רשתות מהירות – חורף תשע"ח
199815
הרצאה מספר 9

יוסי קניזו
ykanizo@telhai.ac.il

המכללה האקדמית תל-חי
למדע, סביבה וחברה

Source: Nick Mckeown, Stanford

איפה אנחנו?

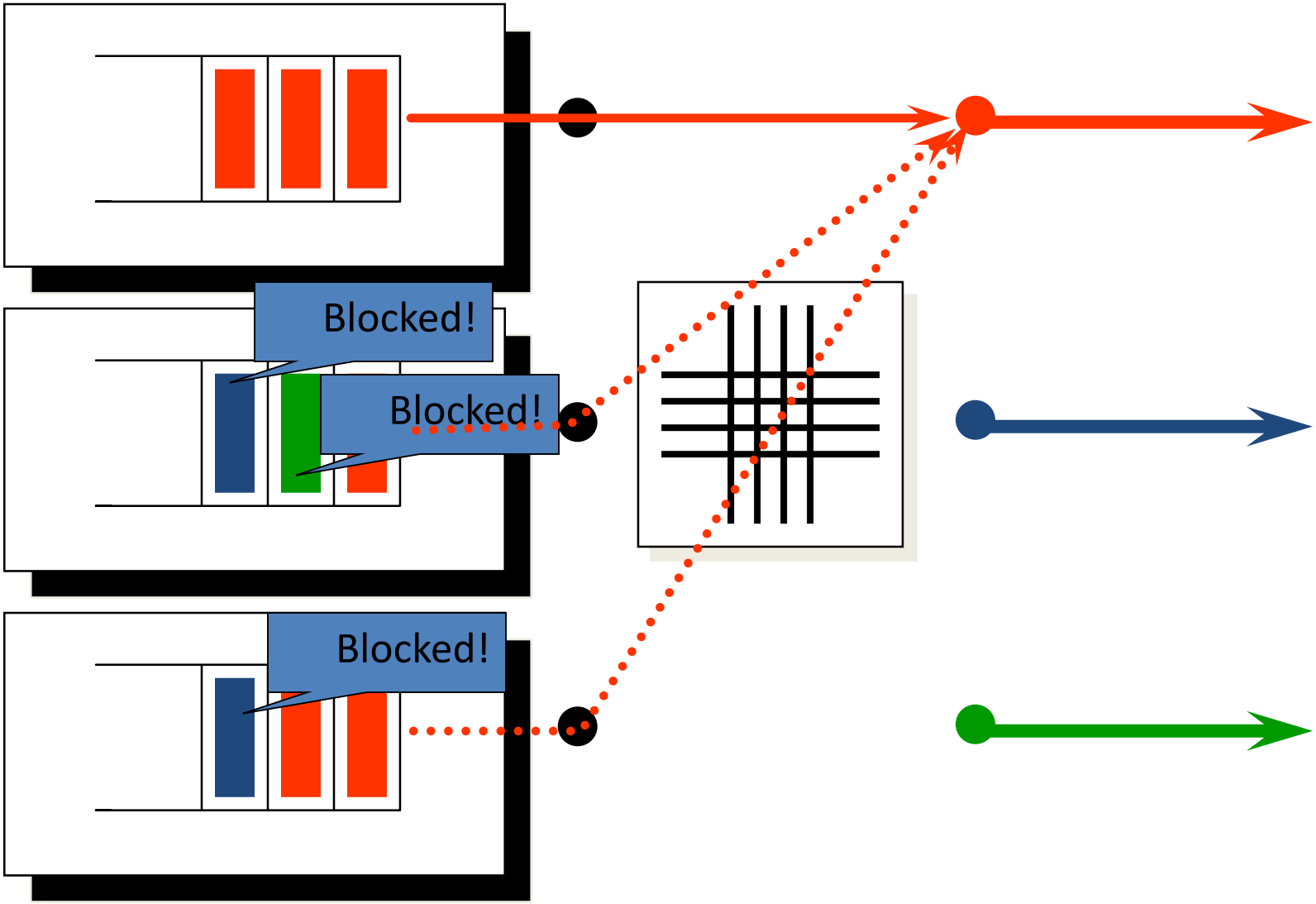
- הצגנו נתבים מבוססי תורי כניסה עם תורי יציאה וירטואלים.
- ראינו שבעיית חסימת ראש התור נפתרת.
  - אבל בכלליות צריך אלגוריתמים טובים להשגת 100% תפוקה.
- ניתן להשיג 100% תפוקה כאשר מטריצת התעבורה ידועה.

איפה אנחנו?

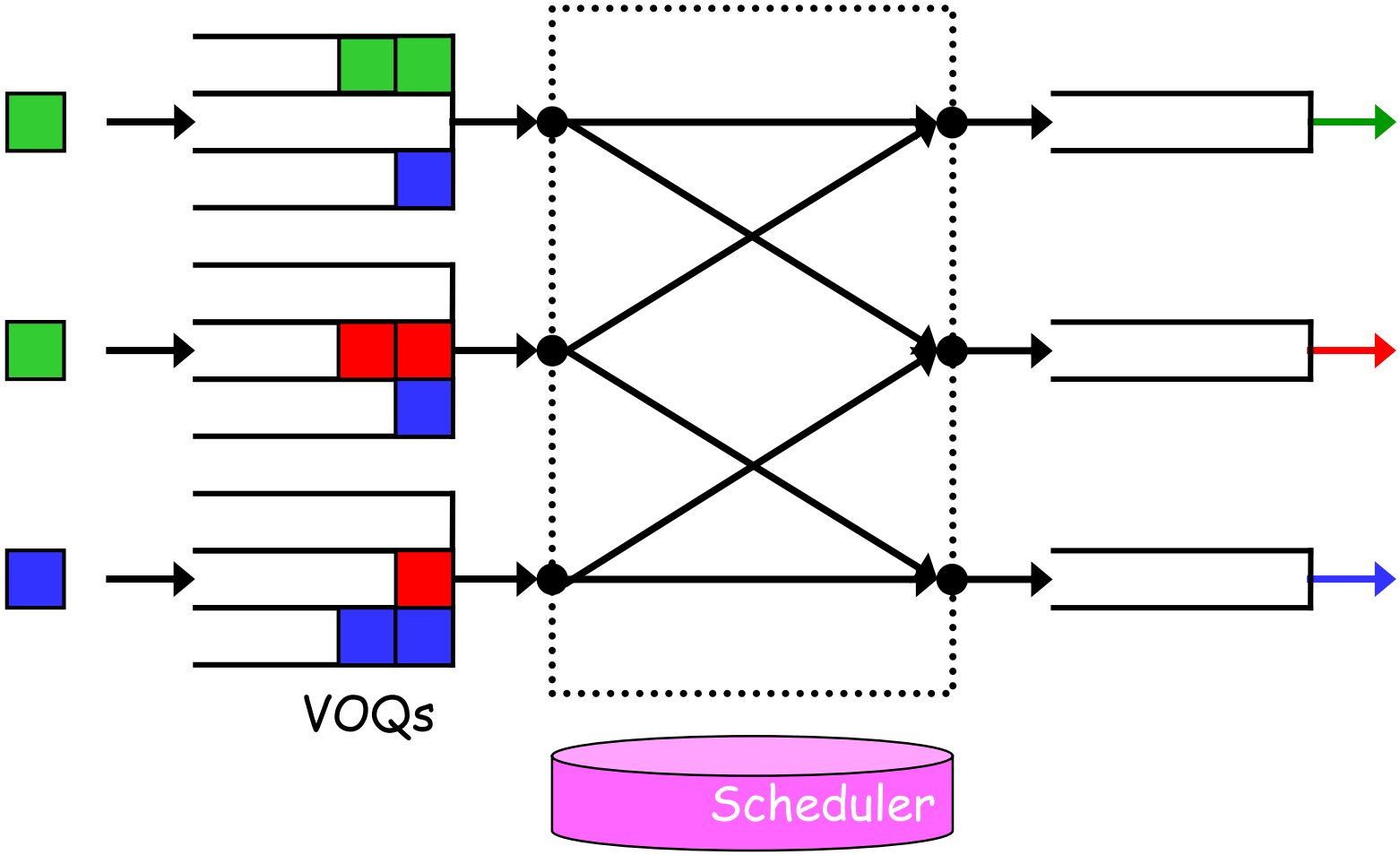
- ניתן להשיג 100% תפוקה גם כאשר מטריצת התעבורה אינה ידועה, על ידי אלגוריתם מבוסס שידוך מקסימום ממשוקל.
  - אבל דורש סיבוכיות גדולה של זמן ריצה.

- היום נמשיך ללמוד על נתבים מבוססי תורי כניסה עם תורי יציאה וירטואלים.
  - שימוש באלגורתמים מבוססים שידוך מקסימלי.

