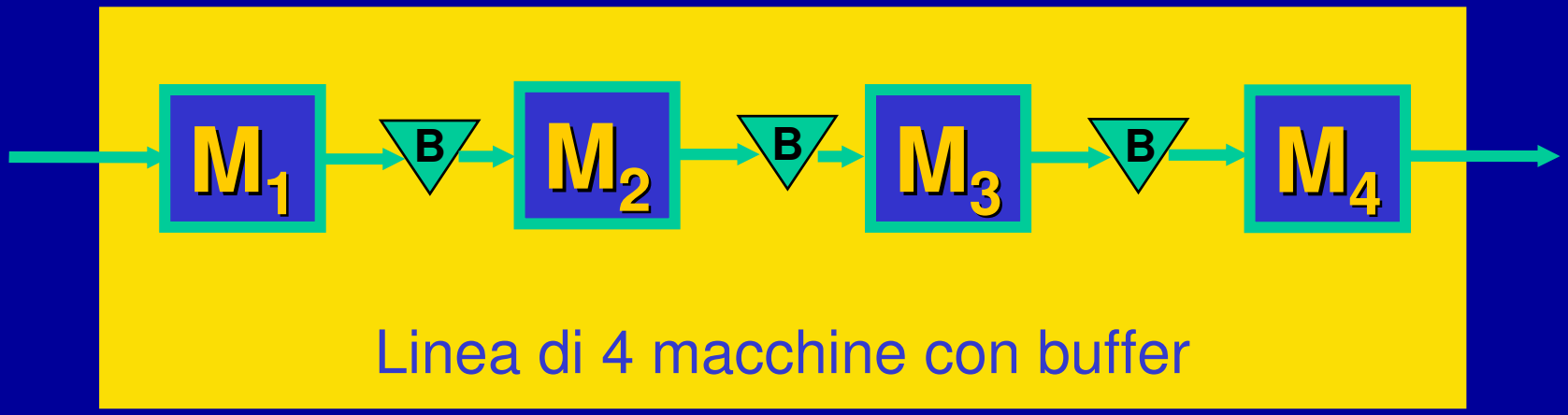
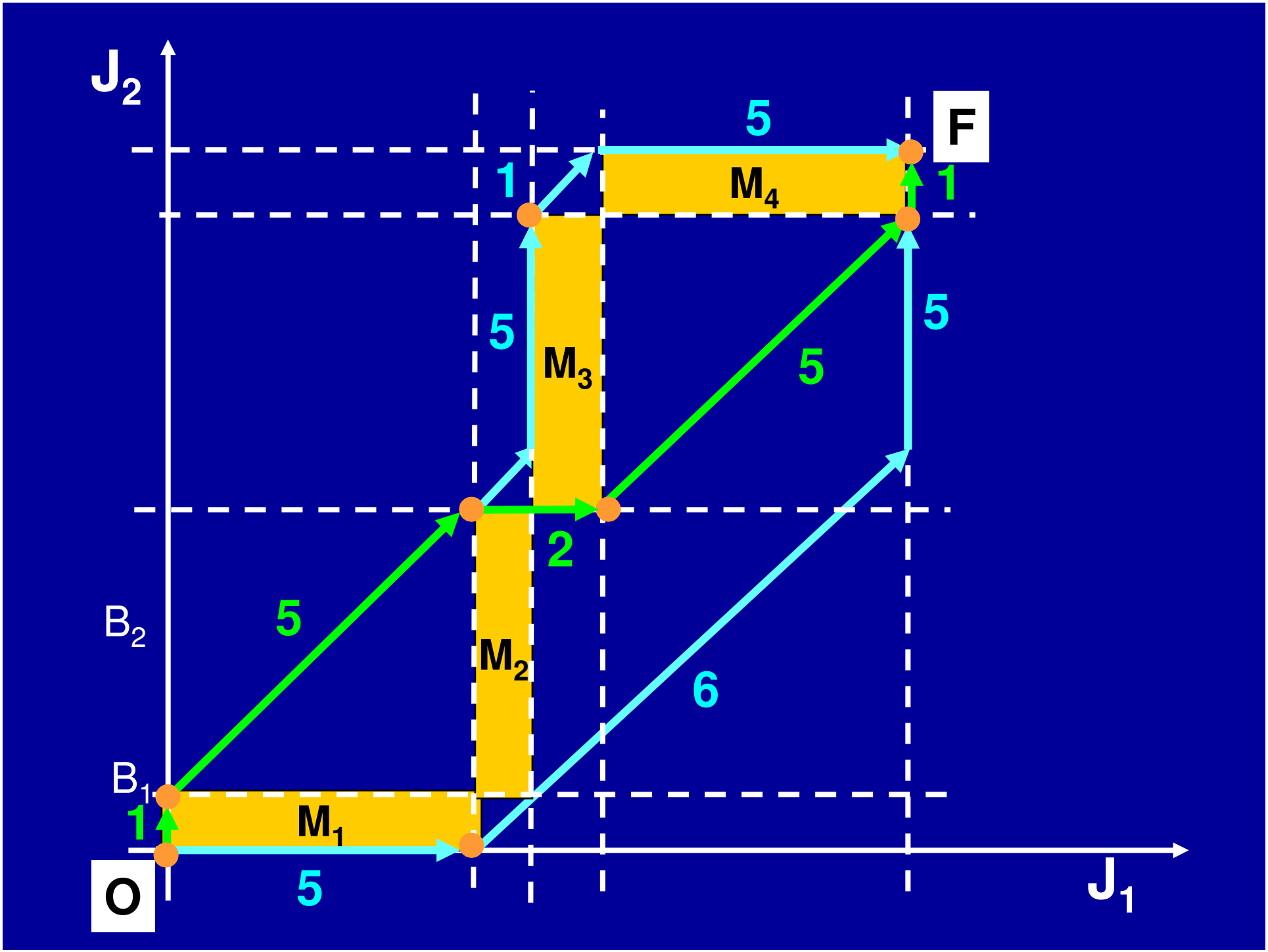
La soluzione del 2JSP è la generalizzazione di quella data da Aker per 2 lavori $\mathbf{J}_i$ di m operazioni in serie da eseguire su una linea con buffer intermedi: ciascuno di lunghezza temporale t_i sulla macchina M_i (i=1,…, m)

