono più di due cicli e scambiamo solo quando i e i+1 appartengono a cicli diversi, allora uniamo i cicli a cui appartengono j e k, ma anche tutti i cicli diversi a cui appartengono i nodi tra j e k: anche se fossero tutti autocicli, la somma dei costi degli scambi non sarebbe maggiore

Comunque,, gli scambi successivi potrebbero influenzarsi a vicenda e bisogna indagare, come si vedrà al punto 3, che segue

3. uno scambio può influenzare il costo dei contigui (non gli altri)

Se $a_{\varphi^*(j)} \geq b_j$ uno scambio $l^*(j, j+1)$ non altera il costo di altri contigui inferiori con $b_i < b_j$



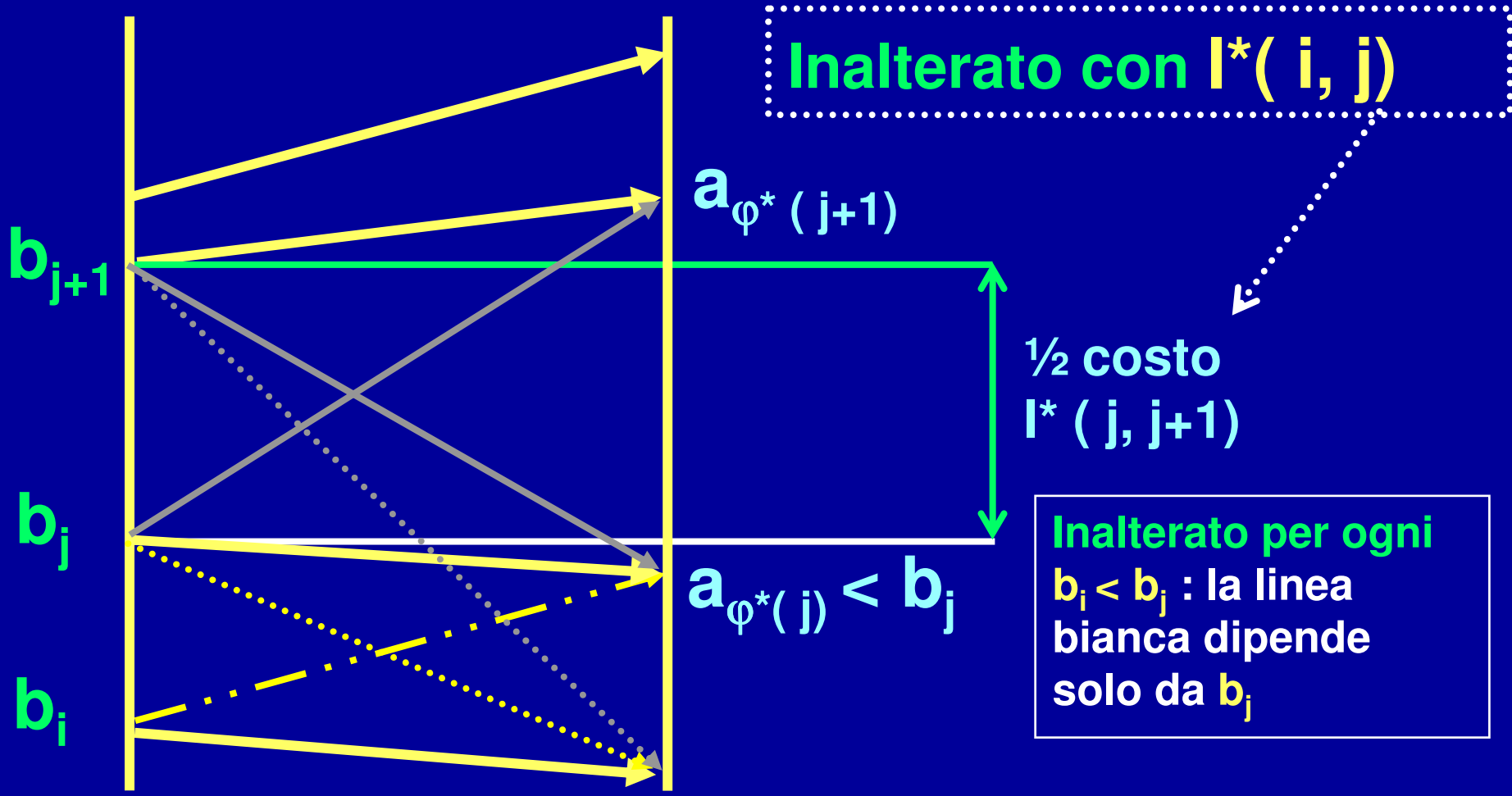
Inalterato per ogni $b_i < b_j$: la linea bianca dipende solo da b_j

½ costo $l(i, j)$

costo $l^*(j, j+1) =$

$2 \text{ Max } [0, \min (b_{j+1}, a_{\varphi^*(j+1)}) - \text{Max } (b_j, a_{\varphi^*(j)})]$

Se $a_{\varphi^*(j)} < b_j$ il costo di uno scambio $l^*(j, j+1)$ non è alterato da $l^*(i, j)$, con $b_i < b_j$



Se ci sono cicli diversi:
Si calcolano e si ordinano gli n costi di scambio
$l^*(j, j+1)$ per $j = 0, \ldots, n-1$
Si applica lo scambio di minor costo con J_j e J_{j+1} in cicli diversi e si continua fino a che non si ottiene un ciclo unico

In quale ordine inserire gli scambi trovati per avere la sequenza ottima?

La considerazione 3. suggerisce la via

Ogni scambio da fare di tipo

$$A: a_{\varphi^*(j)} \geq b_j$$

non influenzerà un successivo contiguo più basso.

Questo suggerisce di eseguirli nell'ordine dall'alto.

Se ci saranno scambi ancora da fare, questi saranno di tipo

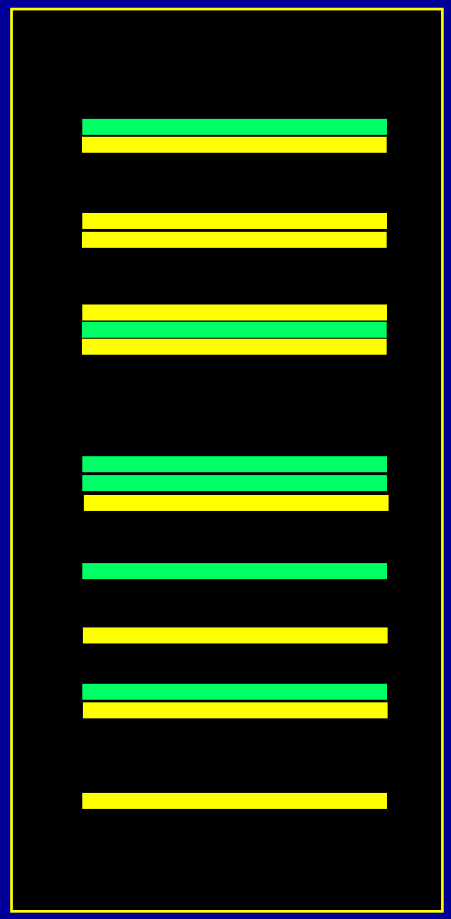
$$B: a_{\varphi^*(j)} < b_j$$

Dopo quelli di tipo A, ciascuno andrà eseguito nell'ordine dal basso:

in tal modo non sarà influenzato da un precedente contiguo più basso

CLASSI DI INTERCAMBI

- A: non influenzano il contiguo inferiore
- B: non sono influenzati dall'inferiore



Gli scambi di tipo A: $b_j \leq a_{\varphi^*(j)}$ possono essere eseguiti a partire dall'alto.

Gli scambi residui di tipo B: $a_{\varphi^*(j)} < b_j$ possono essere eseguiti nell'ordine dal basso.