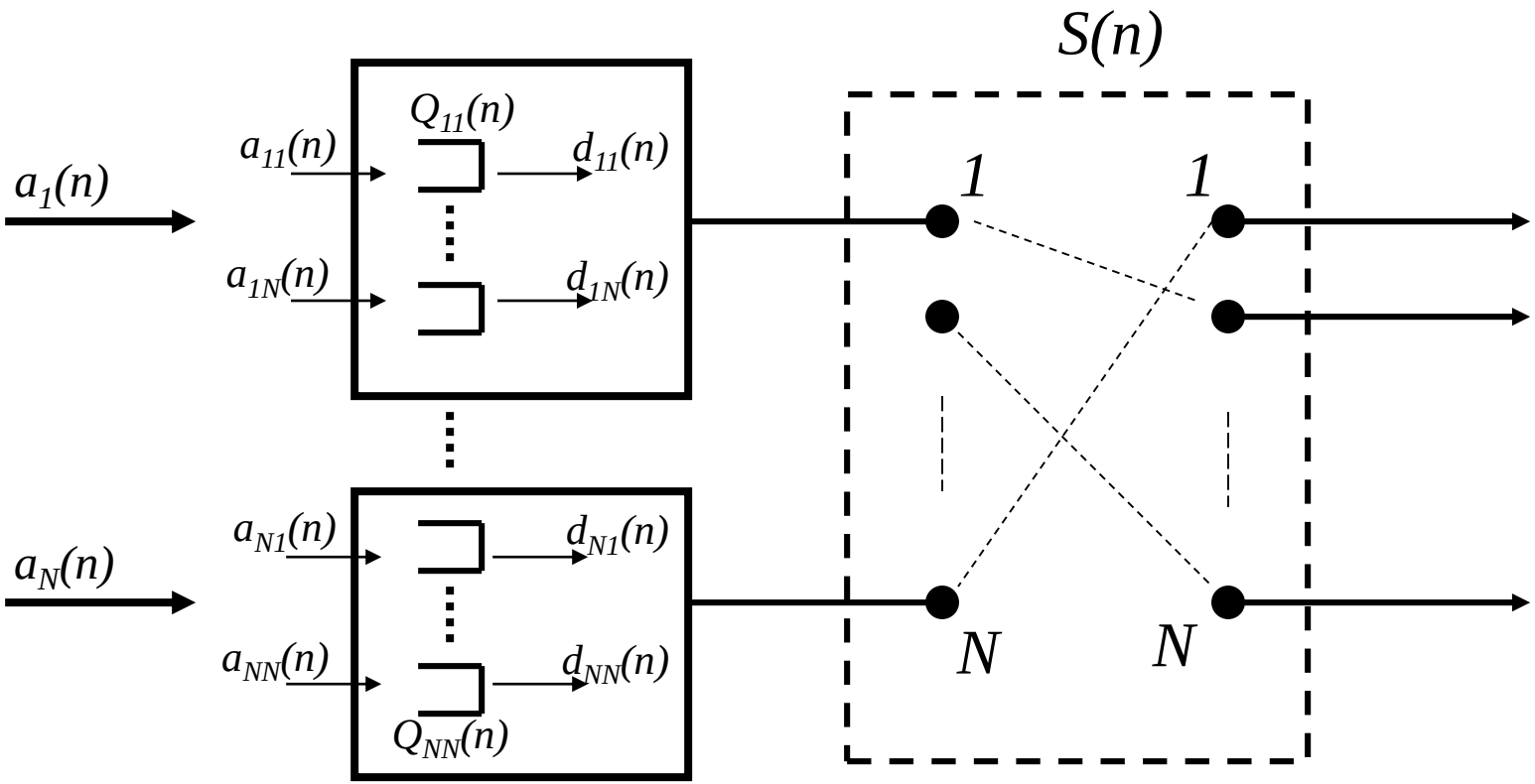
Head-of-Line Blocking



VOQs: איך החבילות מועברות



מודל הנתב


S(n)
Q11(n)
a11(n)
d11(n)
a1(n)
a1N(n)
d1N(n)
1
1
aN1(n)
dN1(n)
aN(n)
aNN(n)
dNN(n)
QNN(n)
N
N

סימונים והגדרות: הגעת חבילות

- $a_{ij}(n)$: חבילות המגיעות מכניסה i ומיועדות ליציאה j ב-time-slot ה-n.
  - מתקיים $a_{ij}(n) = 0$ או $a_{ij}(n) = 1$.
- λ_{ij}: קצב ההגעה הממוצע מכניסה i ליציאה j.
- מטריצת התעבורה היא:

$$\Lambda = \begin{pmatrix} \lambda_{11} & \cdots & \cdots & \cdots & \lambda_{1N} \\ \vdots & \ddots & & \iddots & \vdots \\ \vdots & & \lambda_{ij} & & \vdots \\ \vdots & \iddots & & \ddots & \vdots \\ \lambda_{N1} & \cdots & \cdots & \cdots & \lambda_{NN} \end{pmatrix}$$

סימונים והגדרות: מטריצת התעבורה

- מטריצת התעבורה היא קבילה אם:

– לכל i מתקיים $\sum_j \lambda_{ij} < 1$. כלומר, הקצב הכולל מכניסה מסויימת תמיד קטן (ממש) מ-1.
  - אף כניסה לא "מציפה" את הנתב.

– לכל j מתקיים $\sum_i \lambda_{ij} < 1$. כלומר, הקצב הכולל המגיע ליציאה מסויימת (מכל הכניסות יחד) קטן (ממש) מ-1.
  - אף יציאה לא "מוצפת" בנתב.

סימונים והגדרות: מטריצת השירות

- $Q_{ij}(n)$: אורך התור בתור יציאה הוירטואלי השייך לכניסה i ויציאה j.
- $S_{ij}(n)$: האם ב-time-slot ה-n, מועברת חבילה מכניסה i ליציאה j.
  - $S_{ij}(n) = 0$ או $S_{ij}(n) = 1$.
- $S=[S_{ij}(n)]$ היא מטריצת השירות.
  - מגדירה את החיבוריות (הפיזית) ב-time-slot ה-n בין הכניסות ליציאות.

תנאי על מטריצת השירות

- בכל time-slot:
  - כל יציאה מחוברת בדיוק לכניסה אחת.
  - כל כניסה מחוברת בדיוק ליציאה אחת.
- נובע מכך שבמטריצת השירות S, מתקיים כי בכל שורה ובכל עמודה יש בדיוק פעם אחת 1 (וכל שאר האיברים הם 0).
  - זוהי למעשה מטריצת פרמוטציה.

אלגוריתם התאמה

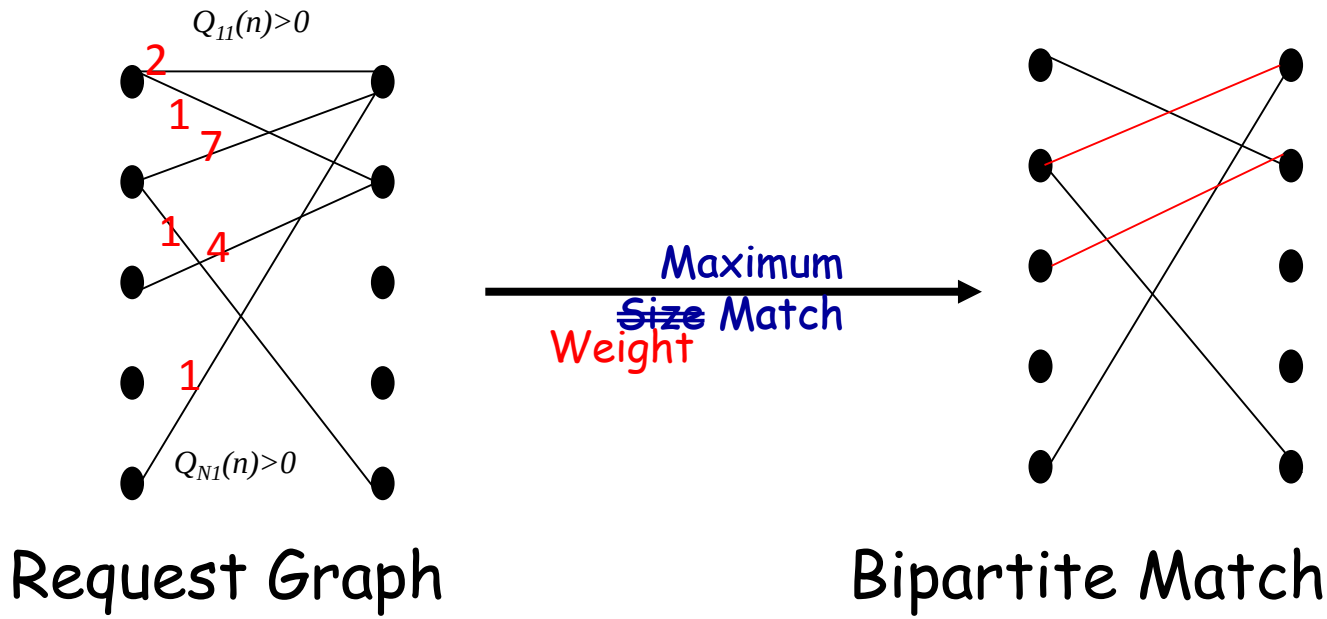
- תפקיד אלגוריתם ההתאמה הוא למצוא את *S(n)* בכל time-slot.
  - לעיתים על ידי שימוש במטריצת התעבורה Λ (אם היא ידועה).
  - ובדרך כלל, על ידי שימוש במצב התורים *Q(n)*, מאחר ומטריצת התעבורה Λ אינה ידועה.
- מטרה:
  - "100% תפוקה".
  - מטרה משנית, למזער ככל הניתן את ההשהיות.

מה זה "100% תפוקה"?

- נתב שאינו משמר עבודה
  - נתב מבוסס תורי כניסה (גם עם תורי יציאה וירטואלים) אינו משמר עבודה באופן כללי.
  - בספרות המקצועית ישנו מספר הגדרות שלא נביאן.
    - במידת הצורך, נגדיר בהמשך.
  - אנחנו נדרוש שקצב השירות הממוצע של כל תור במערכת יהיה גדול מקצב ההגעה הממוצע.

שידוך מקסימום ממושקל (MWM)

- Longest queue first (LQF) – העדף לשרת תורי יציאה וירטואלים עמוסים יותר קודם.



- אלטרנטיבה – העדף לשרת את החבילה שממתינה הכי הרבה זמן (OCF).

ביצועי LQF

- ניתן להראות שאלגוריתם LQF, כלומר MWM עם משקל לפי גודל התורים, משיג 100 אחוז תפוקה גם על תעבורה לא אחידה (ולא ידועה).
  - הוכחה מתמטית לא במסגרת הקורס.

- האם LQF תמיד משרת את התור הכי גדול?
  - לא, הוא רק נותן משקל הכי גדול לתור הכי גדול.

הנושאים להיום

אלגוריתמים מבוססים שידוך מקסימלי:

- iLQF
- WFA
- PIM ו-iSLIP.

סיבוכיות של אלגוריתמי שידוך

- שידוך מקסימום
  - סיבוכיות אופיינית: $O(N^{2.5})$.
- שידוך מקסימום ממושקל
  - סיבוכיות אופיינית: $O(N^3)$.