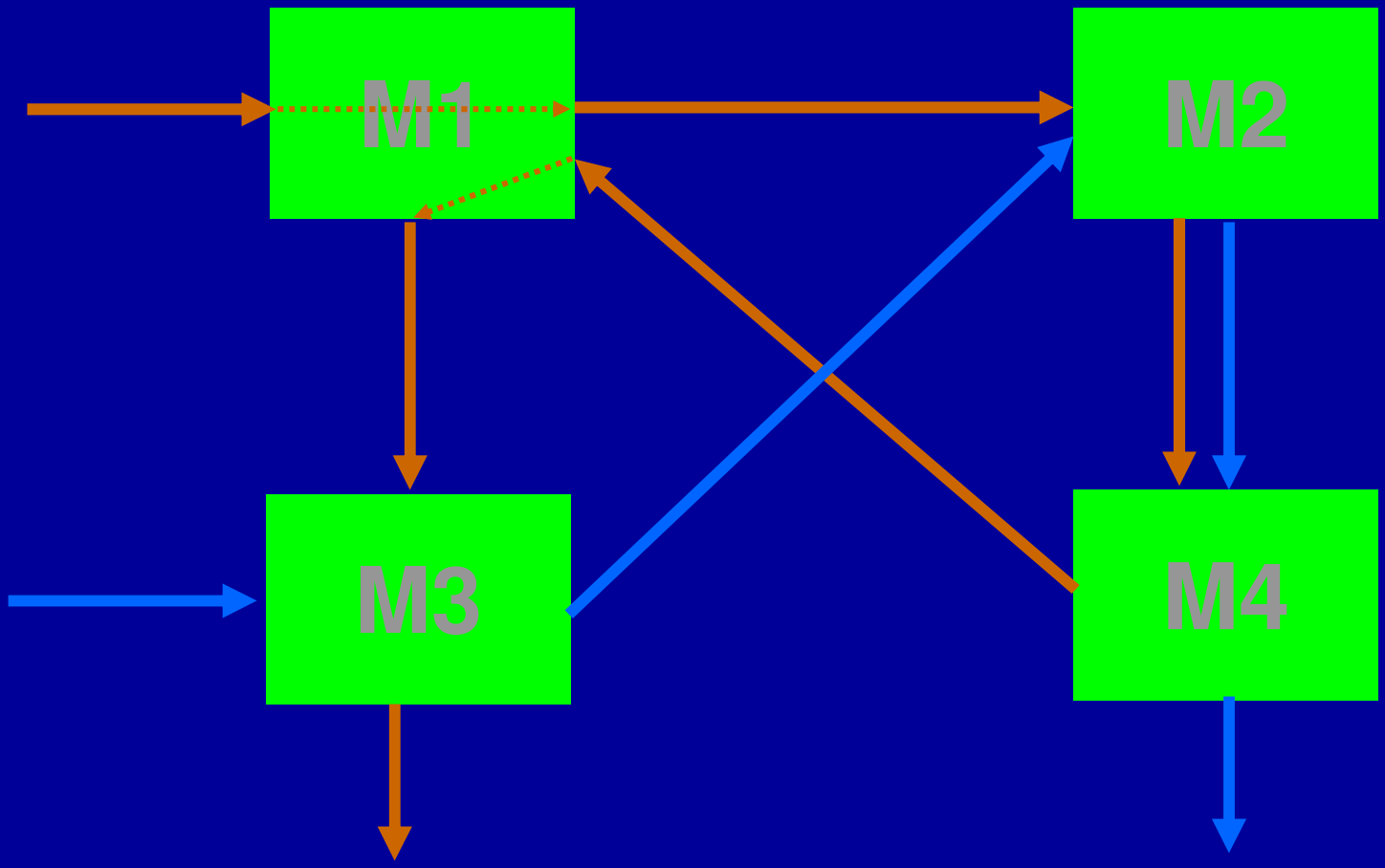


Linea di 4 macchine con buffer

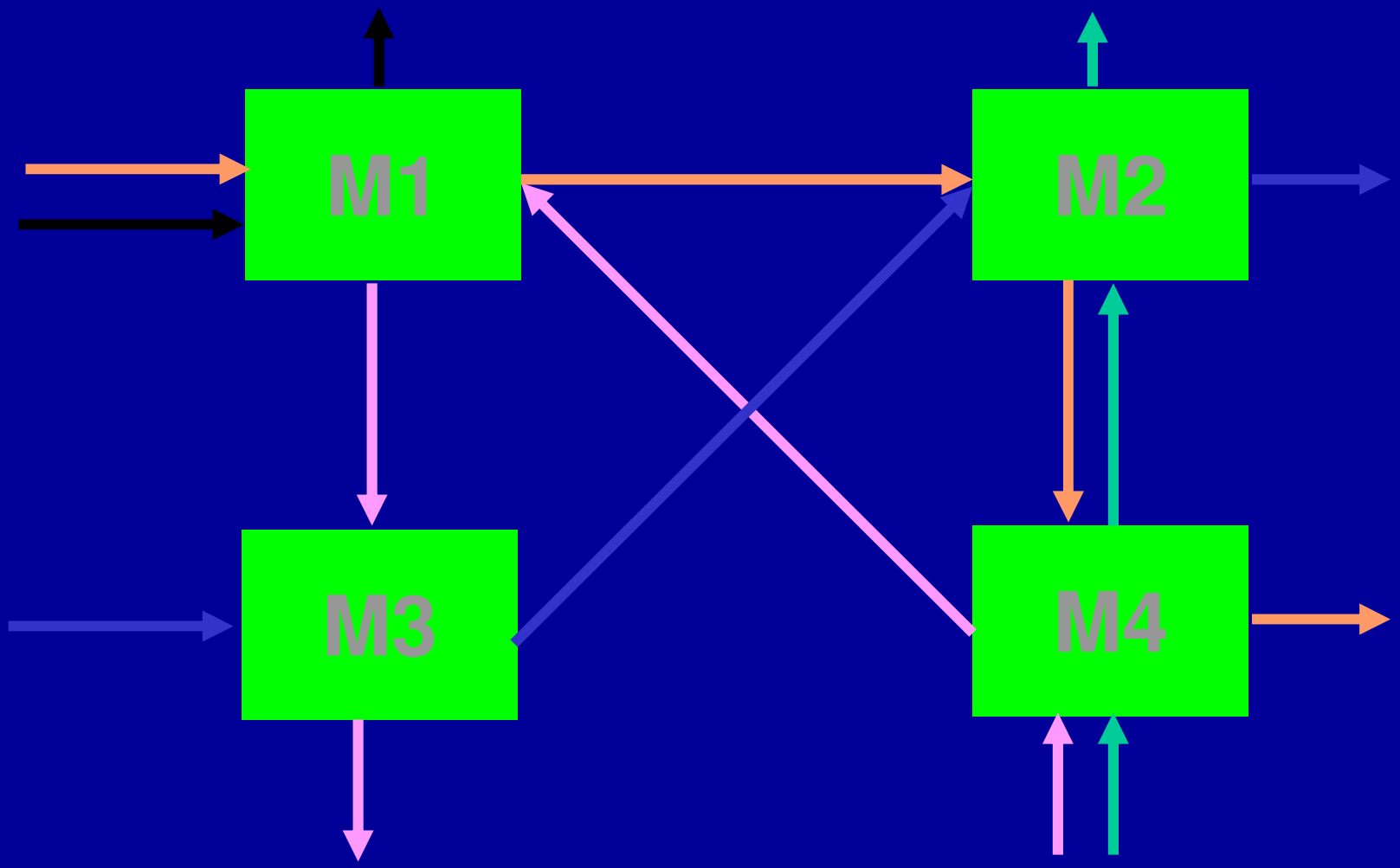
Per minimizzare il tempo di completamento C_m si usa una griglia di condivisione in cui le zone proibite riguardano le macchine


J2
J1
O
F
B2
B1
M1
M2
M3
M4
5
1
5
1
5
5
2
5
6
5

Min C_m si può risolvere come un 2JSP allo stesso modo



Min C_m non si può risolvere allo stesso modo quando i lavori sono n>2 (già per n=3 le zone proibite sarebbero cubi e la progressione greedy potrebbe incidere su una faccia ….)



Flow shop

In una linea esiste un vincolo tecnologico detto “flow shop”:

tutti i lavori devono visitare le macchine nello stesso ordine

Il Flow shop è un caso particolare di Job Shop

Quando non altrimenti specificato, si intende che i lavori possono aspettare tra un’operazione e la seguente

(ATTENZIONE: grazie ai buffer, le macchine possono operarli in ordine diverso)

Permutation schedule

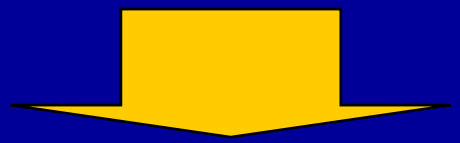
Quando in un flow shop si ha uno schedule nel quale le macchine processano i lavori nello stesso ordine (cioè quando: se la macchina M_1 processa J_i prima J_k, ciò è vero anche per le altre macchine), allora si ha un "Permutation schedule"

ATTENZIONE. Senza buffer si ha sempre un Permutation schedule

INDICI REGOLARI

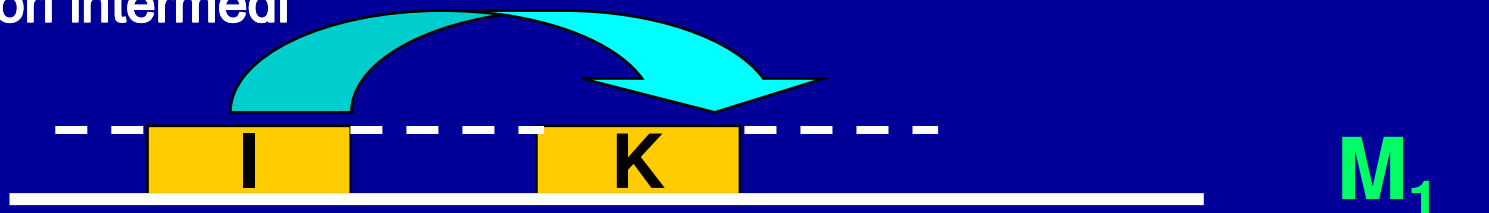
C_m è un particolare, importante, "indice regolare" cioè una funzione non decrescente dei tempi di completamento dei lavori:

$f(C_1 \ldots C_n)$



Quando l'indice di comportamento è regolare, in un Flow Shop, senza perdita di generalità, ci si può limitare a considerare un "Permutation flow shop" sulle prime due macchine

I si inserisce, sulla prima macchina, dopo K, arretrando quest'ultimo e i lavori intermedi



Permutazione differenziata

Permutazione uguale

Abbiamo ottenuto un permutation schedule sulle prime due macchine senza influenzare i lavori su M_2 e quindi i C_i

Indice di comportamento C_m

Quando l'indice è C_m, in un Flow Shop ci si può limitare a considerare un "Permutation flow shop" sulle ultime due macchine , senza perdita di generalità

Permutazione differenziata

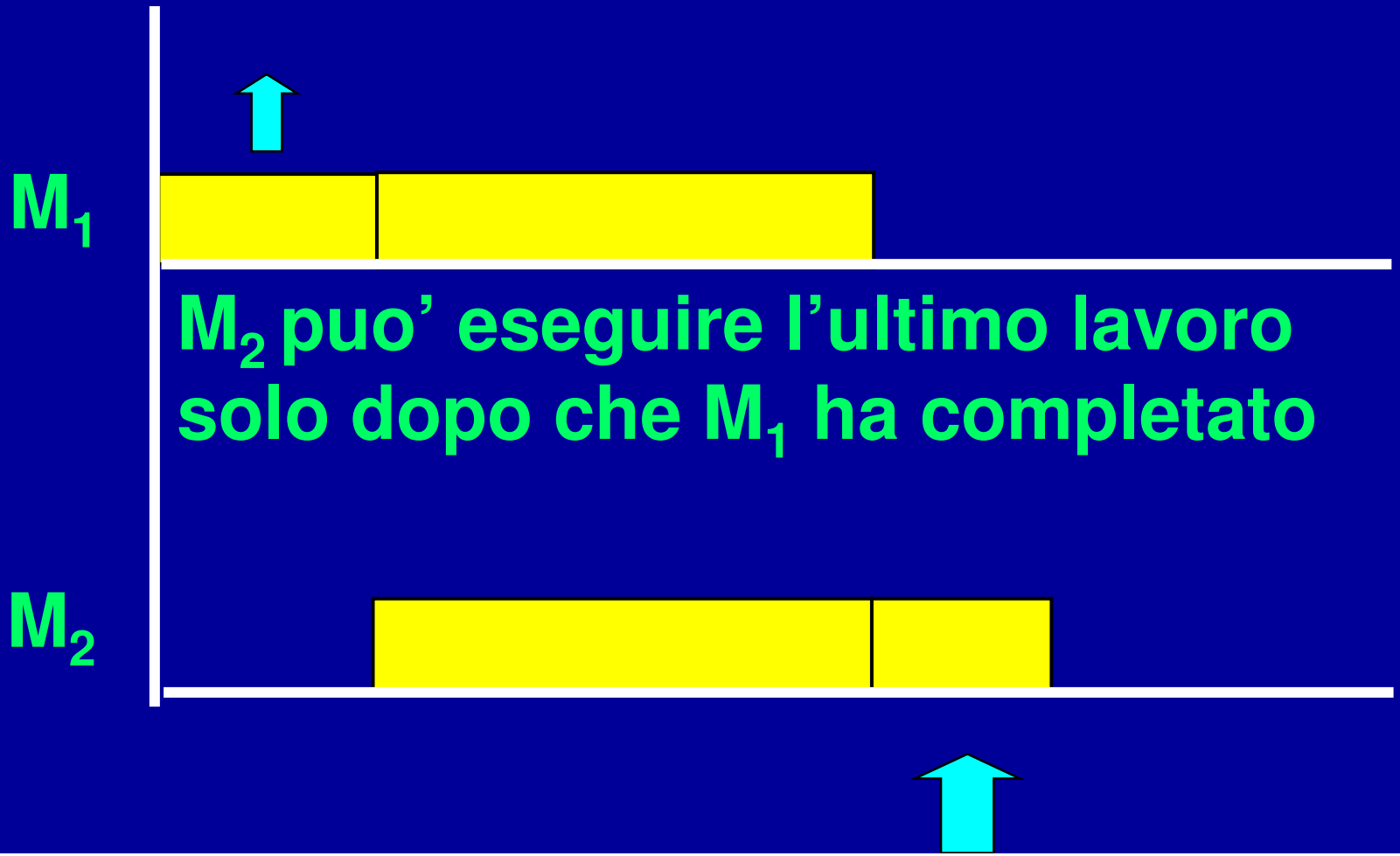
Abbiamo ottenuto un permutation schedule senza influenzare il completamento dei lavori su M_m, cioè C_{max}