Ogni scambio non sarà influenzato° né da uno precedente contiguo più basso di tipo A o B, né da uno più alto di tipo A

°significa che il costo dello scambio rispetto a φ^* è uguale a quello rispetto a φ^* modificato dallo scambio contiguo

3.6 Algoritmo di Gilmore e Gomory per il minimo costo di commutazione

Algoritmo di Gilmore e Gomory (minimo costo di commutazione)

Passo 1

Si riordinano gl'indici :

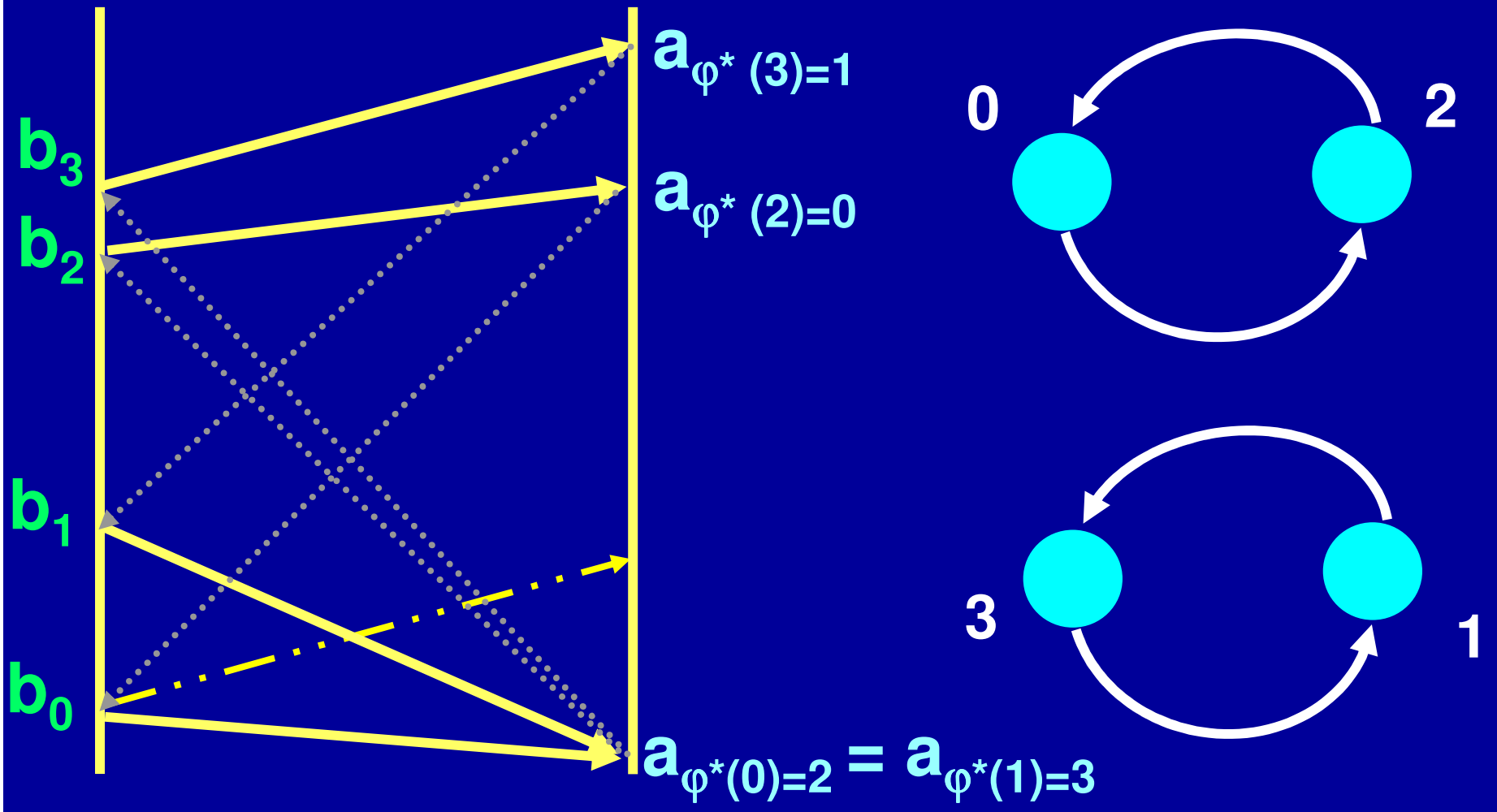
$$b_j \leq b_{j+1} \quad j = 0,\ldots,n-1$$

Si definiscono le permutazioni °

$$\varphi^*(j) : a_{\varphi(j)} \leq a_{\varphi(j+1)} \qquad j = 0, \ldots, n-1$$

° infatti φ^* può non essere unica e a ciascuna può corrispondere una configurazione di cicli diversa

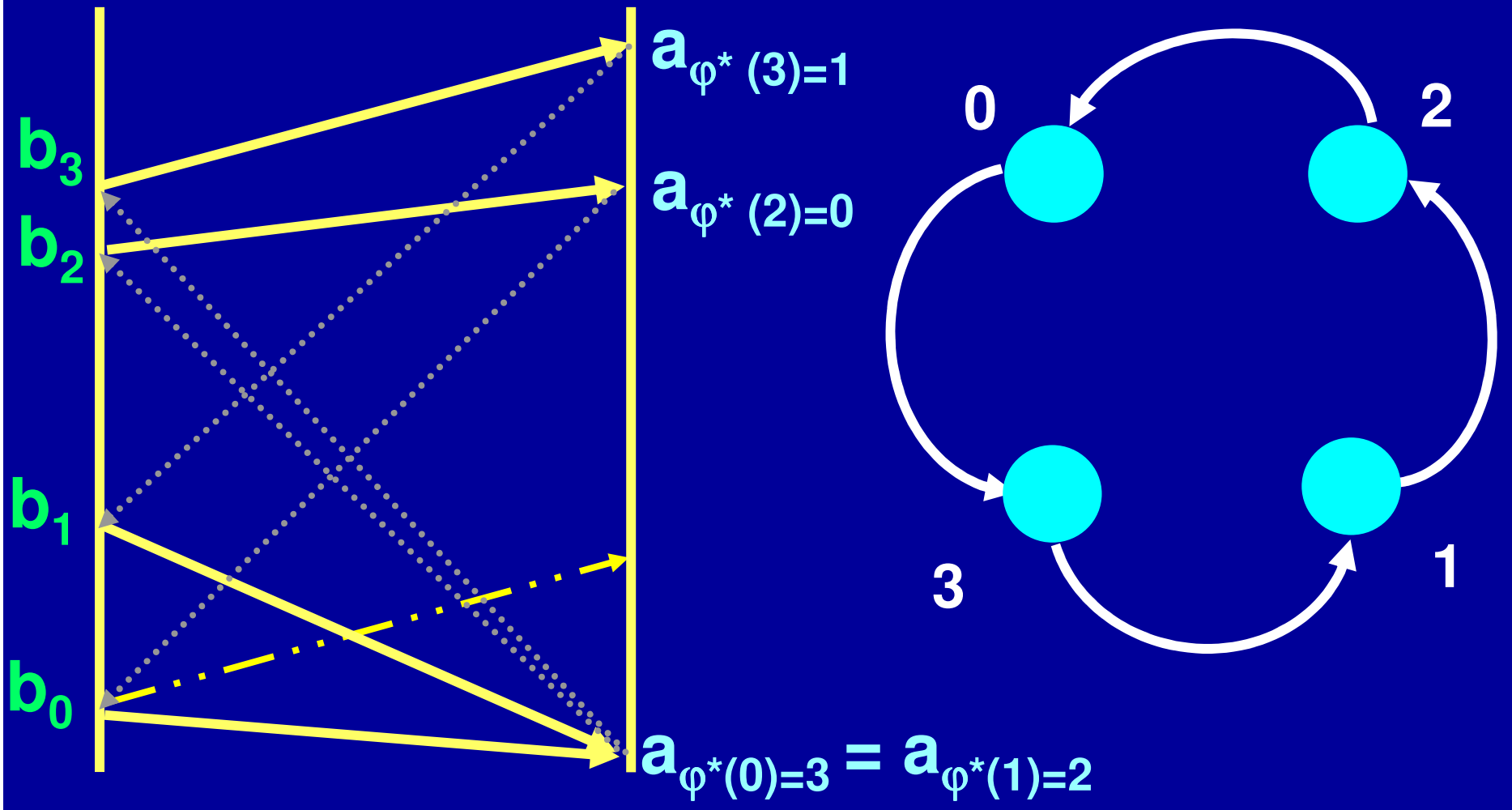
° infatti φ* può non essere unica e a ciascuna può corrispondere una configurazione di cicli diversa



Due cicli °: 0,2 – 1,3

°lo scambio (0,1), a costo zero, unisce i due cicli

° infatti φ* può non essere unica e a ciascuna può corrispondere una configurazione di cicli diversa



Un solo ciclo °: 0, 3, 1, 2

°ottenuto direttamente

Passo 2

Se una $\varphi^*(j)$ definisce un sol ciclo STOP: una S ottima è definita da quella φ^*; altrimenti si continua

Passo 3

A partire da una permutazione φ^* che dia luogo al minor numero di cicli °, si calcolano i costi cl^* degli scambi $l^*(j, j+1)$ per $j = 0, \ldots, n-1$ quando J_j e J_{j+1} appartengono a cicli diversi (se appartengono allo stesso ciclo lo scambio lo separa!)