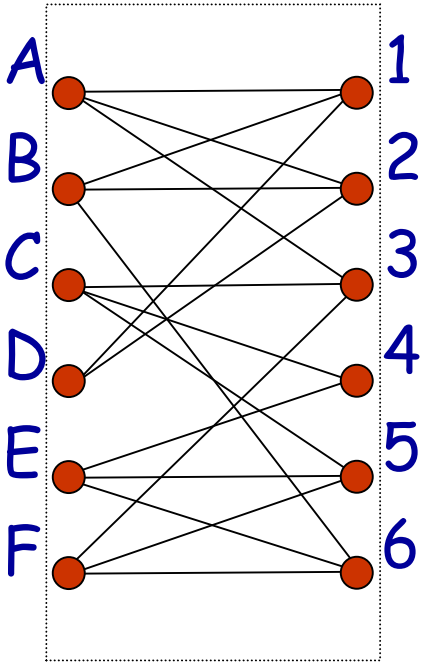
- באופן כללי:
  - קשה למימוש בחומרה.
  - איטי.

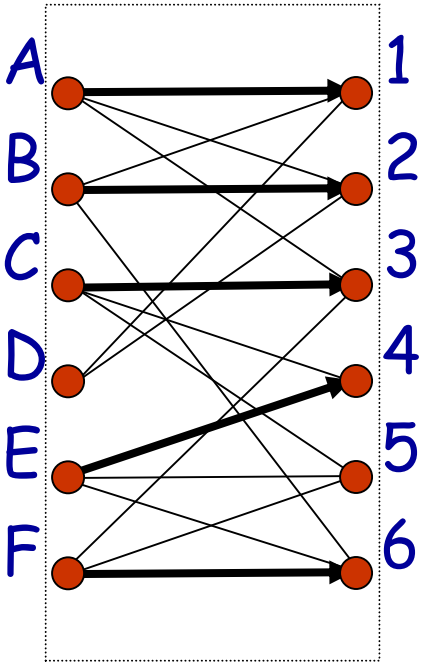
- האם ניתן למצוא אלגוריתמים מהירים יותר?

שידוך מקסימלי

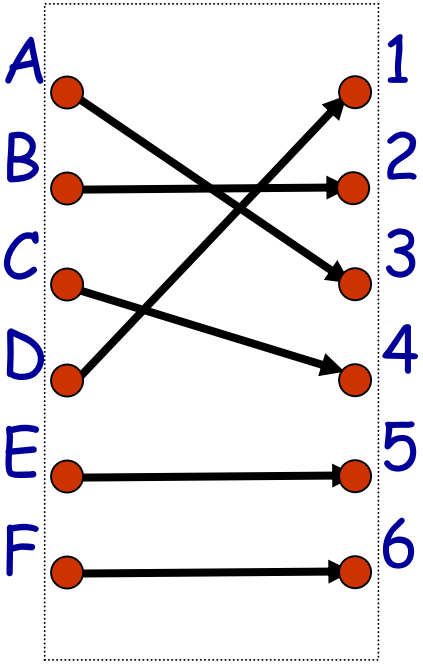
- שידוך מקסימלי הינו שידוך שלא ניתן להגדיל יותר על ידי הוספת קשתות נוספות.
- בדרך כלל בונים שידוך מקסימלי בתהליך איטרטיבי בו מוסיפים כל פעם קשת לשידוך מבלי להוציא קשתות שכבר הוספו.
- סיבוכיות זמן הריצה של אלגוריתמי שידוך מקסימלי הם קטנים יותר באופן מהותי.
  - $O(n)$, כאשר n מספר הצמתים.

דוגמא לשידוך מקסימלי





Maximal
Size Matching



Maximum
Size Matching

שידוך מקסימלי

- באופן כללי, שידוך מקסימלי קל הרבה יותר לממש והוא מהיר יותר.
- הגודל של (כל) שידוך מקסימלי הוא לפחות חצי מהגודל של שידוך מקסימום.
- נלמד את האלגוריתמים הבאים:
  - Greedy LQF
  - WFA
  - PIM
  - iSLIP

אלגוריתם LQF גרידי

- אלגוריתם LQF גרידי (Greedy Longest Queue First) מוגדר באופן הבא:

1. בחר תור יציאה וירטואלי (VOQ) עם המספר הרב ביותר של חבילות. אם יש תיקו בין מספר תורים, בחר תור באופן אקראי. נניח שהתור הנבחר הוא $Q_{i,j}$.
2. מעתה לא ניתן לבחור יותר תורי יציאה וירטואליים שבכניסה i, או שמיועדים ליציאה j (בכניסות אחרות).
3. חזור על שלבים 1 ו-2, עד אשר לא ניתן להוסיף יותר תורים.

תכונות של LQF גרידי

- נקרא גם iLQF (iterative LQF), מאחר והוא מתבצע באיטרציות.
- האלגוריתם מתכנס ב- N צעדים.
  - תחת ההנחה שנתונה רשימת התורים ממויינת.
- אלגוריתם iLQF תמיד מוצא שידוך מקסימלי.
- אלגוריתם iLQF תמיד מוצא שידוך שהגודל (והמשקל שלו) הוא לפחות חצי מהגודל (והמשקל) של השידוך מקסימום.

אפשר יותר מהר?

- נניח שרשימת התורים ממויינת.
  - ראינו שאלגוריתם iLQF עובד ב-N צעדים.
- אבל בכל צעד יש מספר פעולות שמתבצעות על מנת לעדכן\להוציא התורים מהרשימה.
- אמנם זה מתבצע ב- $O(N)=cN$, אבל עבור קבוע c כלשהו גדול מ-1.

אפשר יותר מהר?

- באופן כללי:
  - עבור אלגוריתם סדרתי, נרצה שהקבוע c יהיה כמה שיותר קרוב ל-1 (ואולי אפילו פחות?).
  - ניתן להשתמש באלגוריתמים המפעילים מקבול כלשהו.
  - ניתן להשתמש באלגוריתמים אשר כל "צעד" שלהם פועל בהרבה פחות זמן.

הנושאים להיום

אלגוריתמים מבוססים שידוך מקסימלי:

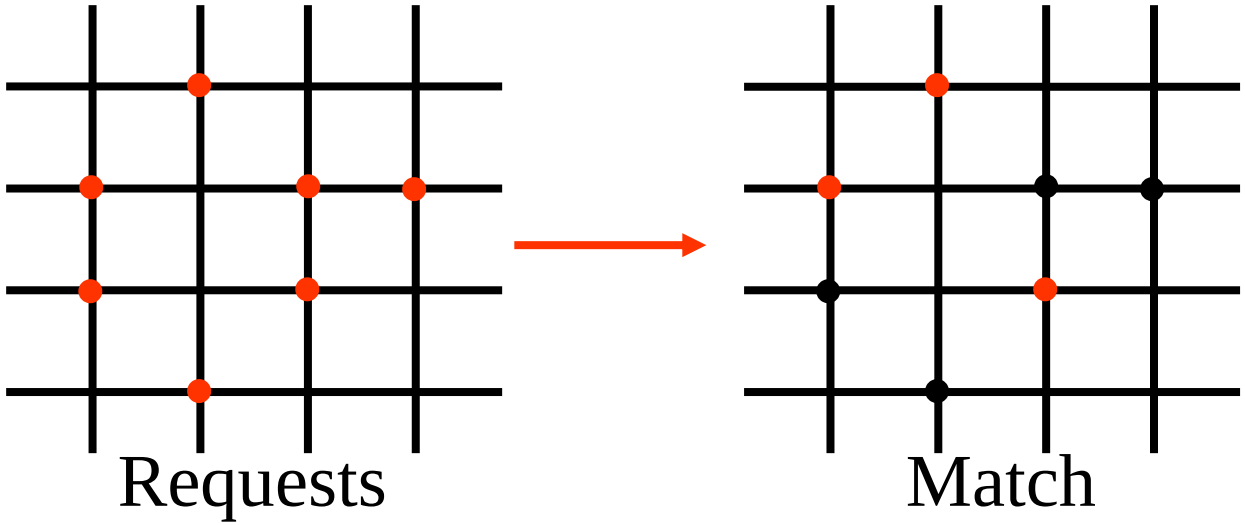
- iLQF
- WFA
- PIM ו-iSLIP.

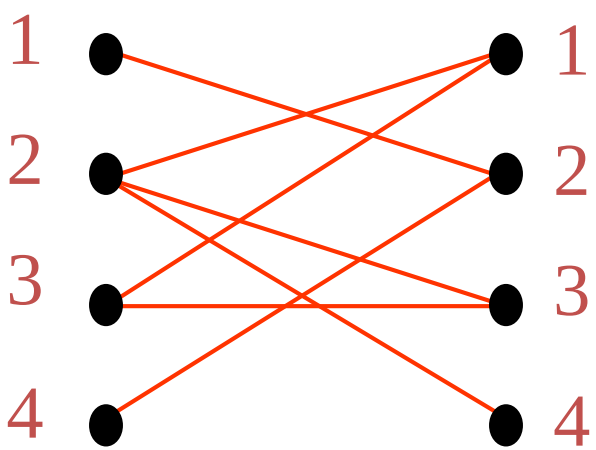
WFA (Wave Front Arbiter)

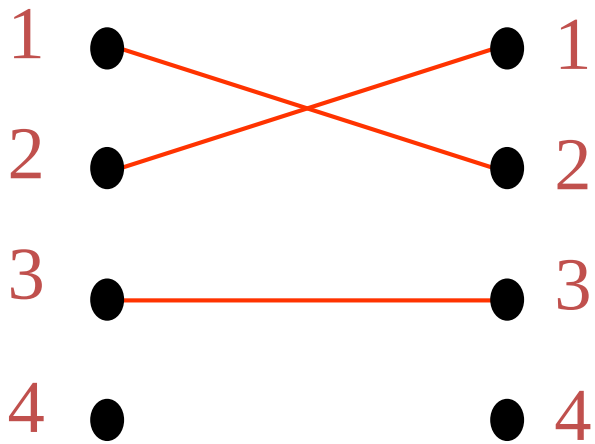
[Tamir and Chi, 1993]

- אלגוריתם למציאת שידוך מקסימלי הממומש בחומרה.
  - אינו משתמש בזיכרון כלל.
  - מבוסס על שערים לוגיים בלבד.
- ראשית נבין את אופן פעולתו.