PROBLEMA

- n LAVORI
- 2 MACCHINE
- Min { F_{max} (= C_m)}

Si usa indicare questo problema con (n /2/F/F_{max}): il primo campo indica il numero dei lavori, il secondo il numero di risorse, il terzo la corrispondenza tra operazioni e risorse utilizzate, il terzo l'indice di comportamento

M_2 deve aspettare che il primo lavoro su M_1 sia terminato



M_2 puo' eseguire l'ultimo lavoro solo dopo che M_1 ha completato

ALGORITMO DI JOHNSON

$a_i = p_{i1}$ tempo di processamento di J_i su M_1

$b_i = p_{i2}$ tempo di processamento di J_i su M_2

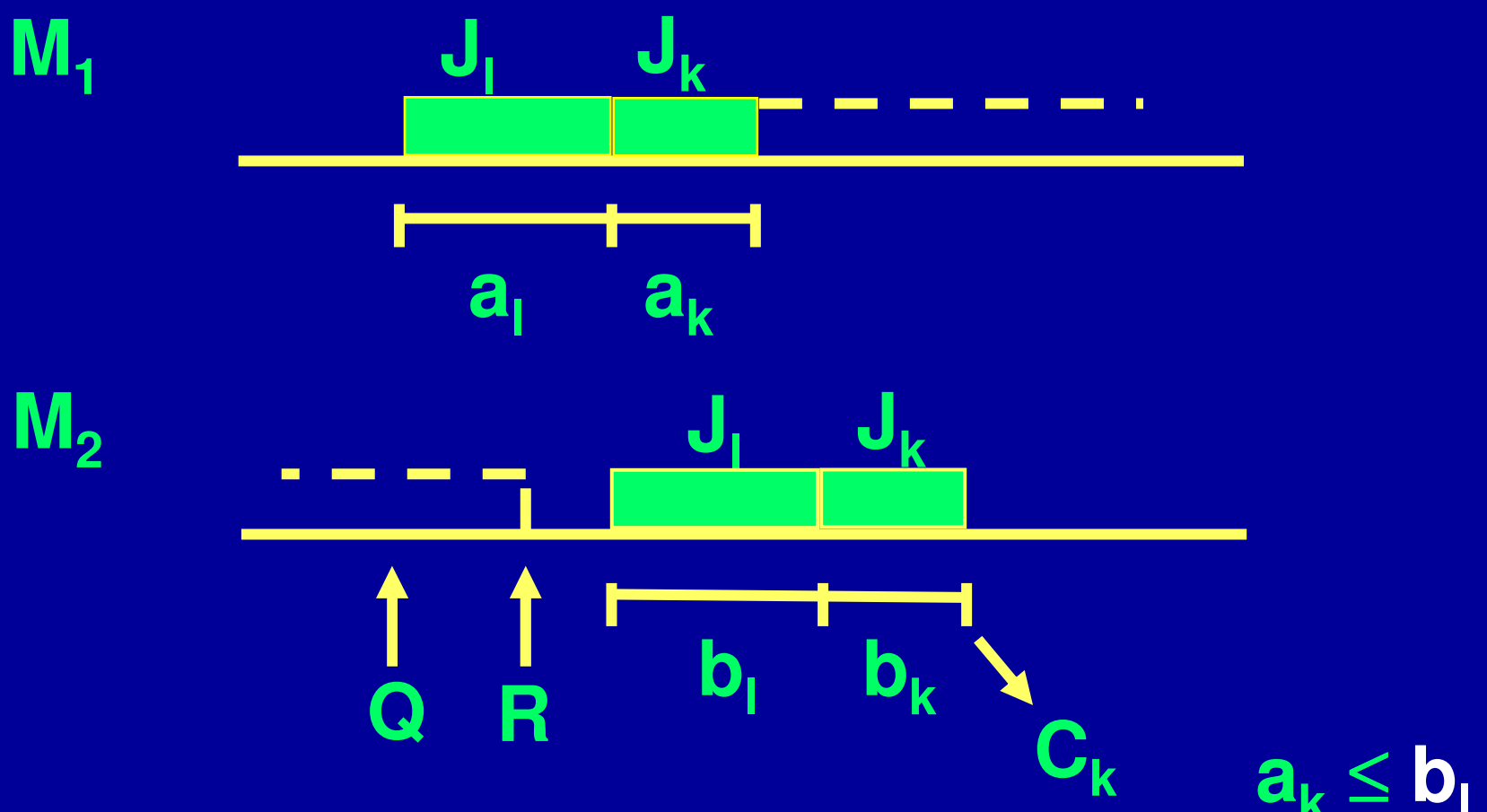
ALGORITMO DI JOHNSON

Se

1) $a_k = \min\{ a_1,..,a_n,b_1,...b_n\}$
   $\exists$ uno schedule ottimo nel quale
   J_k è processato per primo.

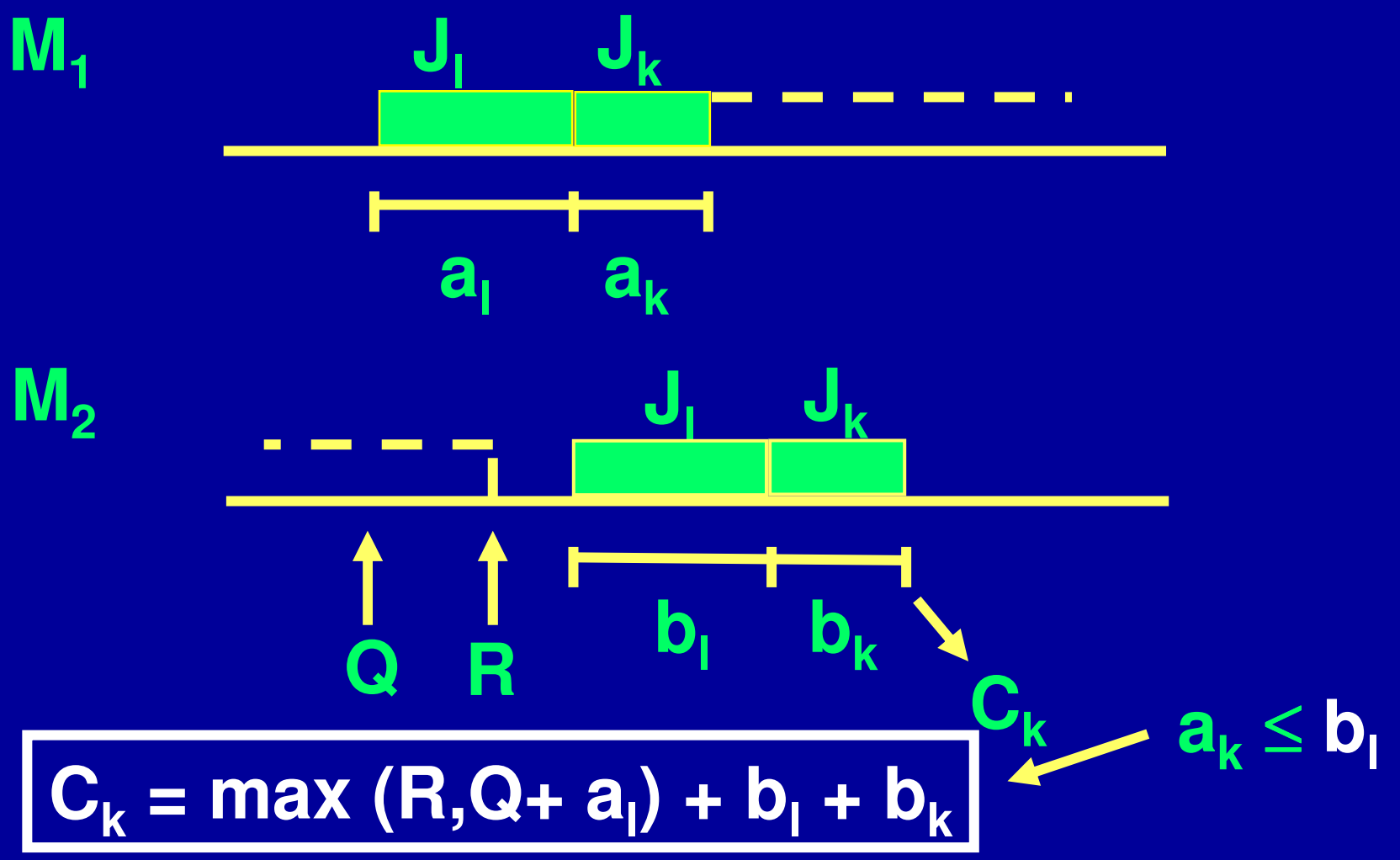
2) $b_u = \min\{ a_1,..,a_n,b_1,...b_n\}$
   $\exists$ uno schedule ottimo nel quale
   J_u è processato per ultimo.