°se se ne prende una qualsiasi il risultato finale non cambia, ovviamente

Passo 4

Si seleziona, tra quelli del passo 3, lo scambio a costo minimo: si va al passo 5

Passo 4'

Si seleziona, tra quelli del passo 3, lo scambio a costo minimo che collega due cicli, originari/o e/o nuovi/o: si va al passo 5

Passo 5

Se si sono collegati tutti i cicli definiti da φ^* : STOP; altrimenti si riprende il passo 4' (fino a che non si sono collegati tutti i cicli)

Passo 6

Si dividono gli scambi selezionati in due gruppi:

$A := [l^*(j,j+1): b_j \leq a_{\varphi^*(j)}]$: ν elementi;

$B := [l^*(j,j+1): b_j > a_{\varphi^*(j)}]$: $m-\nu$ elementi;

se $\nu > 0$: $j(i)$: $l^*(j(i),j(i)+1) \in A$ con $i=1,\ldots,\nu$
e $b_{j(i)} > b_{j(i+1)}$ con $i=1, \ldots, \nu-1$

se $\nu < m$: $j(i)$: $l^*(j(i),j(i)+1) \in B$ con $i=\nu+1,\ldots,m$
e $b_{j(i)} < b_{j(i+1)}$ con $i= \nu +1, \ldots, m-1$

Passo 7

La permutazione φ^{**} ottima (che dà la sequenza ottima) si ottiene da φ^{*} applicando gli scambi
l (j(i), j (i)+1)
per valori crescenti di i = 1, …, m

Esercizio: scrivere un programma in C++ per l'algoritmo

La complessità cresce con il quadrato del numero dei lavori:

complessità Ordine di n^2 : $0(n^2)$

Dimostrazione

φ^* dà l'ottimo se definisce un sol ciclo;

altrimenti φ^{**} ha un incremento di costo pari alla somma dei costi dei singoli scambi, perché questi non si influenzano;

Gilmore e Gomory hanno dimostrato che, scegliendo scambi che **si influenzano** non si può abbassare l'incremento totale di costo.

ESERCIZIO

Una macchina deve processare un insieme di job, tutti disponibili all'istante iniziale.
In particolare ad ogni job j sono assegnati i parametri a_j e b_j, che indicano lo stato di inizio e di fine della macchina, cui è legato, linearmente,
il costo di commutazione.

Trovare il sequenziamento che minimizzi il costo totale di commutazione.

Il tempo di set-up necessario per poter processare il job k dopo il job j e' $|a_k - b_j|$.
I job ed i relativi parametri sono espressi nella seguente tabella:

job	0	1	2	3	4	5	6
b_j	1	15	26	40	3	19	31
a_j	7	16	22	18	4	45	34

Passo 1

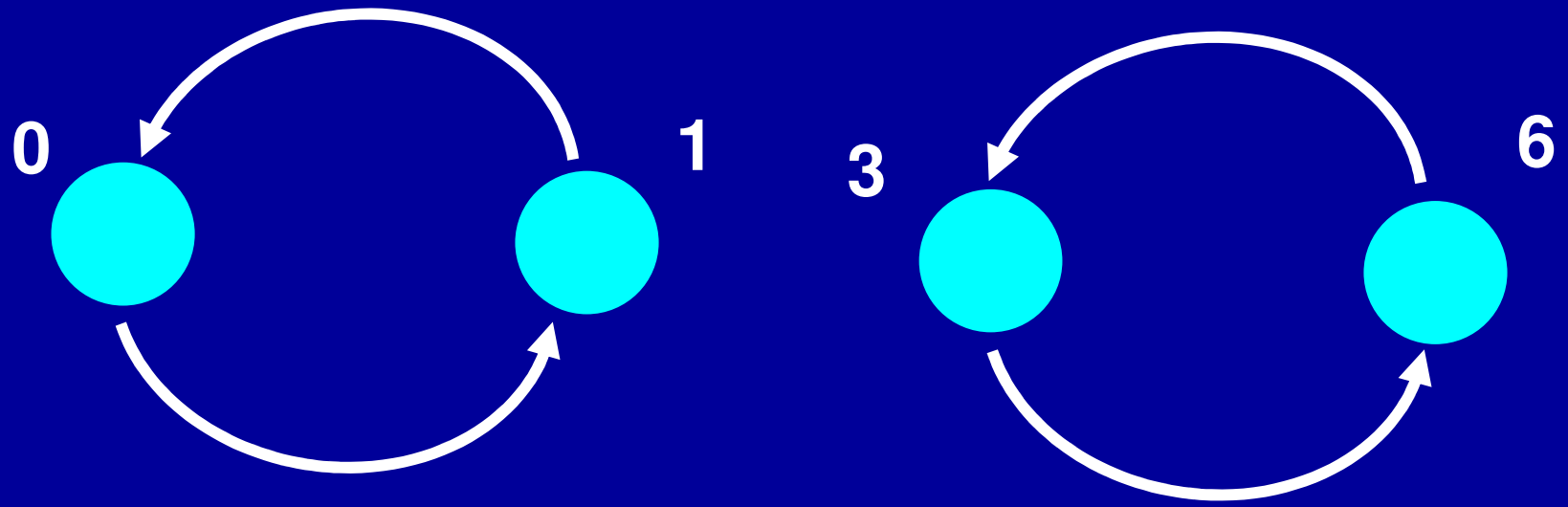
Riordina e rinumera i job secondo i b_j crescenti. Calcola la funzione φ^*.

job	0	1	2	3	4	5	6
b_j	1	3	15	19	26	31	40
a_j	7	4	16	45	22	34	18
$a_{\varphi^*(j)}$	4	7	16	18	22	34	45
$\varphi^*(j)$	1	0	2	6	4	5	3