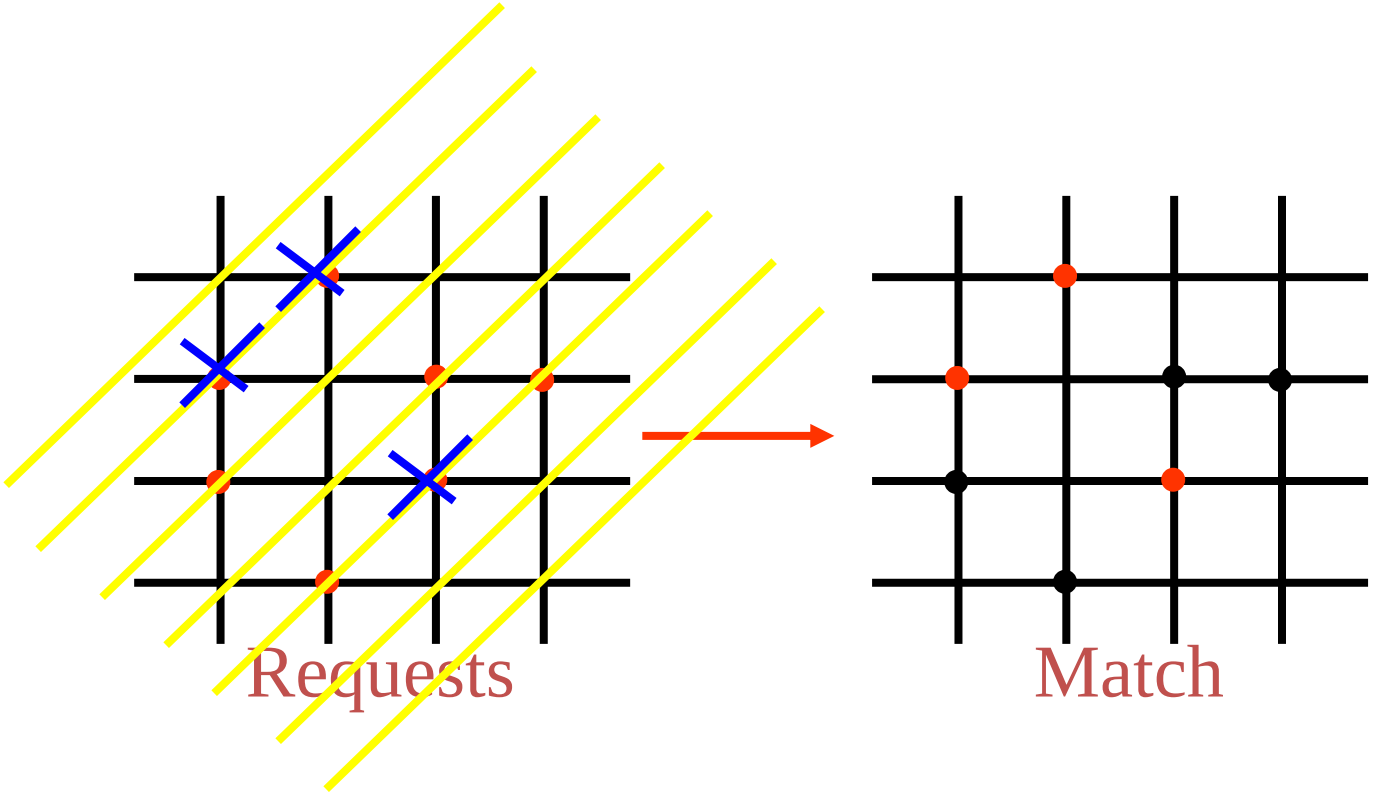
תרשים הבקשות ותרשים השידוך



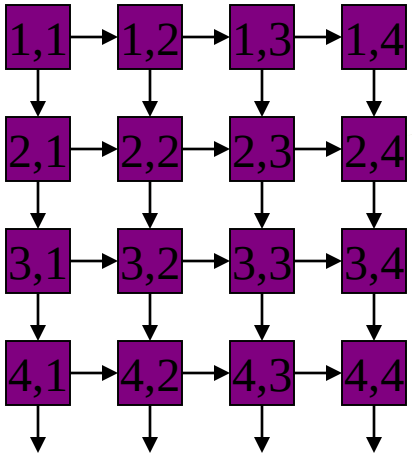




Wave Front Arbiter



מימוש אלגוריתם Wave Front



בלוקים לוגיים שאינם דורשים שימוש בזיכרון.

מימוש בלוק יחיד

- באופן כללי כל בלוק הוא בעל 3 כניסות (משמאל, למעלה, וכניסת "בקשה") , ו-3 יציאות (מימין ,מלמטה, ויציאת "אישור בקשה").
- בלוק יחיד צריך לממש את הפונקציה הבאה:
  - אם כניסת ה"בקשה" היא 0, הוא מעביר ליציאה מימין את מה שקיבל מהכניסה משמאל, ליציאה למטה את מה שקיבל בכניסה למעלה, וליציאה "אישור בקשה" הוא מוציא 0.
  - אם כניסת ה"בקשה" היא 1, הוא מוציא 1 ליציאה "אישור בקשה" אם ורק אם הכניסות משמאל ומלמעלה הן 0. אחרת הוא פועל כמו בסעיף הקודם.

מימוש בלוק יחיד

- כל בלוק מממש לוגיקה פשוטה מאוד.
- ניתן למימוש על ידי שערים סטנדטרטיים בחומרה (AND, OR, NOT וכו).
- אין שימוש בזיכרון בכלל, ולכן כל "צעד" הוא מהיר בהרבה.

תכונות אלגוריתם Wave front

- תמיד מוצא שידוך מקסימלי.
  - צמת תמצא בשידוך אם היא הראשונה עם בקשה בשורה ובעמודה שלה.
- נוטה להעדיף את התור Q_{11}.
  - דרוש מכניזם נוסף על מנת למנוע העדפה זו.
- באופן כללי ניתן למימוש מבוזר, ובלי שימוש בזיכרון.

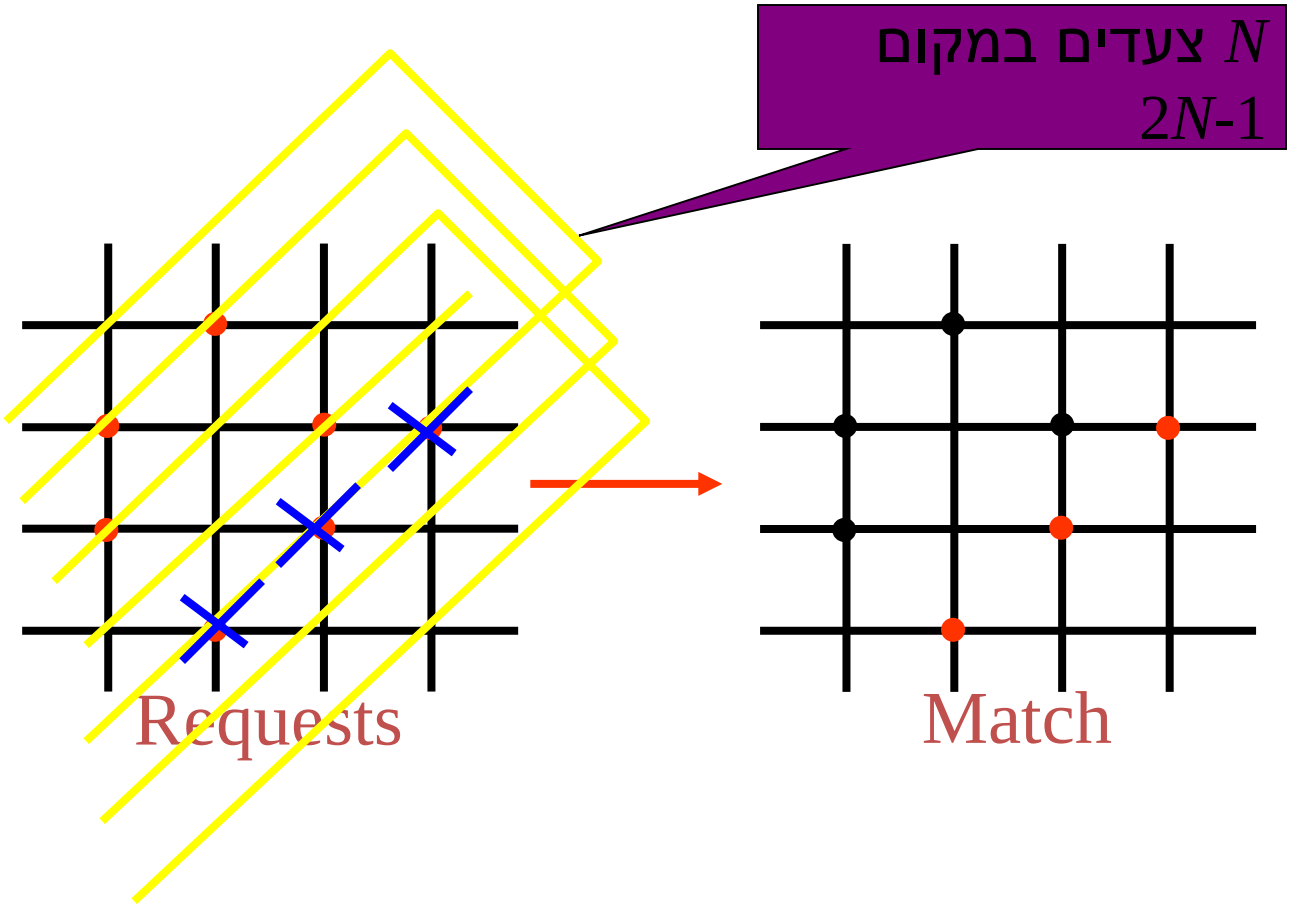
תכונות אלגוריתם Wave front

- אמנם פועל ב-*2N-1* צעדים, אבל כל צעד הוא מהיר מאוד מאחר ואין שימוש בזיכרון.

- בדרך כלל מתכנס תוך מחזור שעון אחד או מספר בודד של מחזורי שעון.

שיפור לאלגוריתם Wave Front

Wrapped WFA (WWFA)



תכונות *Wrapped WFA*

- מוצא תמיד שידוך מקסימלי.
  - מתי צמת עם בקשה תמצא בשידוך, ומתי לא?
  - תלוי במיקום שלה.
- עדיין נוטה להעדיף את התור Q_{11}.
- מעט מסובך יותר אבל מתכנס מהר יותר.