Sequenziamento S



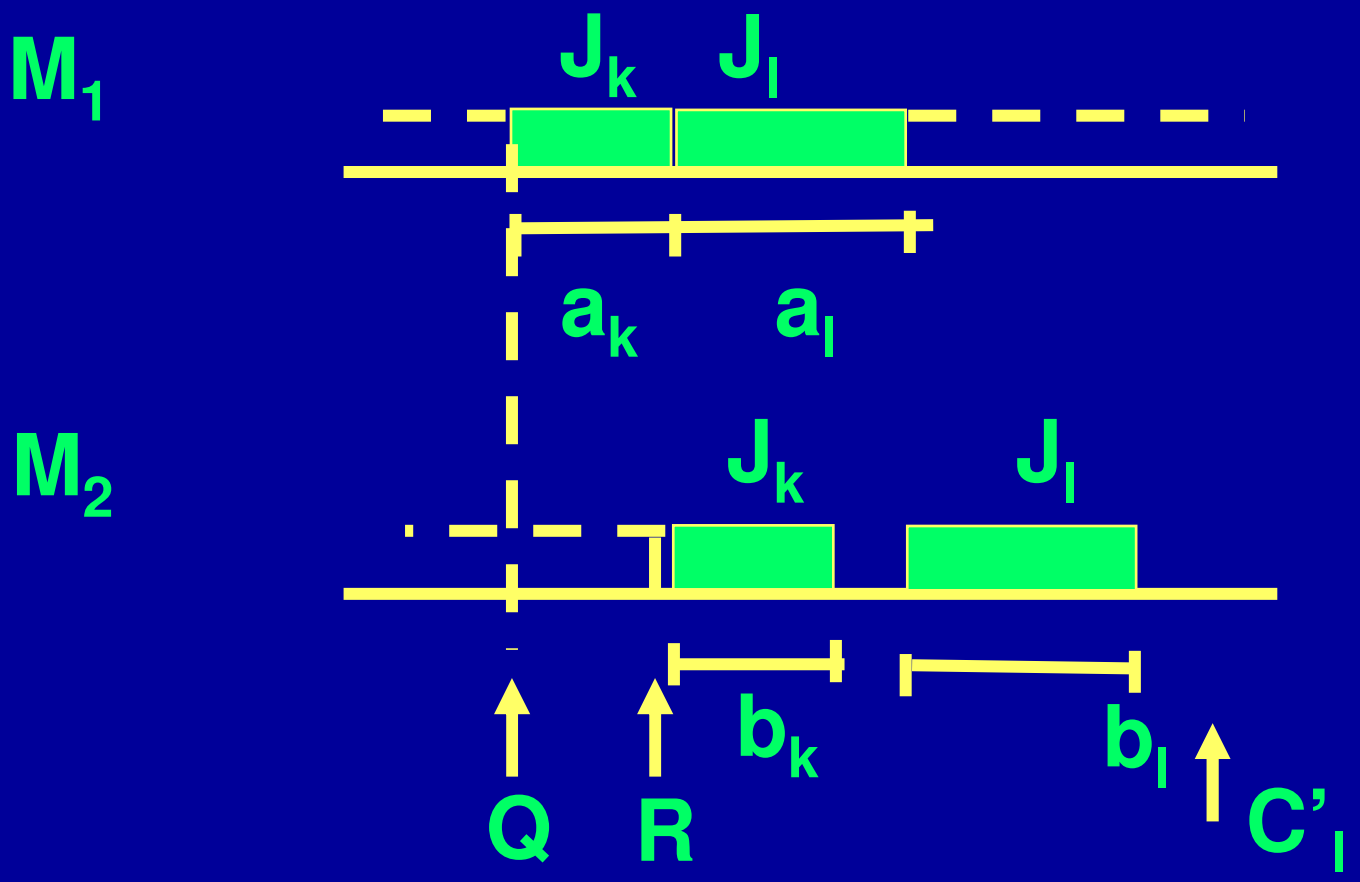
$a_k \leq b_l$

Sequenziamento S

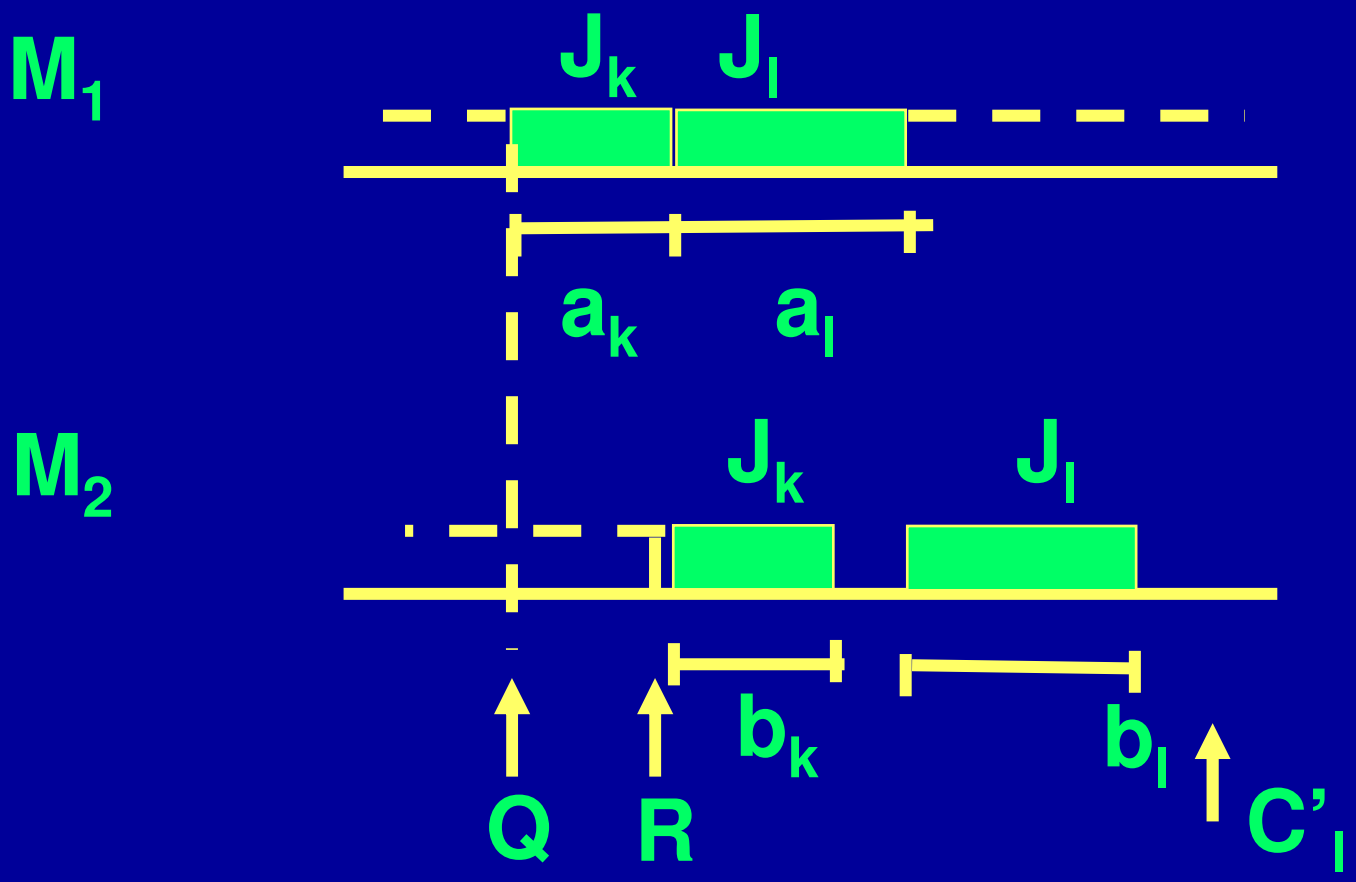


$$C_k = \max (R, Q + a_l) + b_l + b_k$$

Sequenziamento S’



Sequenziamento S’



$$C'_l = b_l + \max\{\max(R, Q + a_k) + b_k,\ Q + a_k + a_l\}$$

TESI

$a_k = \min\{ a_1,..,a_n,b_1,...b_n\}$

$\exists$ uno schedule (S') ottimo nel quale J_k è processato per primo.

$\Rightarrow \quad C'_I \leq C_k$

$$c_k = \max(R, Q + a_l) + b_l + b_k$$

$$c_k \geq \max\{R, Q + a_k\} + b_l + b_k$$

$a_k \leq a_l$

$$c_k \geq Q + a_l + b_k + b_l$$

$$\geq Q + a_l + a_k + b_l$$

$a_k \leq b_k$

$$c_k \geq c'_l$$

$$C'_l = \max\{\max(R, Q + a_k) + b_k,\ Q + a_k + a_l\} + b_l$$

ALGORITMO

Si è dimostrato che si può anticipare J_k rispetto al J_l che lo precede in una schedule ottima senza alcuna ipotesi sulla schedule dei lavori sequenziati prima di J_l. Ne consegue che, assegnato J_k al primo posto, lo stesso ragionamento si applica agli n-1 lavori ancora da schedulare. Da cui l'algoritmo che segue

Algoritmo di Johnson (1954)

Passo 1 $K = 1,\ l = n$

Passo 2 $NS := \{ J_1, \ldots J_n \}$

Passo 3 Se $a_j = \min_{i : J_i \in NS} (\min (a_i , b_i))$

$i(k) = j$

$NS := NS / \{J_j\}$ $k := k+1$

Algoritmo di Johnson (1954)

Passo 4 Se $b_j = \min_{i : J_i \in NS} (\min (a_i , b_i))$

$i(l) = j$

$NS := NS / \{J_j\}$ $l := l-1$

Passo 5 Se $NS \neq \{\varnothing\}$ → Passo 3

Altrimenti la sequenza ottima è $\{J_{i(.)}\}$

Esercizio su Johnson

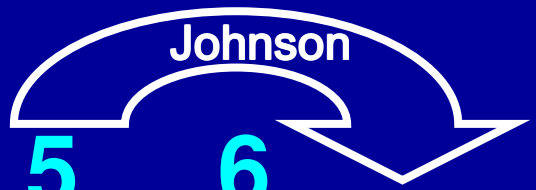
I tempi sulle macchine sono:

Lavori:	1	2	3	4	5	6
su M_1	1	2	3	4	5	7
su M_2	6	6	6	5	3	5

Per vedere quanto si guadagna con il buffer, si minimizzi rispetto alla sequenza C_m quando non c'è buffer intermedio (con Gilmore e Gomery)

Esercizio su Johnson

I tempi sulle macchine sono:



Lavori:	1	2	3	4	5	6
su M_1	1	2	3	4	5	7
su M_2	6	6	6	5	3	5

La sequenza ottima, di Johnson, è: 1-2-3-4-6-5

Dal Gantt si vede che:

$C_m = a_{i(1)} + \Sigma_{i=1,..n} b_i = 32$ cioè, dopo la prima, non ci sono attese in M_2, così come non ci sono in M_1 per la presenza del buffer intermedio

Se inoltre, come accade con questa sequenza di Johnson, $a_{i(k)} \leq b_{i(k-1)}$ per k=2,…,n, allora eliminando il buffer intermedio C_m non aumenta, anche se si possono creare attese di M_1

Sequenziamento generale* su due macchine

$(n /2/G^*/F_{max})$

Tipo A	Lavori solo su M_1
Tipo B	Lavori solo su M_2
Tipo C	prima su M_1 poi su M_2
Tipo D	prima su M_2 poi su M_1

*senza rivisitazione della stessa macchina

Sequenziamento generale su due macchine

$(n /2/G^*/ F_{max})$

S_A qualsiasi sequenza dei lavori di tipo A

S_B qualsiasi sequenza dei lavori di tipo B

S_C sequenza di Johnson di quelli di tipo C

S_D sequenza di Johnson di quelli di tipo D

Le sequenze ottime sono allora:

su M_1 S_c S_A S_d

su M_2 S_d S_B S_C

Sequenziamento con tre macchine

Sequenziamento per flusso su tre macchine

Con tempi di processamento qualsiasi è un problema difficile. Se

$$\min_{i=1,\ldots,n} p_{i1} \geq \max_{i=1,\ldots,n} p_{i2}$$

il problema è facile.