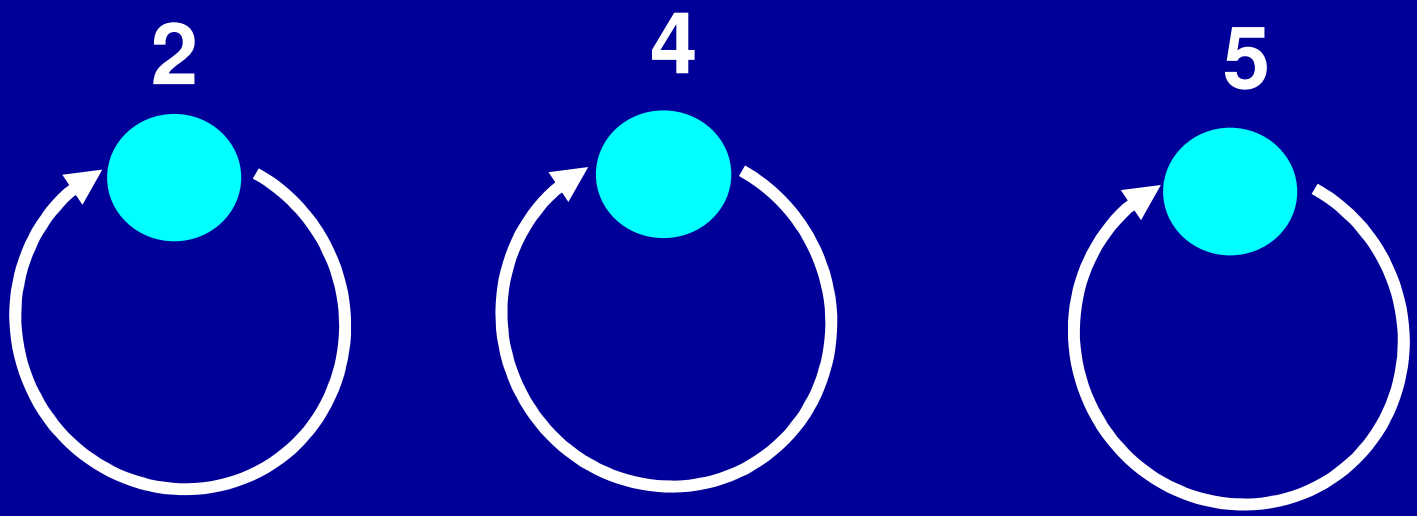
Passo 2

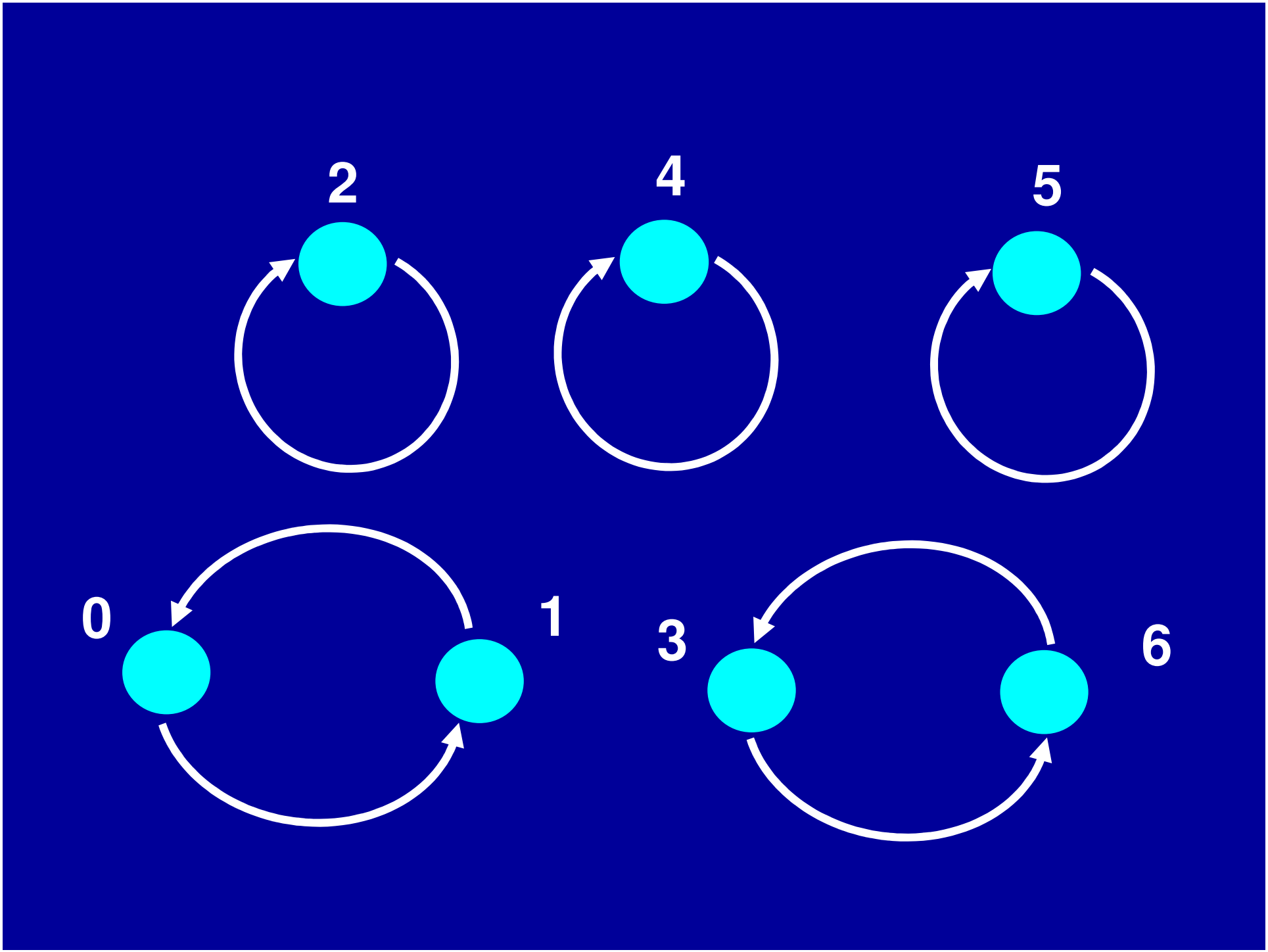
Dai valori di j, φ *(j) si deduce che:

i nodi 0 e 1 ed i nodi 3 e 6 sono connessi l'uno all'altro,



i nodi 2, 4, 5 sono indipendenti (ognuno, cioè, è connesso con sé stesso)




2
4
5
0
1
3
6

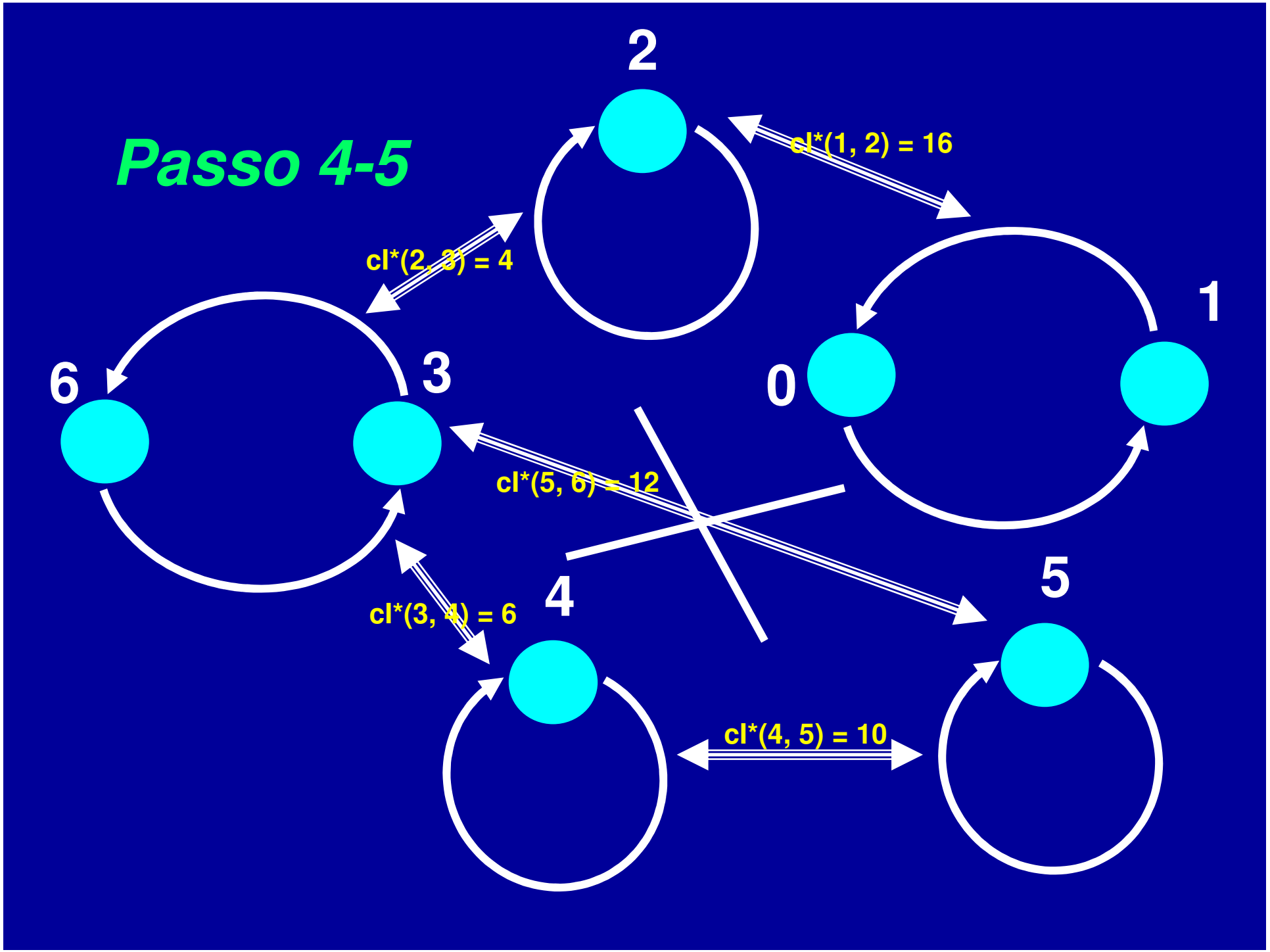
Passo 3

Calcolo dei costi di interscambio

$cl^*(j,j+1)$

job	0	1	2	3	4	5	6
b_j	1	3	15	19	26	31	40
$a_{\varphi^*(j)}$	4	7	16	18	22	34	45

J_0 e J_1 appartengono allo stesso ciclo
costo $l^*(1, 2) = 2(15 - 7) = 16$;
costo $l^*(2, 3) = 2(18 - 16) = 4$;
costo $l^*(3, 4) = 2(22 - 19) = 6$;
costo $l^*(4, 5) = 2(31 - 26) = 10$;
costo $l^*(5, 6) = 2(40 - 34) = 12$;