הנושאים להיום

אלגוריתמים מבוססים שידוך מקסימלי:

- iLQF
- WFA
- PIM ו-iSLIP.

Parallel Iterative Matching

- הכח הגדול של אלגוריתם *WFA* הוא שאינו משתמש כלל ברכיבי זיכרון.
  - מנצל את היות החומרה מהירה ביותר.
- נראה עתה אלגוריתם נוסף, שנוקט בגישה אחרת של מקבול התהליך למציאת שידוך מקסימלי.

Parallel Iterative Matching

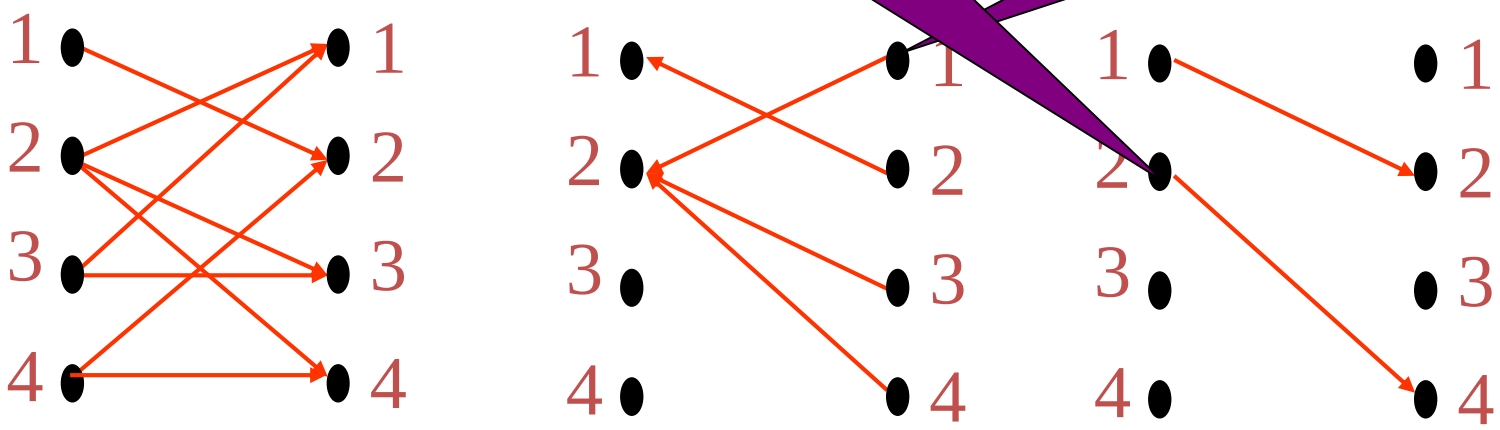
[A

בחירה אקראית

בחירה אקראית

Iteration:

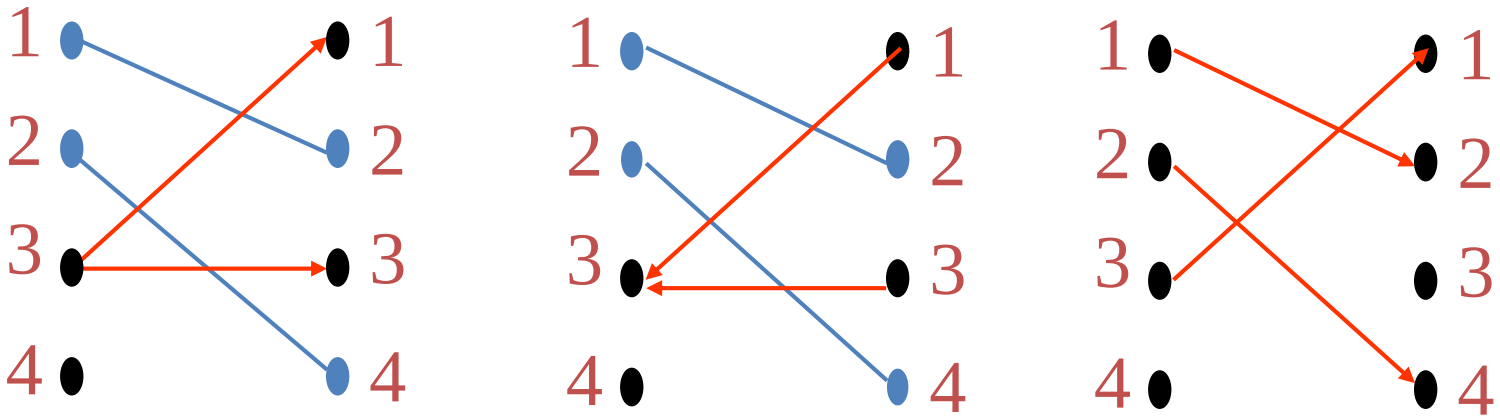
#1



ϕ1: Requests ϕ2: Grant ϕ3: Accept/Match

#2

.....



תכונות PIM

- בהכרח מתכנס לשידוך מקסימלי תוך N צעדים.
  - למה?
- בכל שלב, כל כניסה וכל יציאה יכולים לעבוד באופן בלתי תלוי בכניסות וביציאות האחרות.
- באופן כללי, צפוי להתכנס הרבה יותר מהר מ-N איטרציות.
- כמה איטרציות עלינו להריץ?

כמה איטרציות צריך?

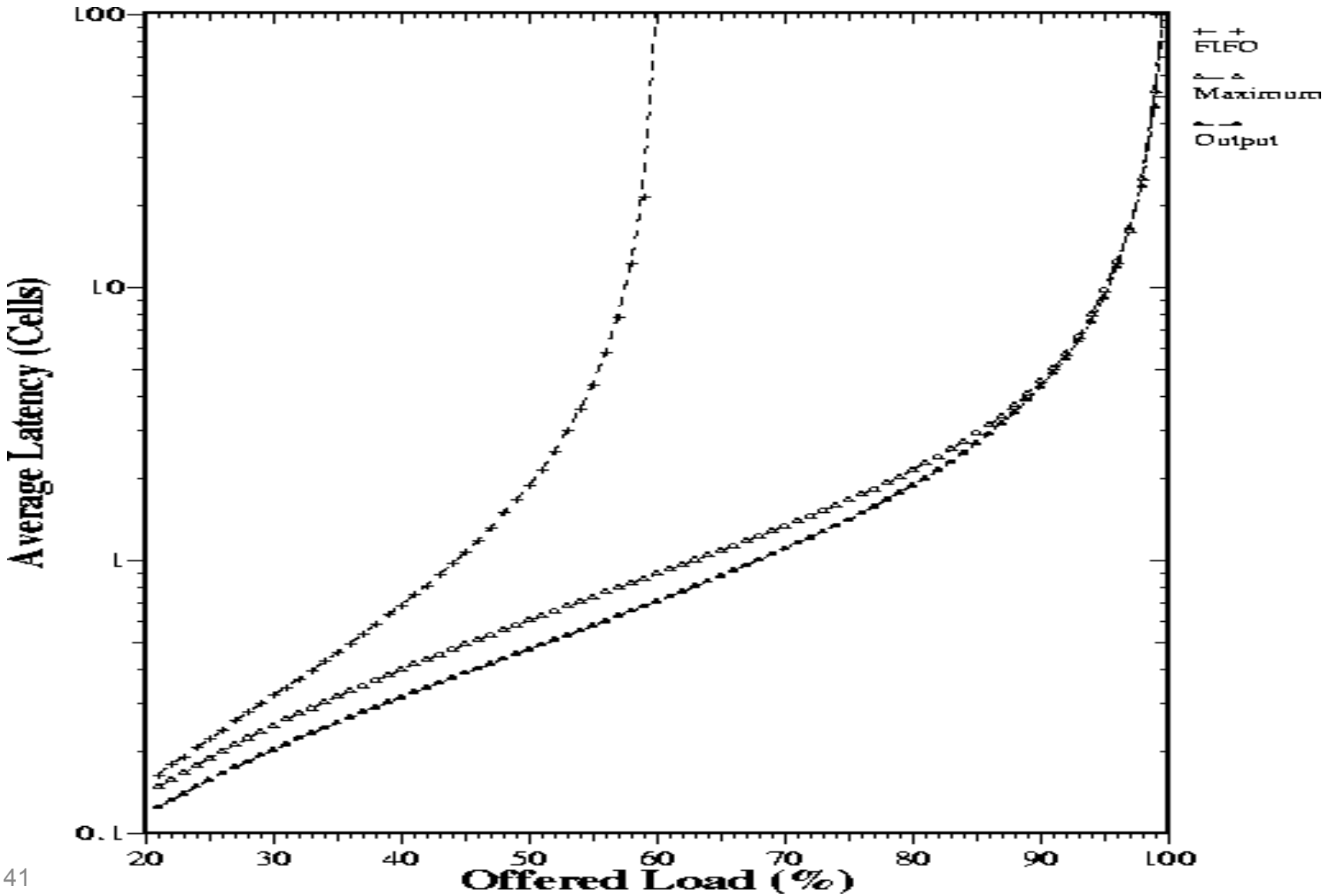
- נתונות התוצאות הבאות מהספרות:
- נסמן ב- U_i את מספר הצמתים שאינם בשידוך אבל ניתן להכניסן לשידוך.
  - אזי: $E(U_i) \leq N^2/4^i$
- כתוצאה מכך, אם C הוא מספר האיטרציות הדרושות להתכנסות נקבל: $E(C) \approx \log(N)$

Anderson et al., "*High-Speed Switch Scheduling for Local Area Networks,*" 1993.