$a_i = p_{i1} + p_{i2}$

$b_i = p_{i2} + p_{i3}$

Algoritmo di Johnson

Sequenza ottima

si devono considerare solo le permutazioni (stesso sequenziamento sulle tre macchine)

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** $\mathbf{b_i}$

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

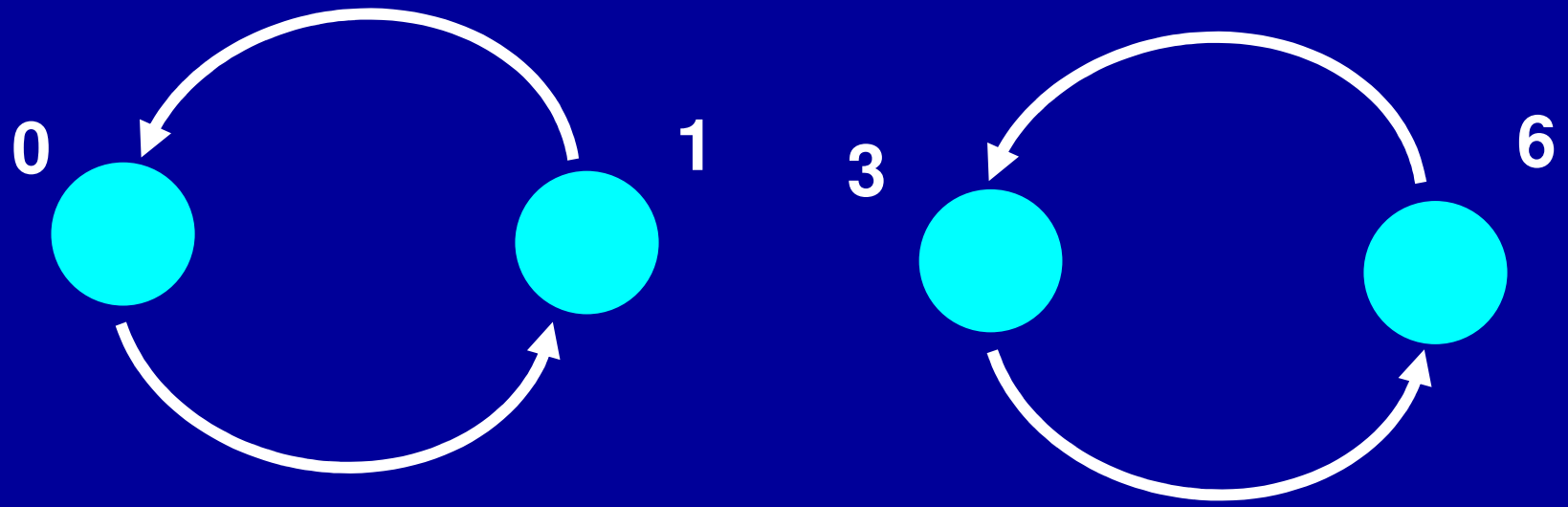
Passo 1

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

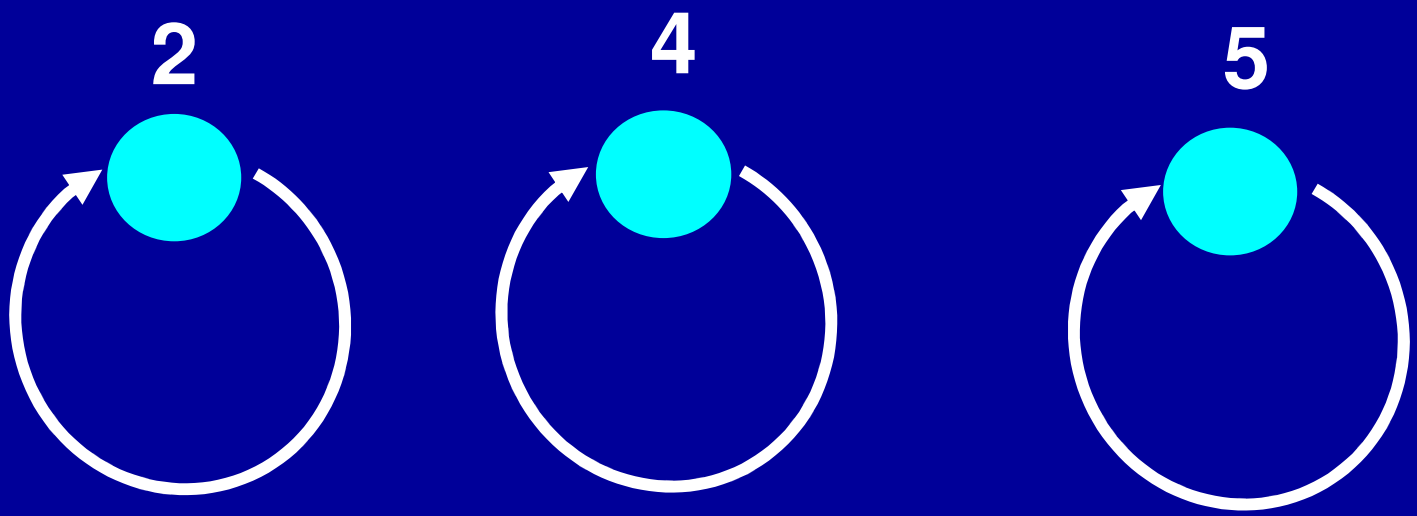
Passo 2

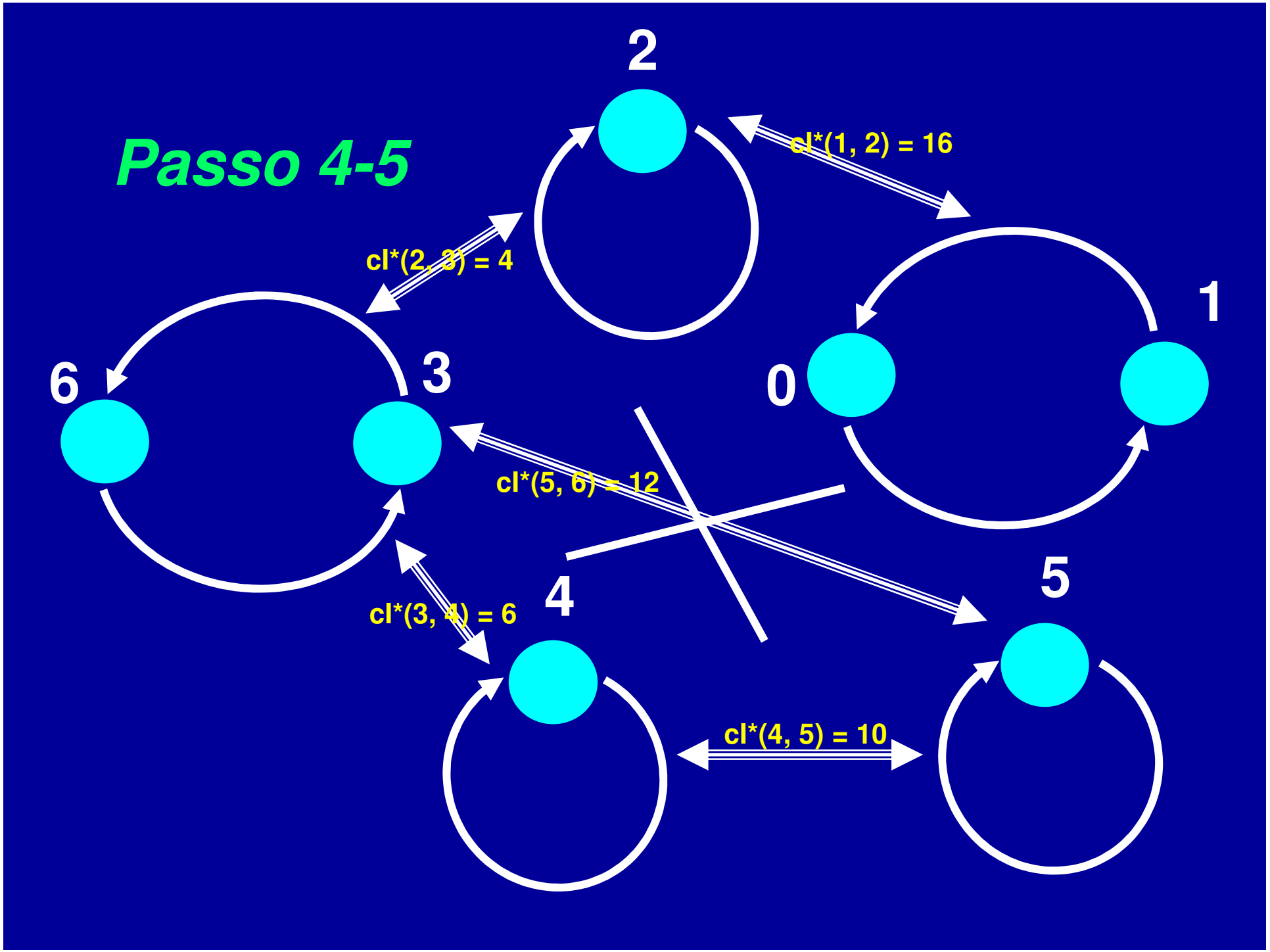
Dai valori di j, φ *(j) si deduce che:

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)