Passo 4-5
2
1
0
6
3
4
5
cl*(1, 2) = 16
cl*(2, 3) = 4
cl*(5, 6) = 12
cl*(3, 4) = 6
cl*(4, 5) = 10

Passo 6

Per sapere a quale gruppo appartengono gli scambi individuati, ogni bj deve essere confrontato con il corrispondente a $\varphi^*(j)$:

scambi	bj	$a_{\varphi^*(j)}$	gruppo
I (1, 2)	3	7	A
I (2, 3)	15	16	A
I (3, 4)	19	18	B
I (4, 5)	26	22	B

$j_1=2$, $j_2=1$, $j_3=3$, $j_4=4$.

Passo 7

Il ciclo ottimo è ottenuto dopo il seguente scambio:

φ^{**}=I(4,5) I(3,4) I(1,2) I(2,3) φ^{*}.

φ^{*}: 0,1 – 2 - 3,6 – 4 – 5

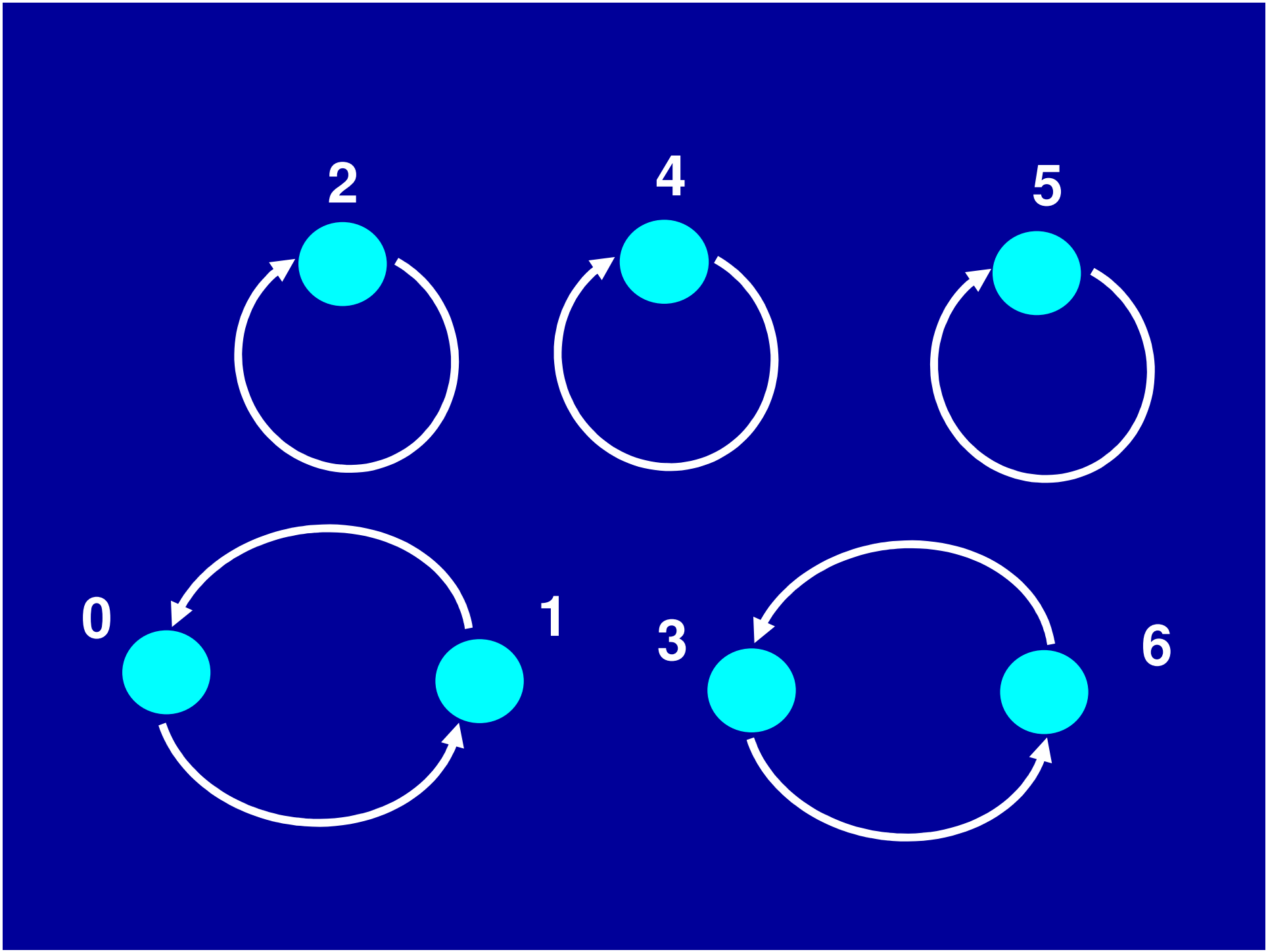
Si ha quindi che il ciclo ottimo è°:

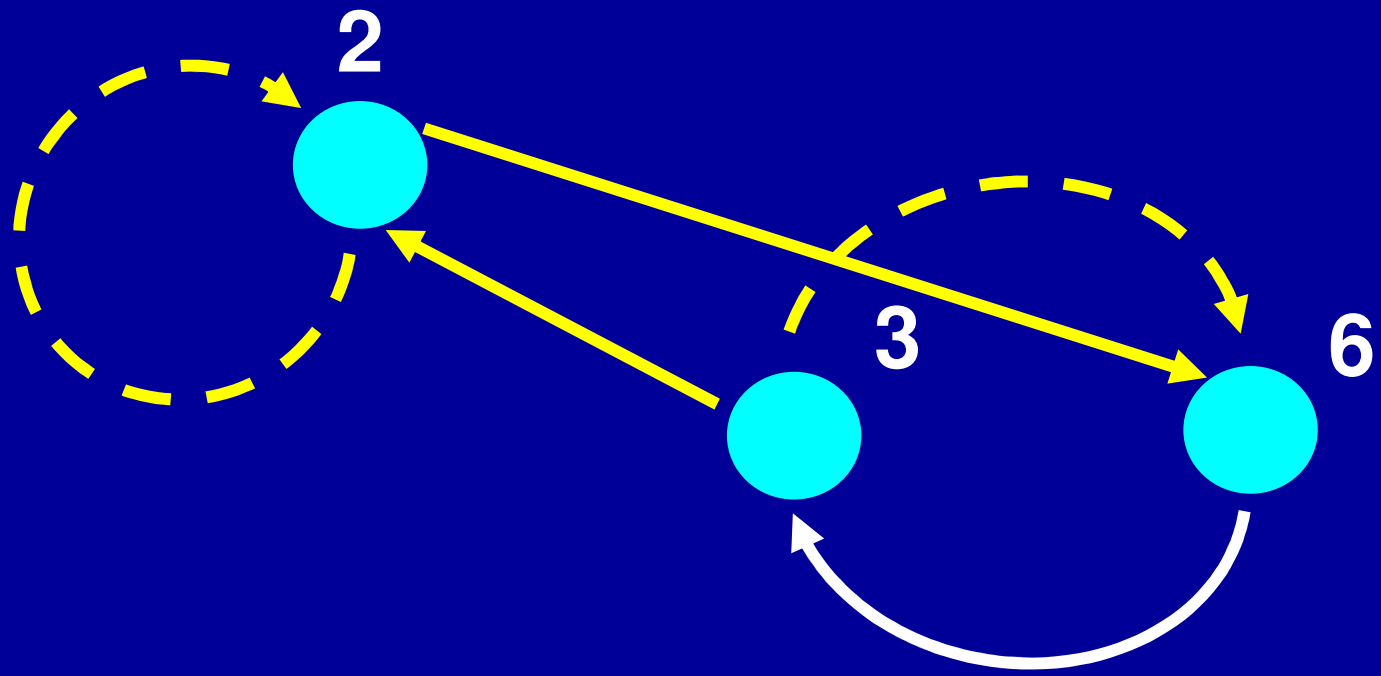
0 1 6 3 4 5 2 0

ed il costo totale

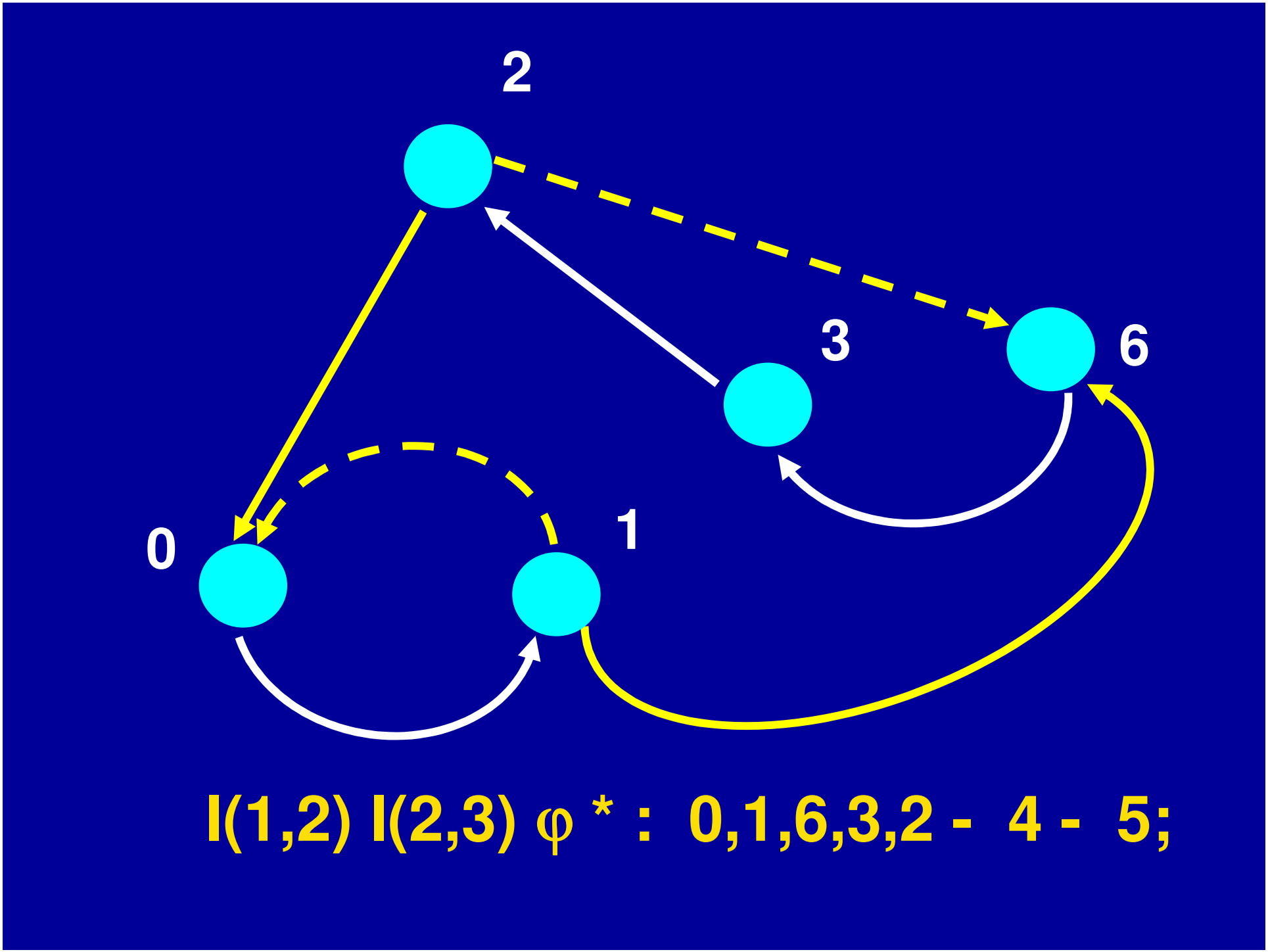
3 + 15 + 5 + 3 + 8 + 15 + 8 = 57.