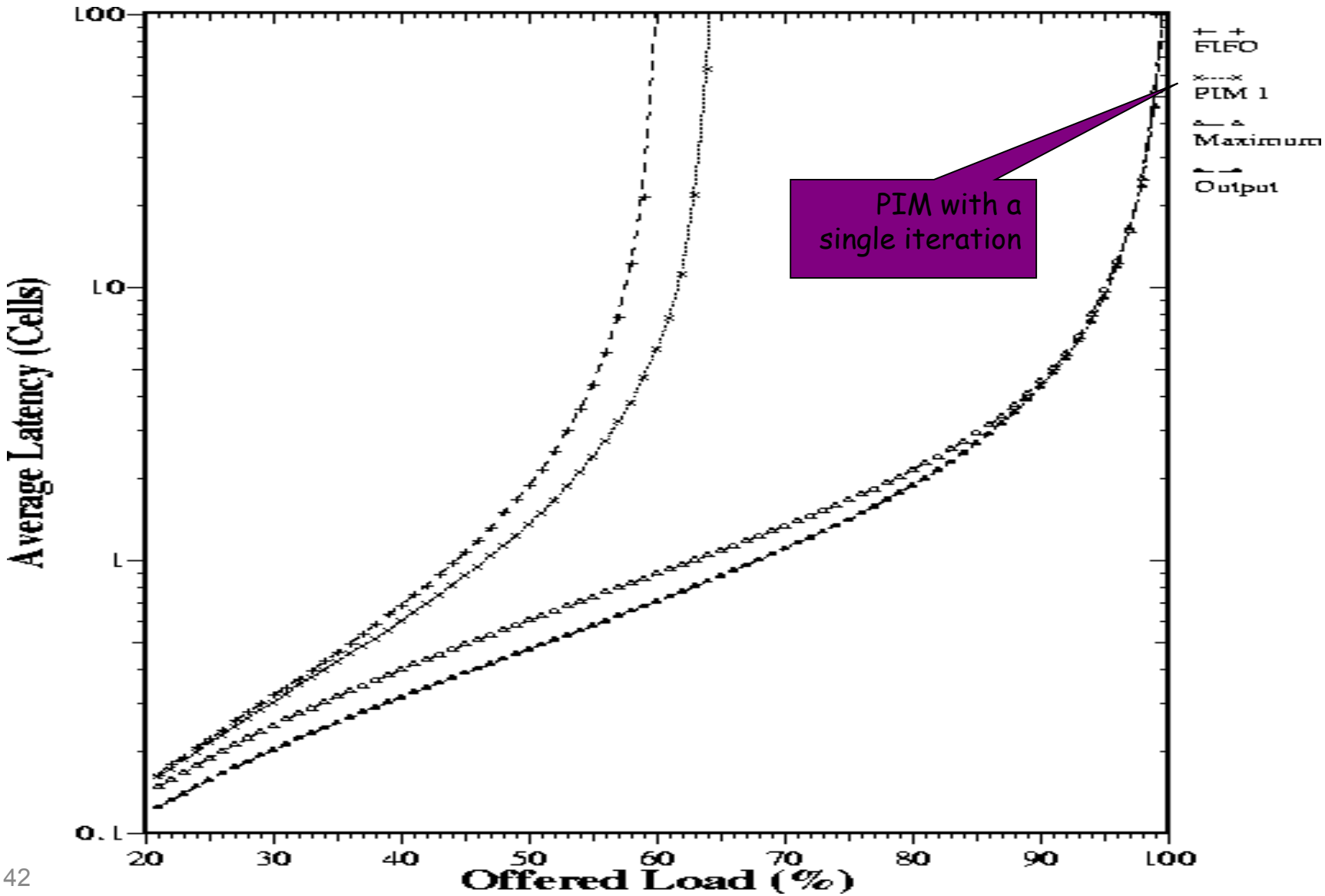
ביצועי PIM

- באופן כללי, קשה מאוד לנתח מתמטית את ביצועי אלגוריתמים מבוססי שידוך מקסימלי.
  - בדרך כלל הערכת הביצועים מבוצעת באמצעות סימולציות.
- בגרפים הבאים:
  - FIFO = תורי כניסה פשוטים ללא VOQs (58% תפוקה).
  - Maimum = אלגוריתם שידוך מקסימום.
  - Output = ביצועי נתב מבוסס תורי יציאה (הביצועים האופטימליים, כפי שלמדנו).