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

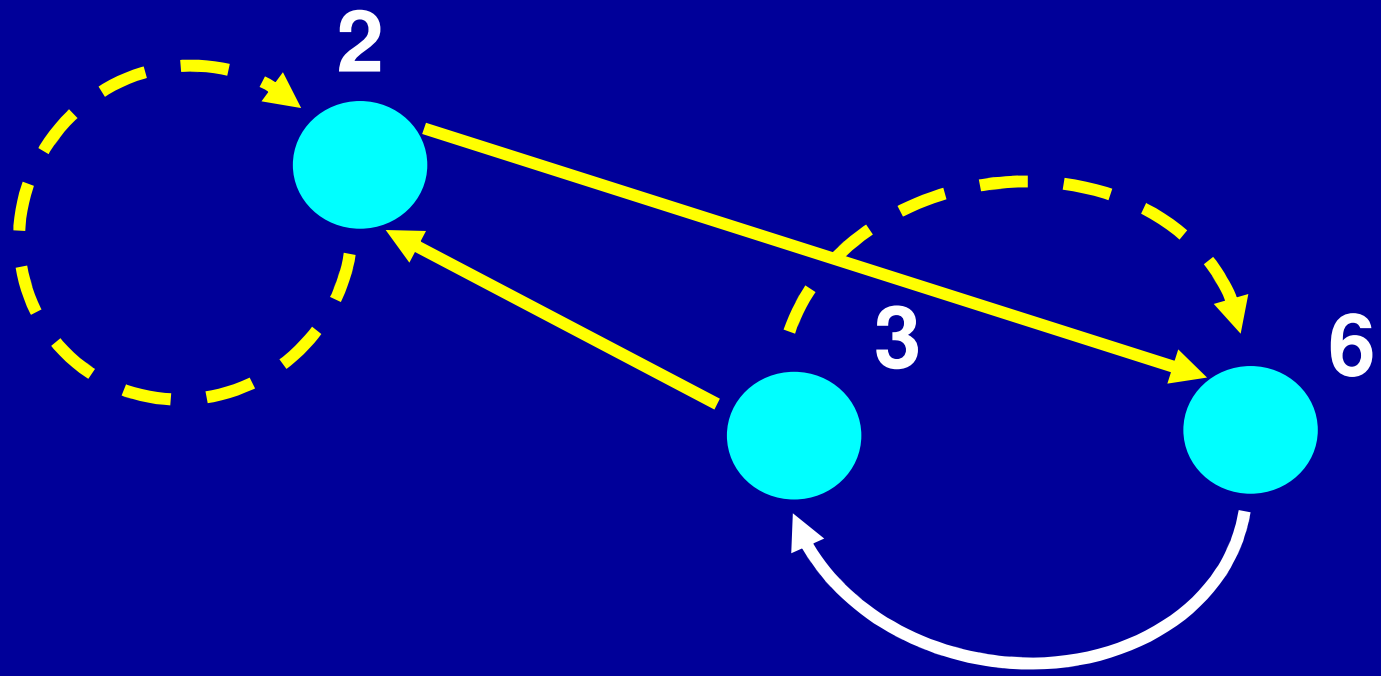
Si ha quindi che il ciclo ottimo è°:

0 1 6 3 4 5 2 0

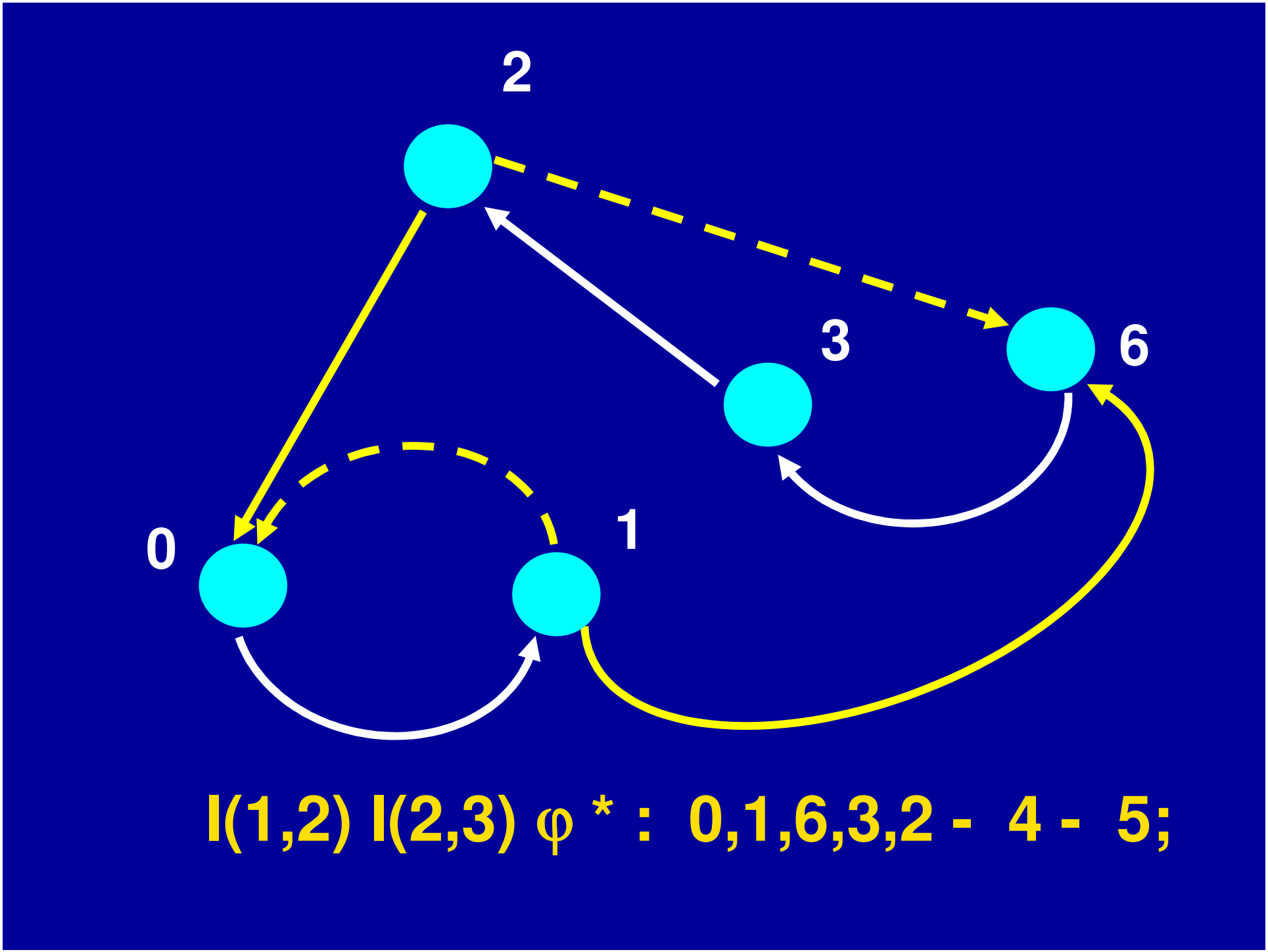
ed il costo totale

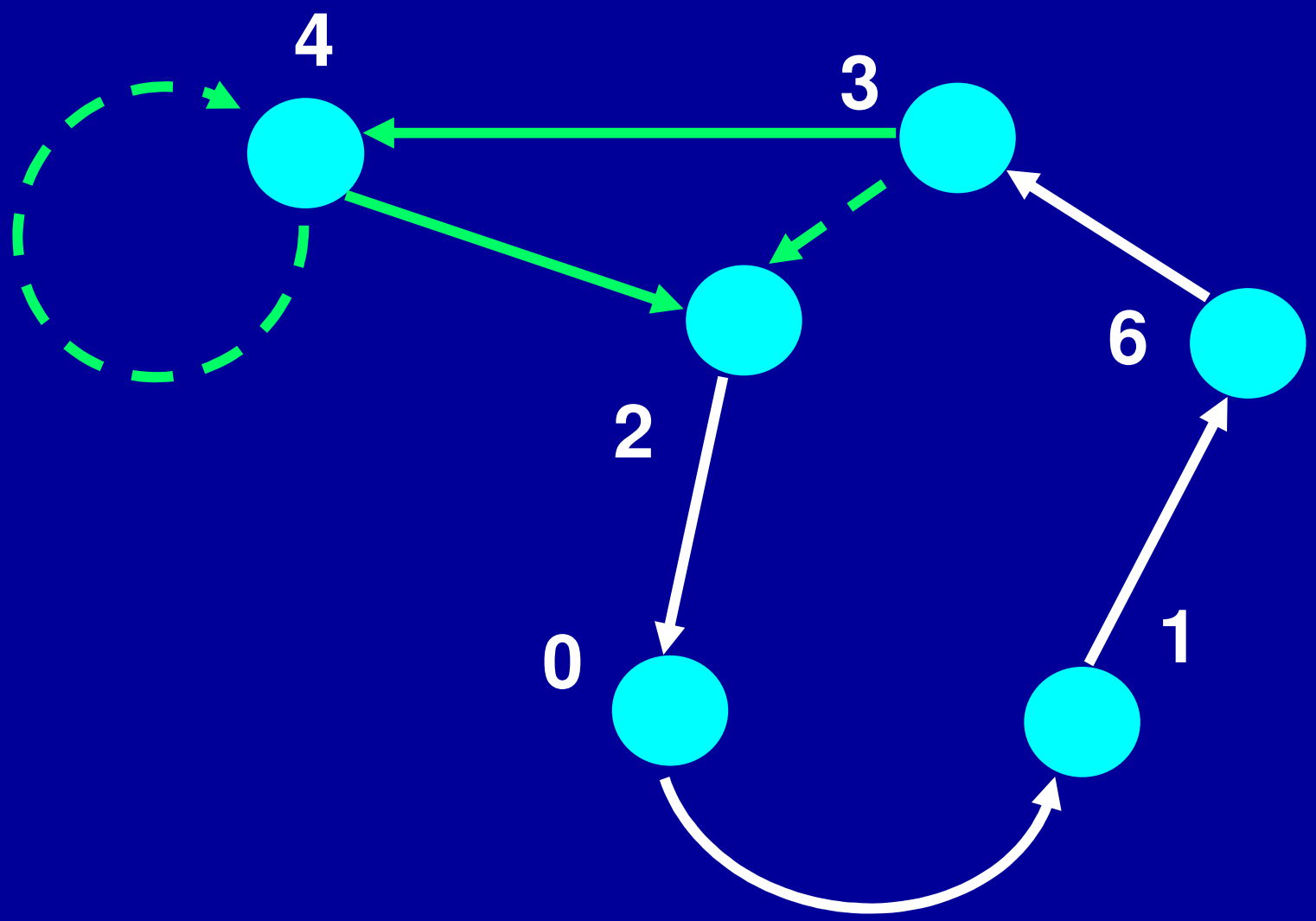
3 + 15 + 5 + 3 + 8 + 15 + 8 = 57.