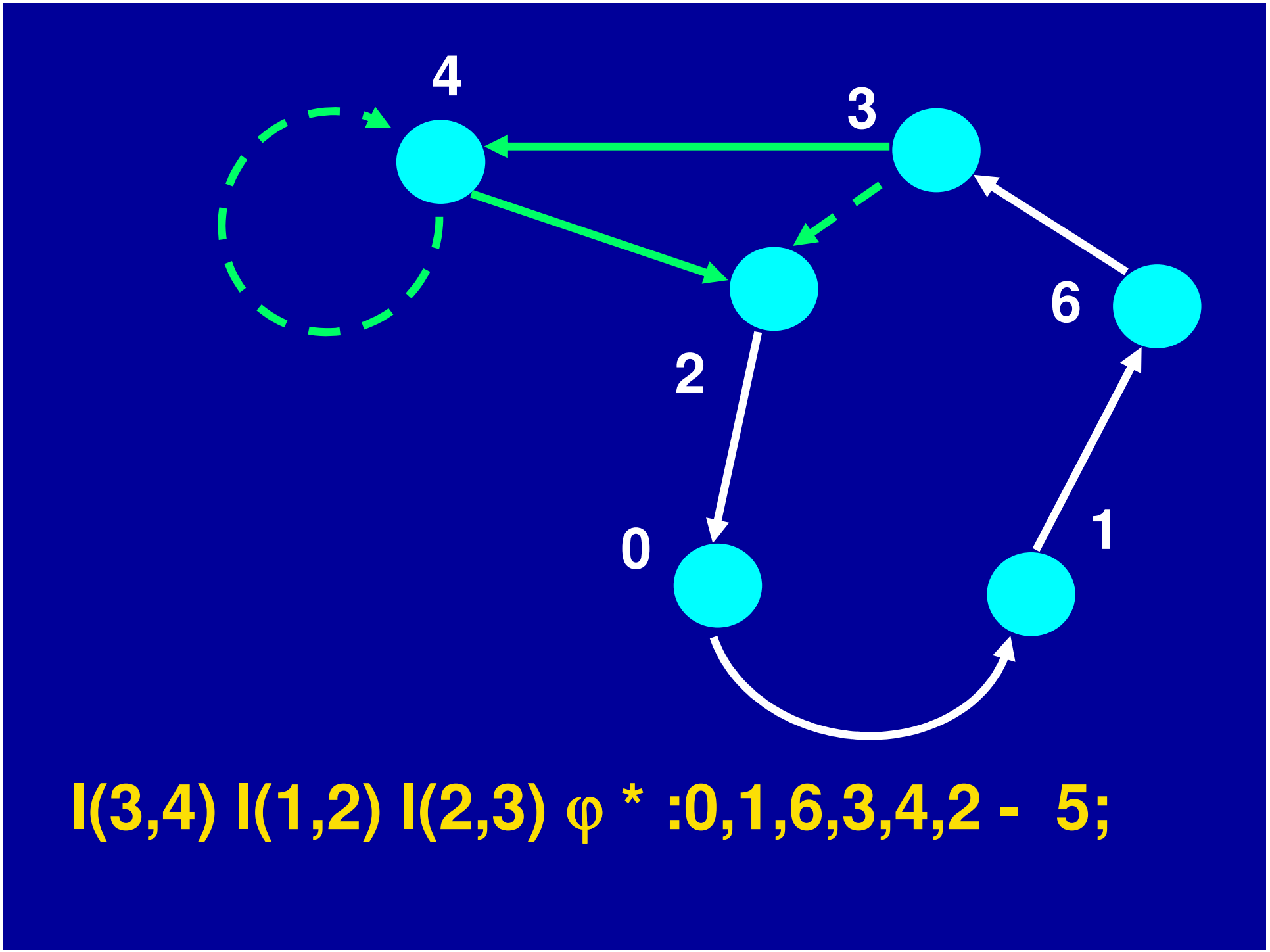
°dopo gli scambi di cui a seguire


2
4
5
0
1
3
6

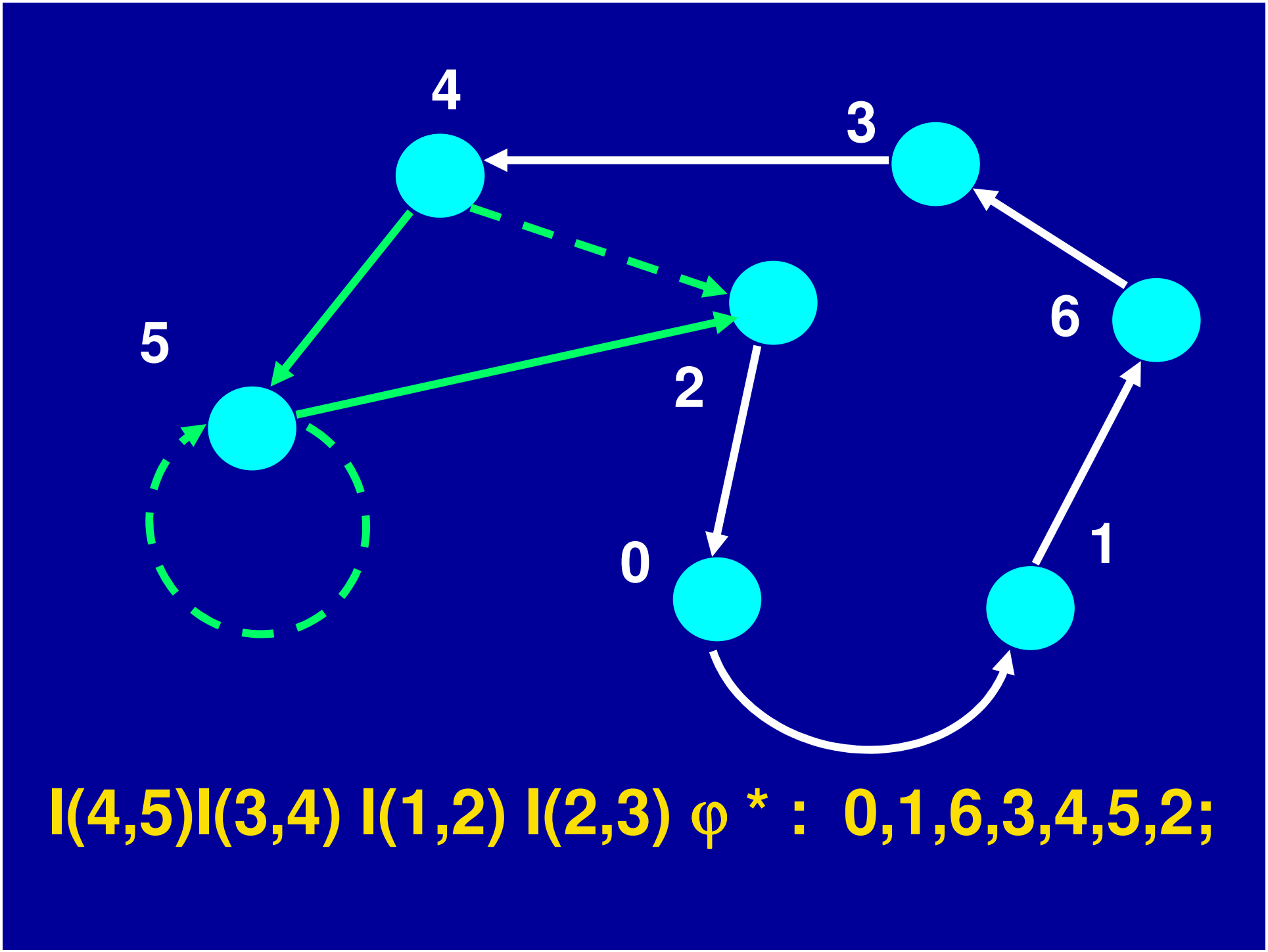


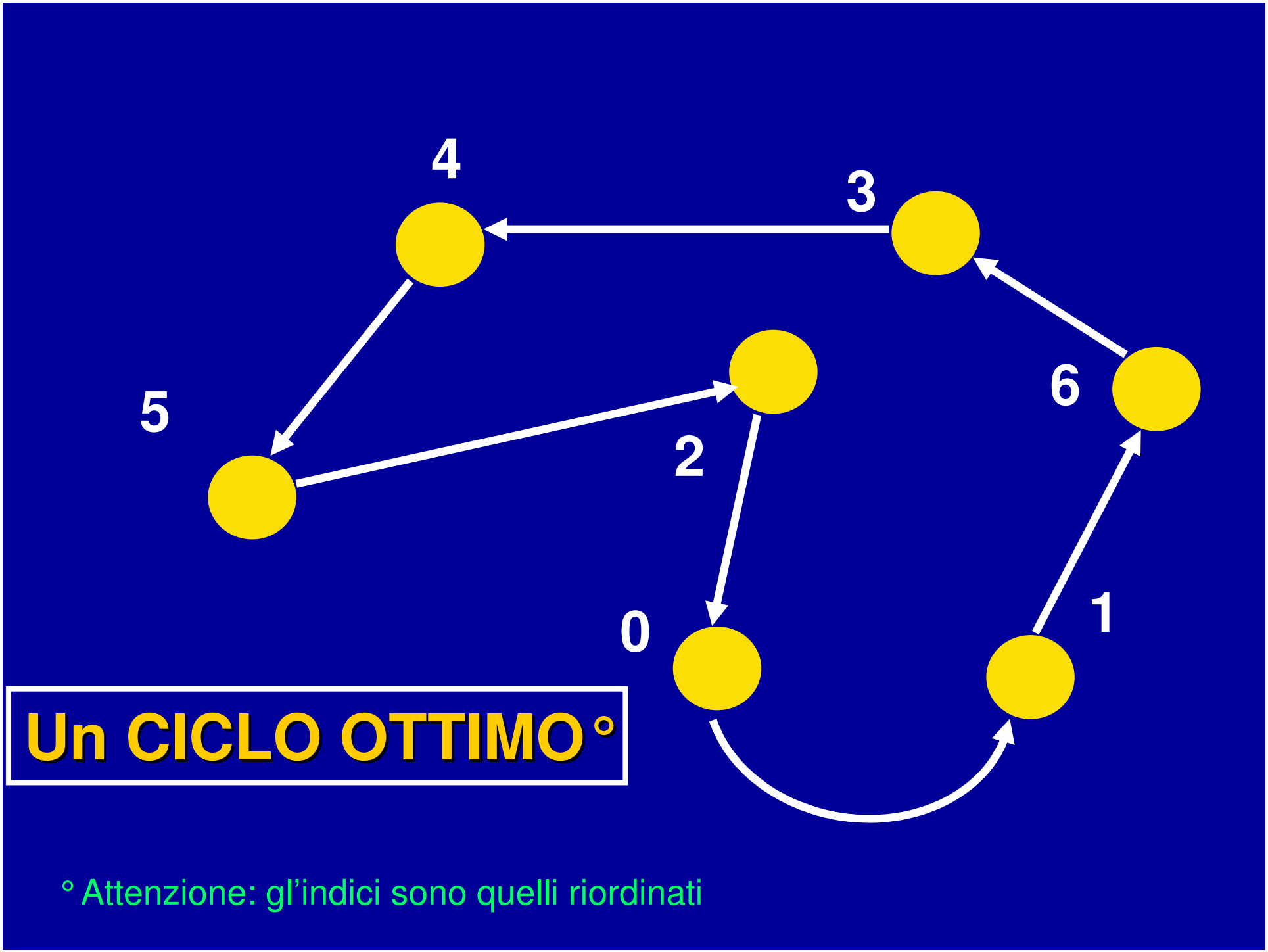
I(2,3) φ * : 0,1 - 3, 2,6 - 4 - 5;


2
3
6
0
1
I(1,2) I(2,3) φ * : 0,1,6,3,2 - 4 - 5;



I(3,4) I(1,2) I(2,3) φ * :0,1,6,3,4,2 - 5;