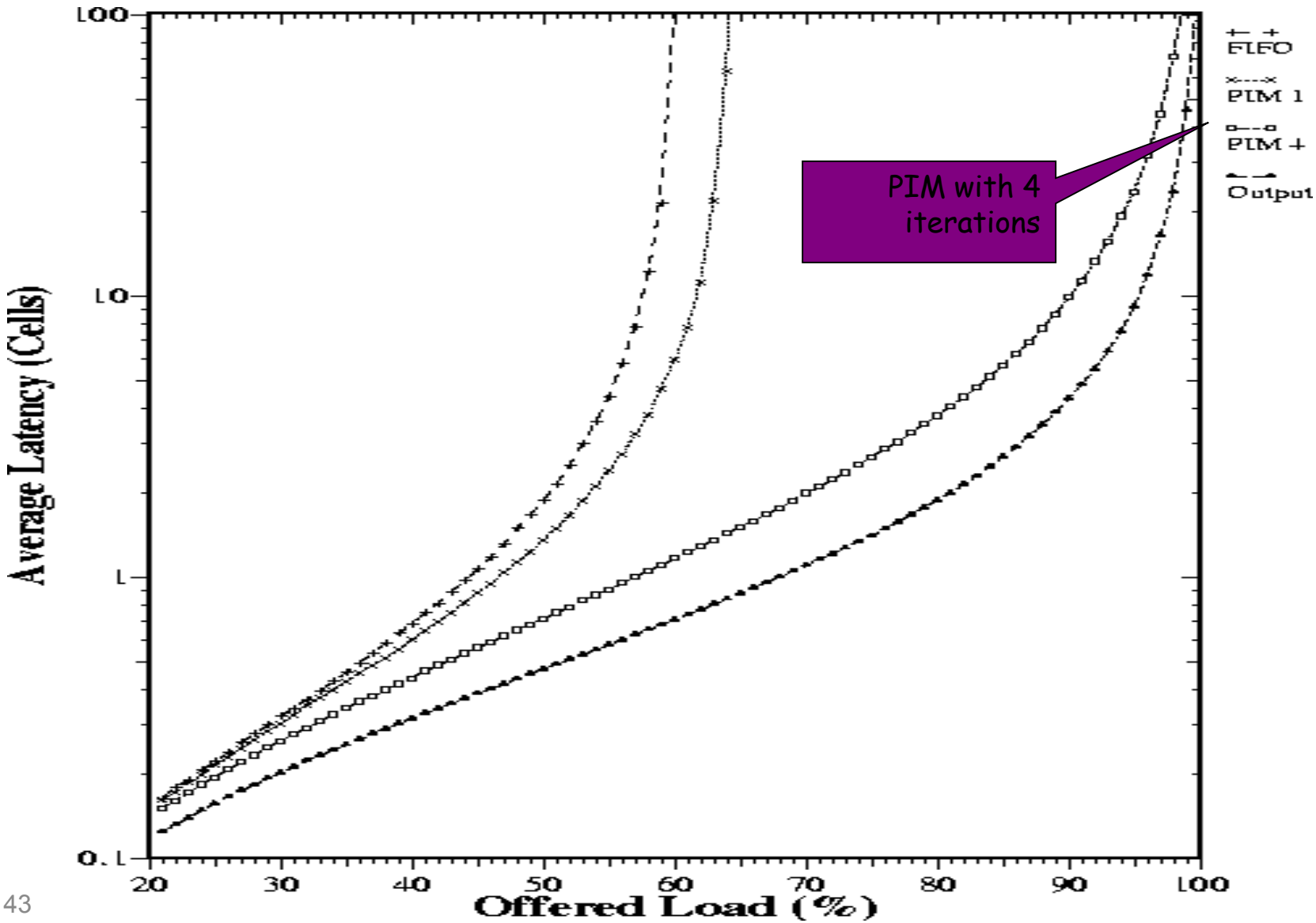
Parallel Iterative Matching



Parallel Iterative Matching



Parallel Iterative Matching



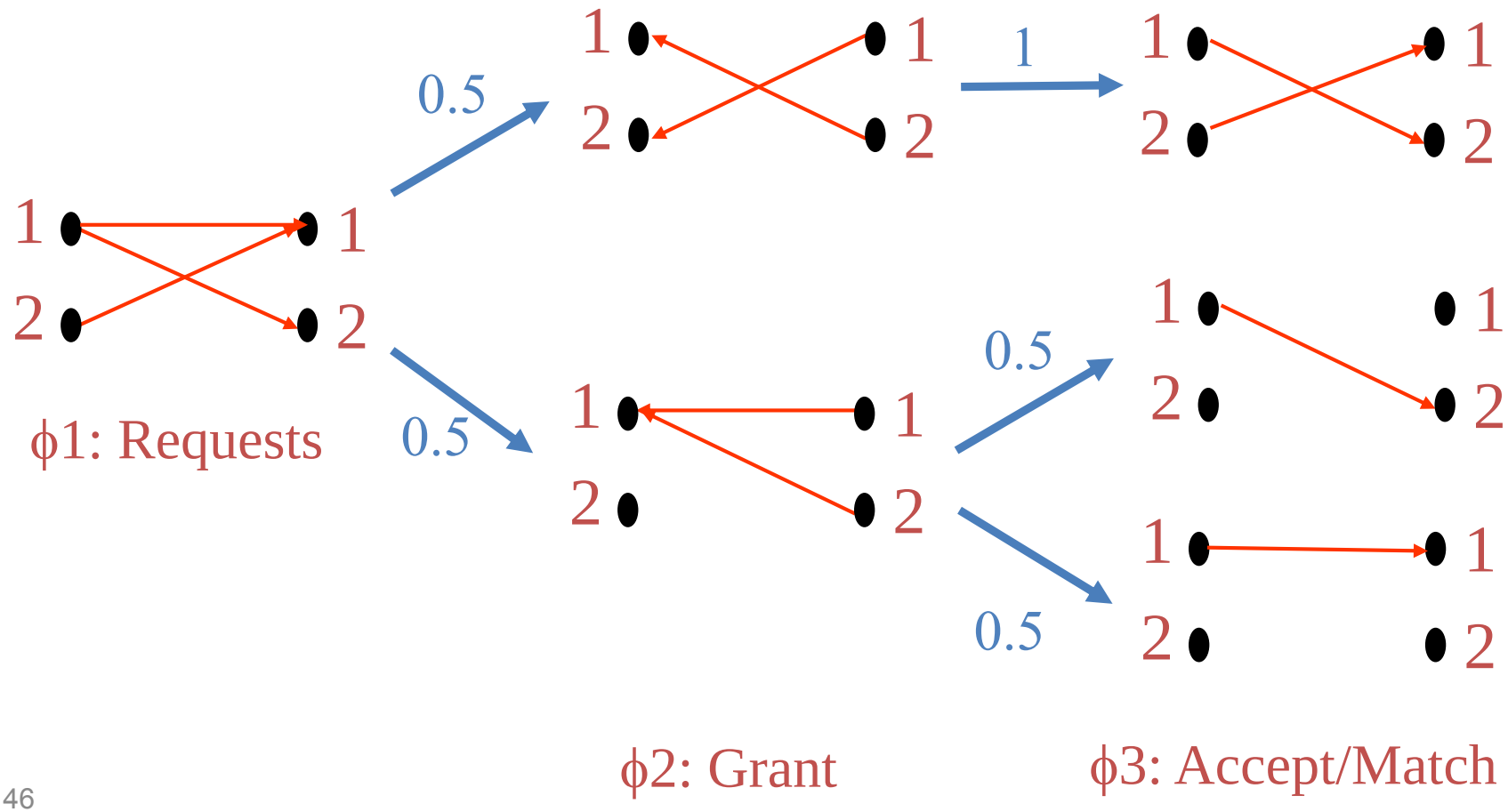
iSLIP

- אלגוריתם PIM יעיל מאוד, אבל בעומסים גבוהים, ישנן שתי בעיות:
  - איטרציות מיותרות שנובעות מאופי הפעולה האקראי שלו.
    - בשלב ה-grant ובשלב ה-accept הבחירה היא אקראית.
  - חוסר הוגנות.

חוסר הוגנות PIM

- לצורך מתן אינטאיציה, נניח נתב 2x2, כאשר ישנן חבילות בכניסה הראשונה לשתי היציאות, ובכניסה השנייה ליציאה הראשונה בלבד.
  - בנוסף נניח כי מריצים רק איטרציה אחת.
- נציג בשקף הבא את תרשים עץ ההסתברויות.

חוסר הוגנות PIM - המשך

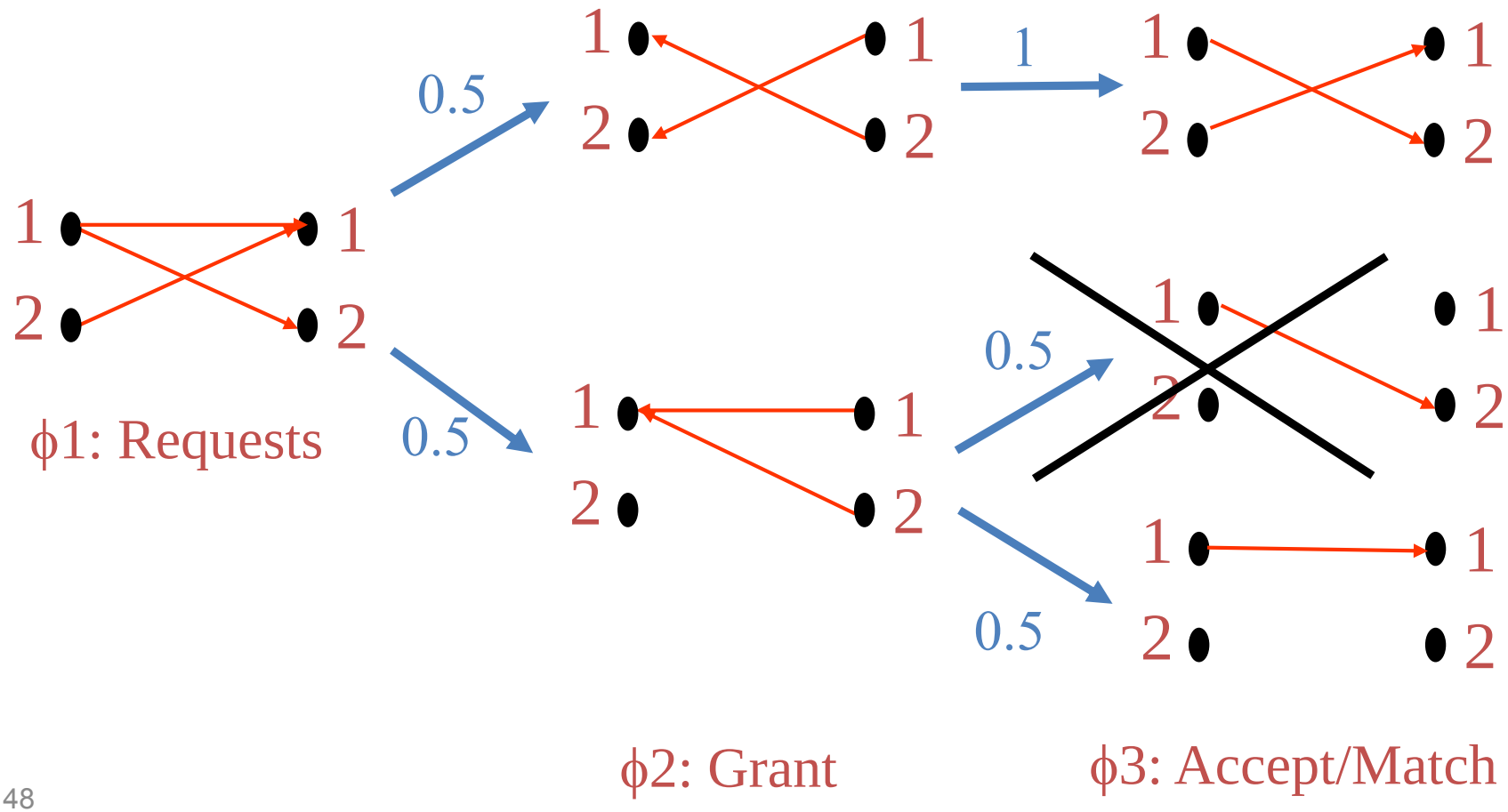


חוסר הוגנות PIM - המשך

- קיבלנו:
  - תור Q_{11} משורת בהסתברות 0.25.
  - תור Q_{12} משורת בהסתברות 0.75.
  - תור Q_{21} משורת בהסתברות 0.5.

- היה הוגן יותר עם כולם היו בהסתברות 0.5.

חוסר הוגנות PIM - המשך

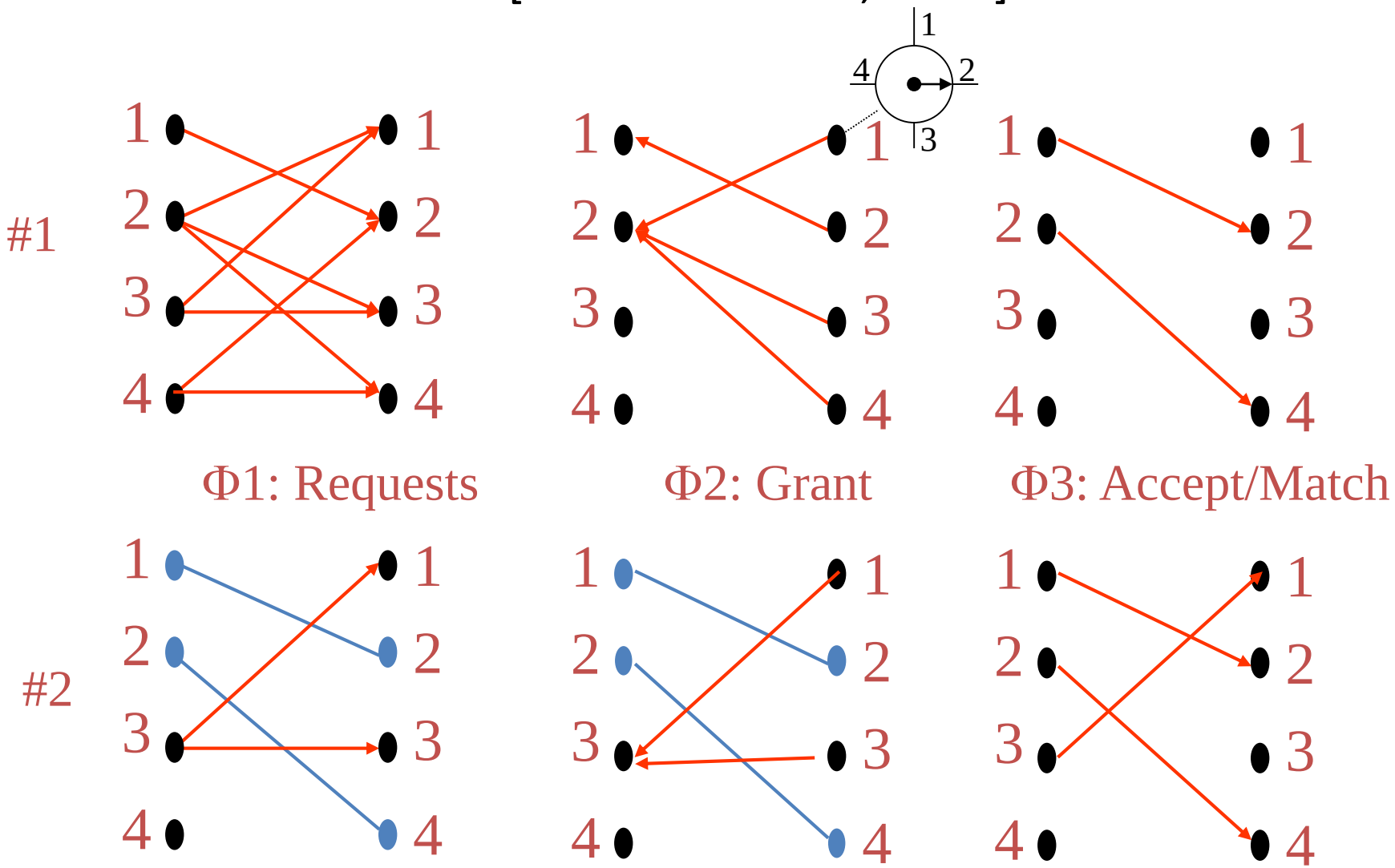


חוסר הוגנות PIM - המשך

- צריך לאפשר לכל התורים שירות הוגן ככל האפשר.
  - לתת עדיפות נמוכה למי ששורת אחרון.
- אלגוריתם PIM נותן אותה עדיפות לכל תור (כאשר מצב התורים זהה).