°dopo gli scambi di cui a seguire

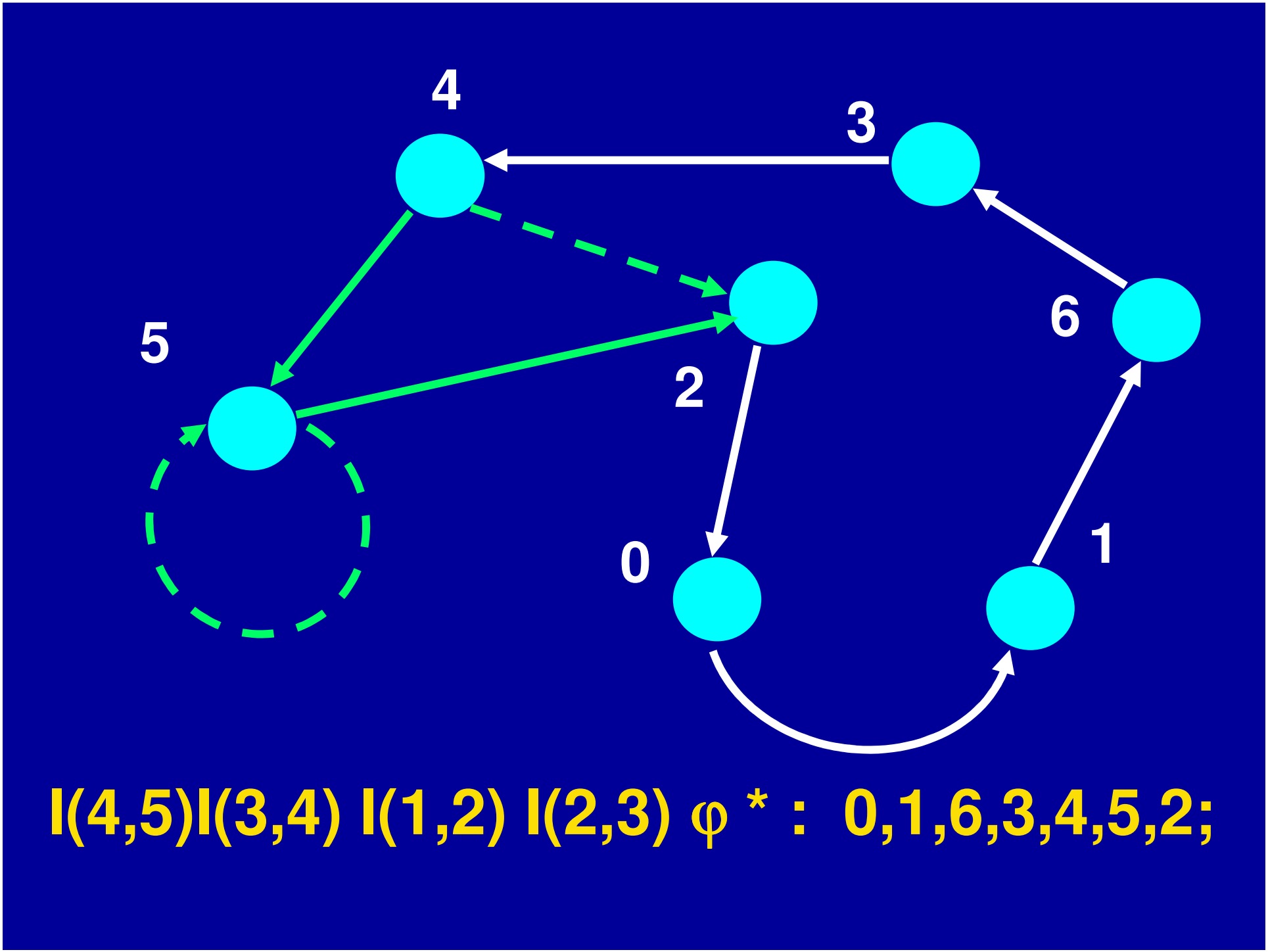


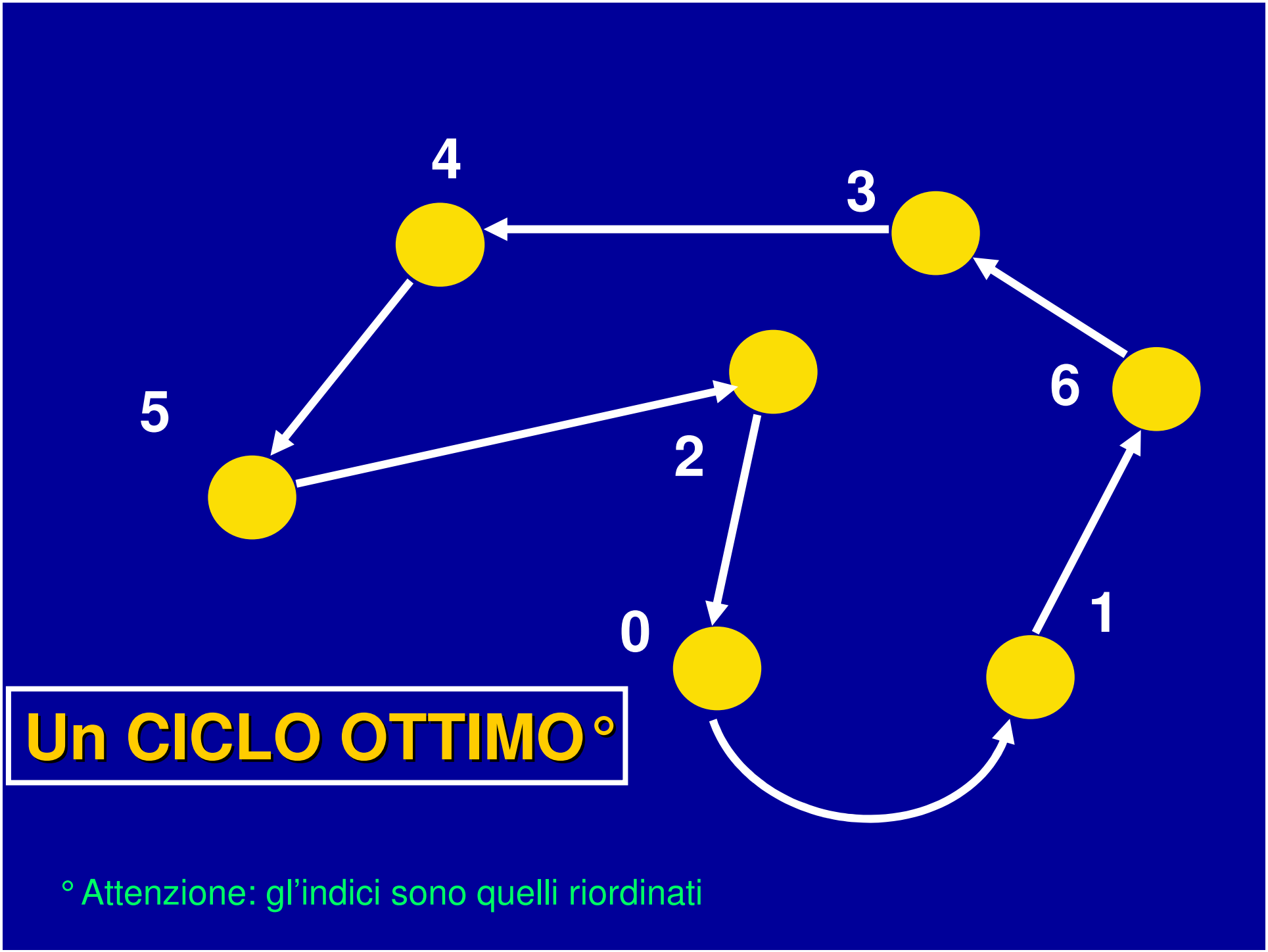
I(2,3) φ * : 0,1 - 3, 2,6 - 4 - 5;


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°

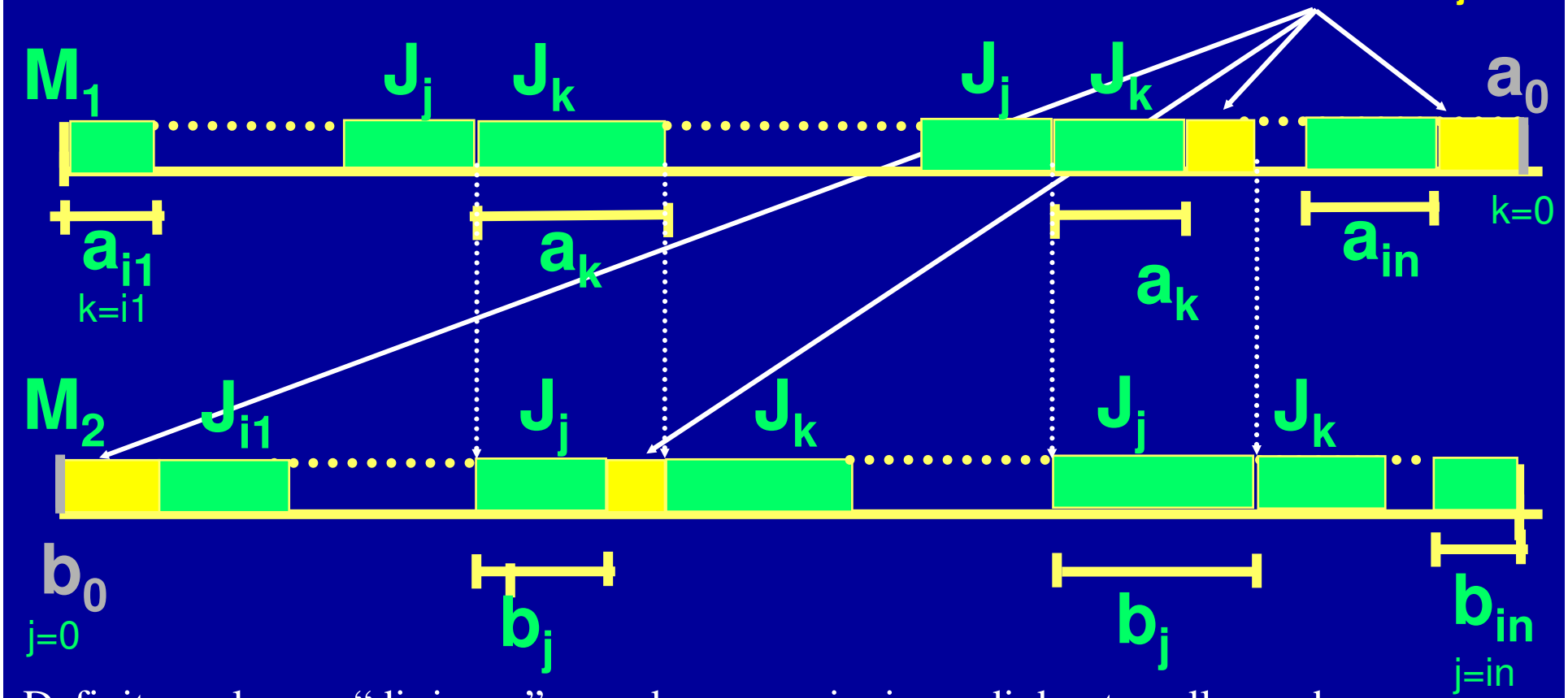
° Attenzione: gl'indici sono quelli riordinati

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

ESEMPIO 3): UNA APPLICAZIONE DI A*

Problema

Un flusso continuo di pezzi subisce ciascuno una sequenza di 10 operazioni: su una prima macchina le operazioni da O_1 a O_{v-1} e poi su una seconda macchina da O_v a O_{10}, senza buffer fra le due macchine.