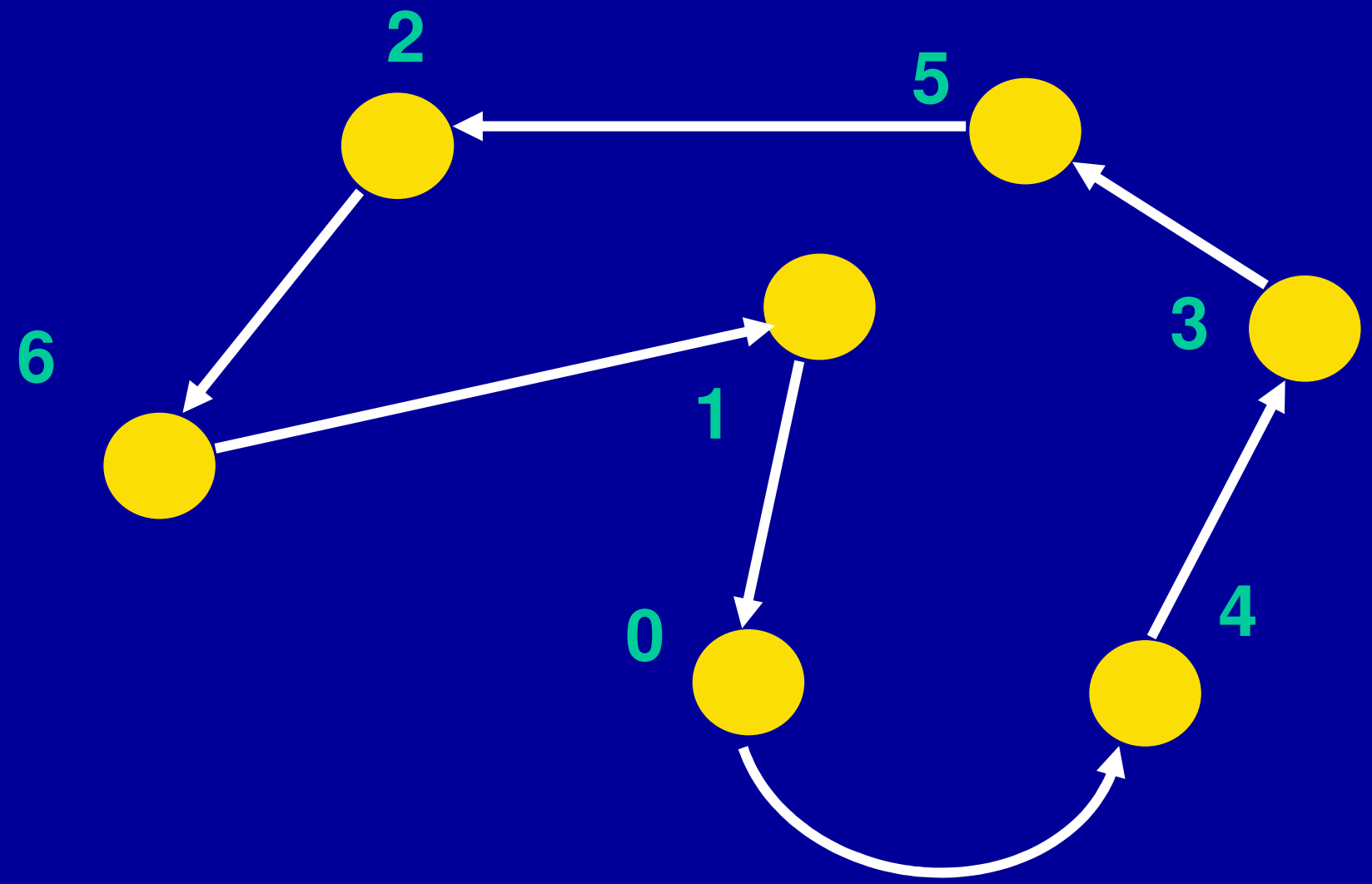

4
3
5
2
6
0
1
l(4,5)l(3,4) l(1,2) l(2,3) φ * : 0,1,6,3,4,5,2;



Un CICLO OTTIMO°

° Attenzione: gl'indici sono quelli riordinati

Cioè, con gl'indici originari:



CICLO OTTIMO° riconvertito

n lavori $\mathbf{J_i}$ di due operazioni in serie da eseguire su una linea senza buffer intermedio: il primo di lunghezza temporale a_i sulla macchina M_1; il secondo di lunghezza b_i sulla su M_2

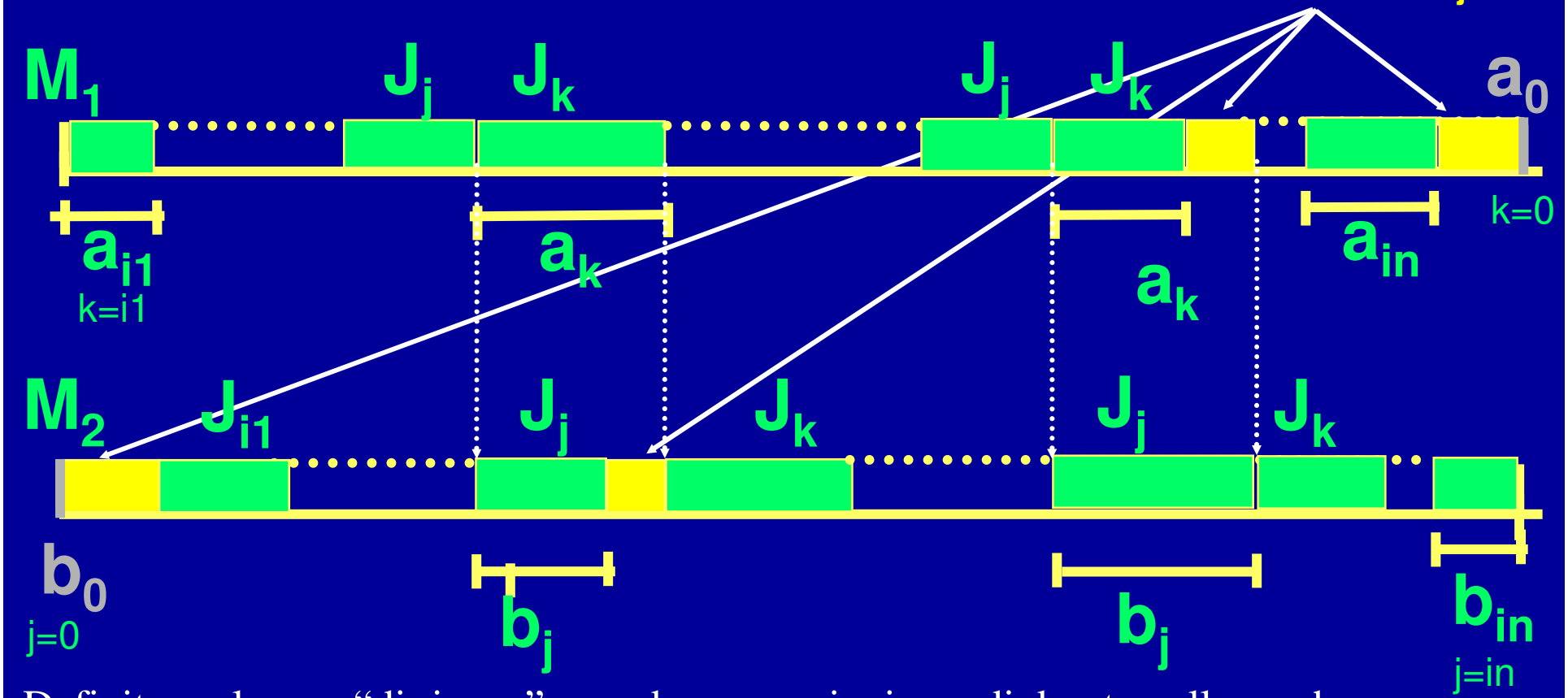
Per minimizzare il tempo di completamento C_m degli n lavori (tutti disponibili in ingresso: $r_{i1}=0$) si può applicare Gilmore e Gomery, attribuendo ad ogni lavoro $\mathbf{J_i}$ stato di inizio $\mathbf{a_i}$ **e** stato **di fine** $\mathbf{b_i}$

Sequenz. dei lavori su linea di due macchine senza buffer intermedio

$\min_S C_m$ → $\min_S \sum_{jk \in S} | b_j - a_k |$

J_i : due operazioni con tempi a_i su M_1 e poi b_i su M_2

attese: $|b_j - a_k|$



Definito un lavoro "di riposo", con due operazioni, ma di durata nulla, anche se strettamente ordinate fra loro come negli altri lavori, il minimo si ottiene minimizzando la somma dei valori assoluti delle differenze tra b_j e a_k, dove j precede k, nella permutazione degl'indici da 0 a n, relativa alla sequenza S