*i*SLIP

[McKeown *et al.*, 1993]



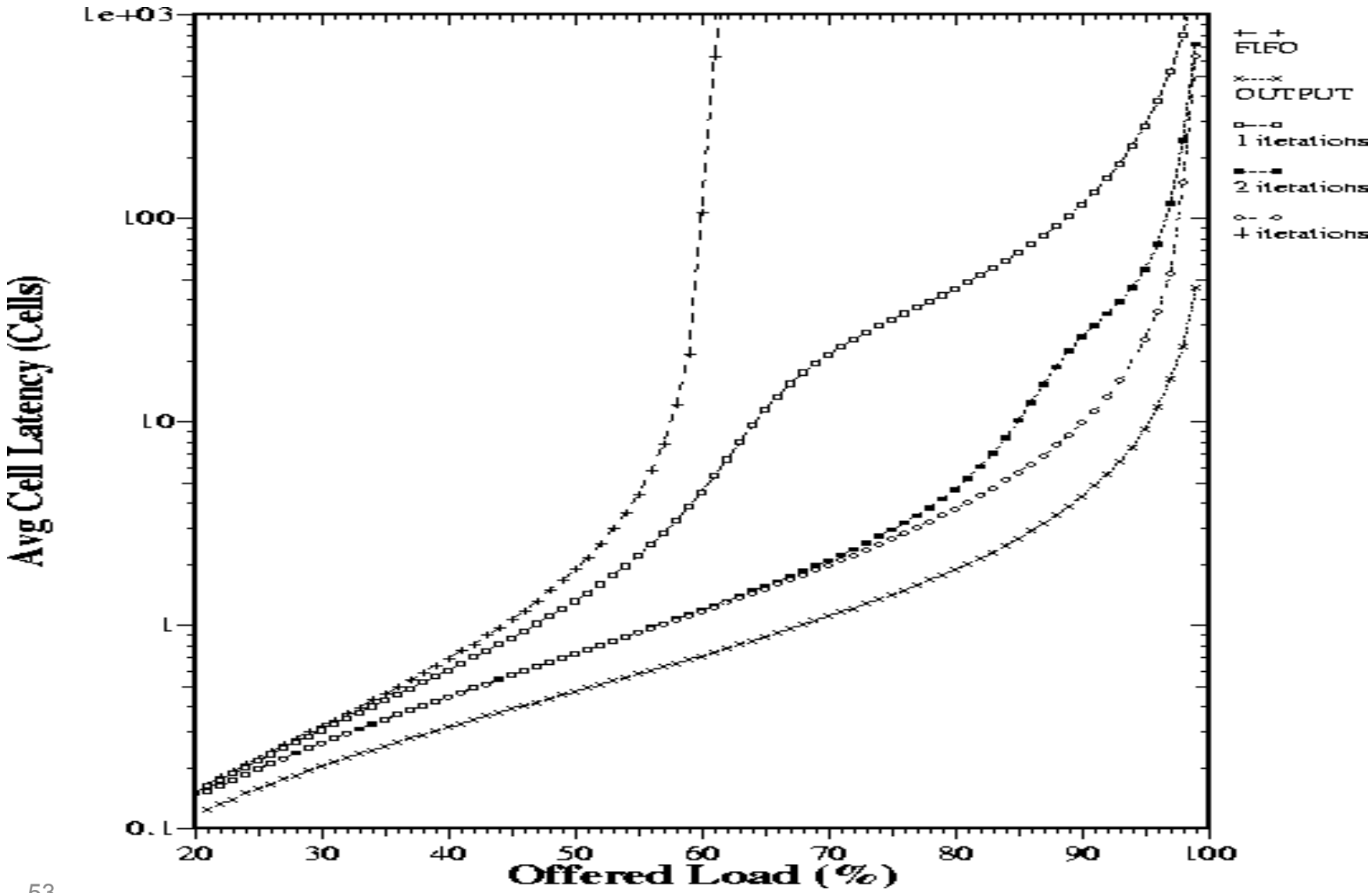
פעולת ה- iSLIP

- **שלב ה-Grant**: כל יציאה בוחרת את הכניסה כפי שמצויין במצביע שלה, או את כניסה שאחריה, בסדר של round-robin. **עדכון של המצביע מתבצע רק עם התקבל accept באיטרציה הראשונה.**
- שלב ה-Accept: כל כניסה בוחרת את היציאה על פי המצביע שלה, או יציאה שאחריה, בסדר של round-robin. **מתבצע עדכון של המצביע אם התקבל accept ורק באיטרציה הראשונה.**

תכונות ה- iSLIP

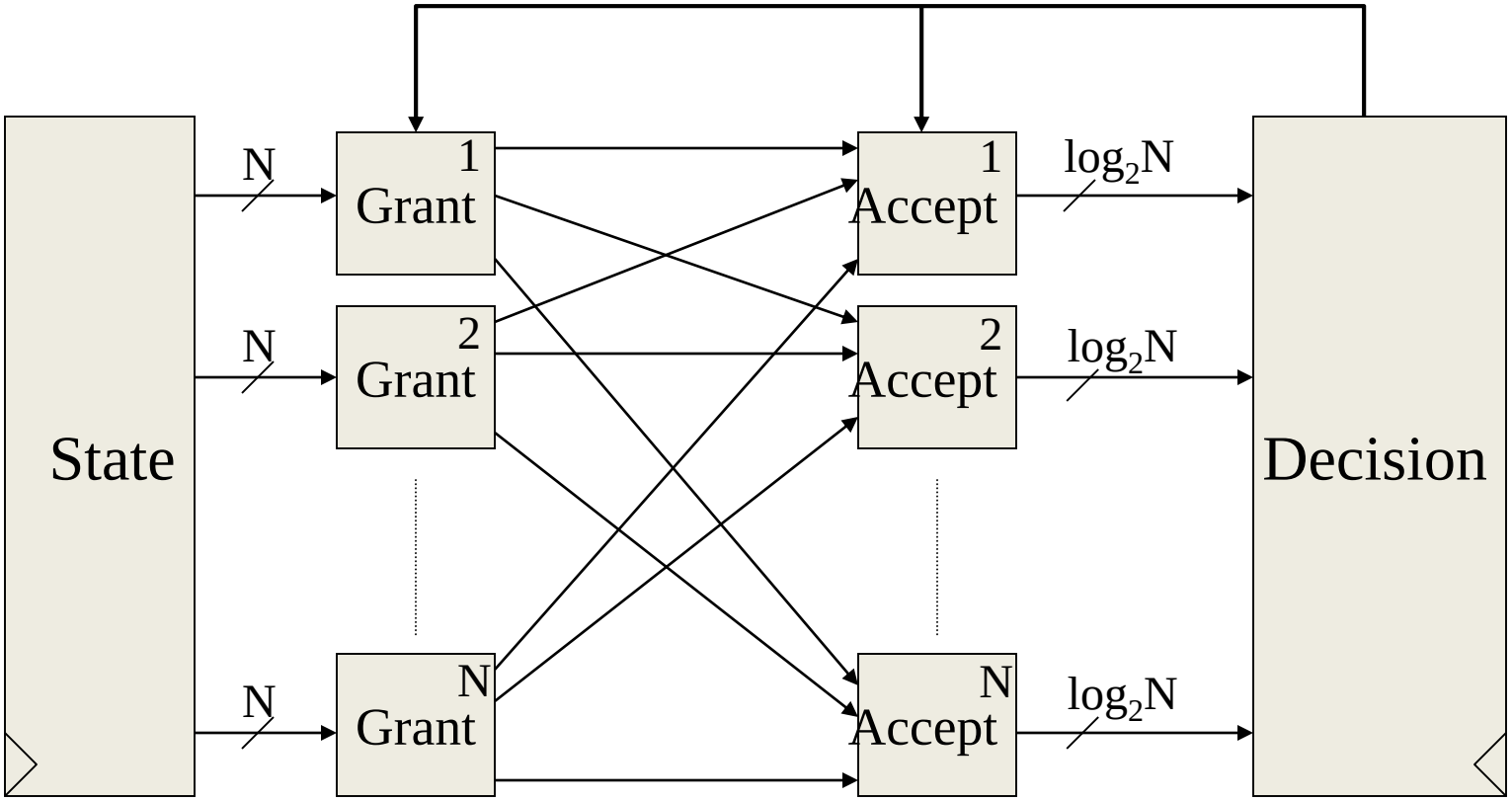
- תמיד מוצא שידוך מקסימלי.
- מתכנס בהכרח תוך N צעדים ובפועל תוך *log(N)* בממוצע.
- תחת עומסים כבדים, המצביעים נוטים להיות שונים אחד מהשני.
  - פחות איטרציות להתכנסות מאשר ה-PIM.
- בעומסים נמוכים מתנהג כמו ה-PIM.
- נותן עדיפות נמוכה למי ששורת אחרון.
  - הוגנות.

*i*SLIP


1e+03
100
10
1
0.1
20
30
40
50
60
70
80
90
100
Avg Cell Latency (Cells)
Offered Load (%)
FIFO
OUTPUT
1 iterations
2 iterations
4 iterations

*i*SLIP

Implementation



מה הלאה?

- ראינו מספר אלגוריתמים מבוססי שידוך מקסימלי.
- אלגוריתמים אלו נפוצים מאוד בתעשייה.
  - במיוחד WFA ו-iSLIP.
- בדרך כלל לא מריצים את PIM ו- *i*SLIP עד ההתכנסות.
  - התועלת של כל איטרציה פוחתת לעומת האיטרציה הקודמת.
  - לכן "מוותרים" אחרי מספר קטן של איטרציות.

מה הלאה?

- על מנת לפצות על כך, עובדים עם speedup.
- שיעור הבא נלמד על speedup > 1, ועל ההשלכות (החיוביות) הנובעות מכך.
  - הצגת נתב מבוסס תורי כניסה ויציאה (CIOQ).