Flusso continuo senza buffer significa che la prima macchina esegue la sottosequenza da O_1 a O_{v-1} nello stesso periodo di tempo in cui la seconda macchina esegue la sottosequenza da O_v a O_{10}.

UNA APPLICAZIONE DI A*

Problema

Le operazioni richiedono i seguenti tempi:

Operazioni	1	2	3	4	5	6	7	8	9	10
Tempi	3	2	1	4	5	6	4	3	2	1

Tra le operazioni vi sono le seguenti incompatibilità, dovute alla condivisione di risorse terze, indicate con lettere maiuscole: $(O_1O_3O_6)_A$; $(O_2O_7)_B$; $(O_4O_7)_C$; $(O_6O_8)_D$.

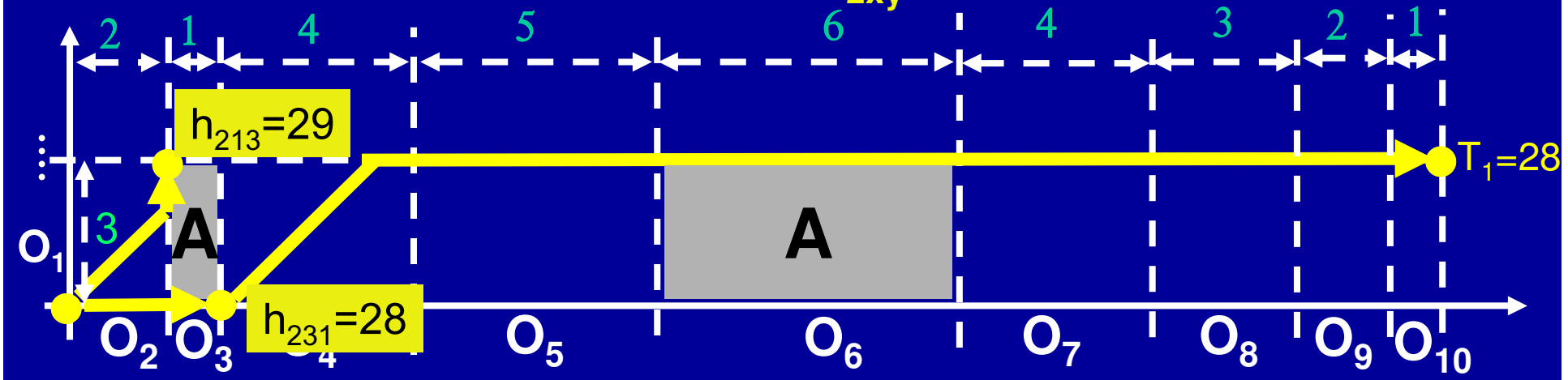
Utilizzando il grafo generato dalla griglia di condivisione delle risorse terze, si scelga v (ovviamente da 2 a 10) in modo da minimizzare il tempo di ciclo, cioè di esecuzione in parallelo delle due sottosequenze.

Si noti che, invece di usare 9 griglie, una per ciascun valore di v, che generano 9 grafi con un'origine e una destinazione, si può usare una sola griglia per generare i 9 grafi, ciascuno con la sua origine e la sua destinazione.

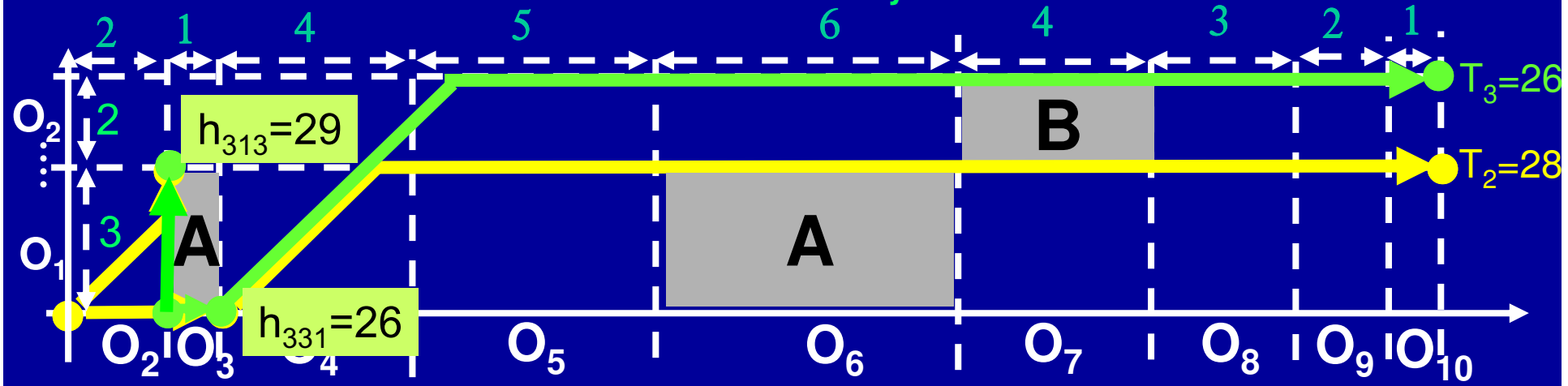
UNA APPLICAZIONE DI A*

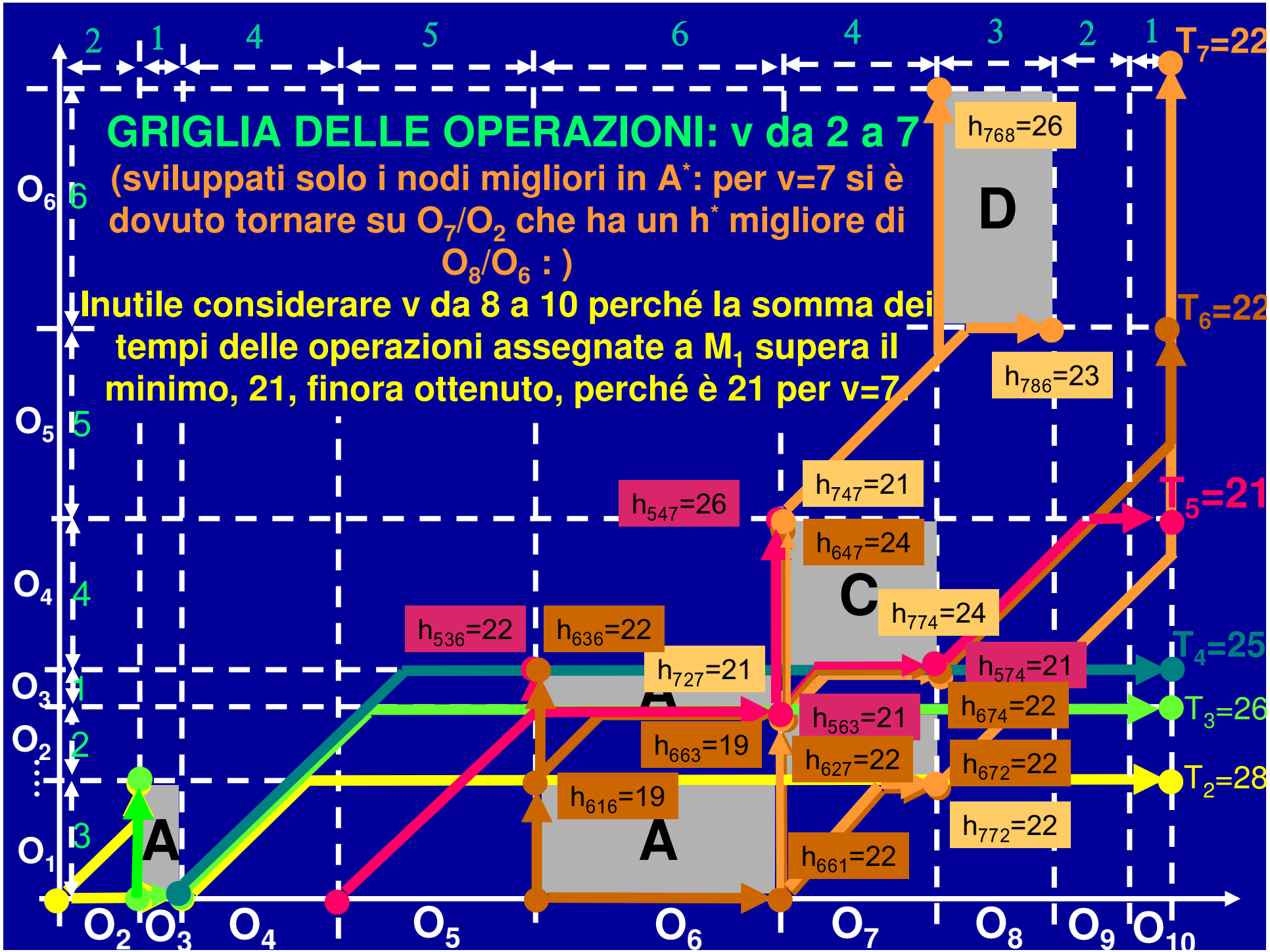
h_{ixy}: stima del tempo minimo di un percorso che inizia con O_i e passa per O_x/O_y

GRIGLIA DELLE OPERAZIONI e h_{2xy}: v = 2



GRIGLIA DELLE OPERAZIONI e h_{3xy}: v = 3




GRIGLIA DELLE OPERAZIONI: v da 2 a 7
(sviluppati solo i nodi migliori in A*: per v=7 si è dovuto tornare su O7/O2 che ha un h* migliore di O8/O6 :)
Inutile considerare v da 8 a 10 perché la somma dei tempi delle operazioni assegnate a M1 supera il minimo, 21, finora ottenuto, perché è 21 per v=7
2 1 4 5 6 4 3 2 1
T7=22
T6=22
T5=21
T4=25
T3=26
T2=28
O6 6
O5 5
O4 4
O3 1
O2 2
O1 3
O2 O3 O4 O5 O6 O7 O8 O9 O10
h768=26
h786=23
h747=21
h547=26
h647=24
h774=24
h536=22
h636=22
h727=21
h574=21
h674=22
h563=21
h663=19
h627=22
h672=22
h616=19
h772=22
h661=22
D
C
A
A
A