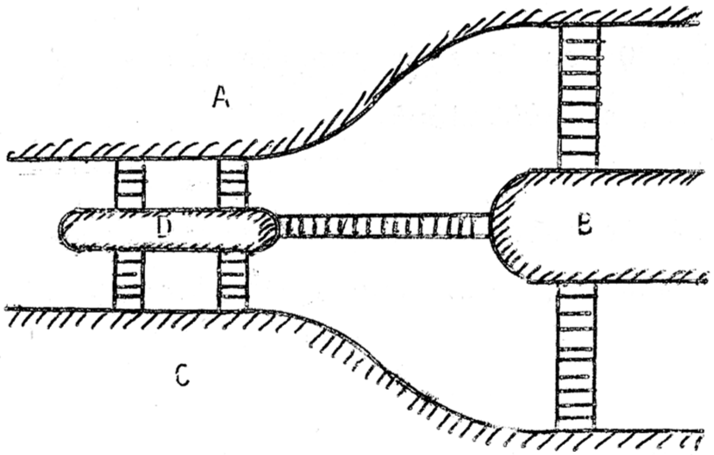
יסודות מערכות תובלה ושינוע

מצגת 7

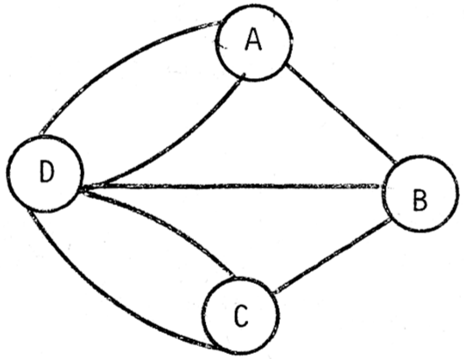
מודלים וכלים – ניתוב רכבים

מסלולי ומעגלי אויילר

- לאונרד אויילר (Euler, 1736) - מתמטיקאי ופיזיקאי שווייצרי חשוב, שבילה את רוב חייו ברוסיה ובגרמניה. תרם תרומה מכרעת לתחומים רבים ומגוונים במתמטיקה, ביניהם החשבון הדיפרנציאלי והאינטגרלי ותורת הגרפים. אויילר הגה רבים מהמינוחים ומסימני המתמטיקה המודרניים, במיוחד בתחום האנליזה מתמטית, כדוגמת סימון הפונקציה. כמו כן, הוא ידוע בזכות עבודותיו במכניקה, באופטיקה ובאסטרונומיה .



מפת קוניגסברג (קלינינגרד)



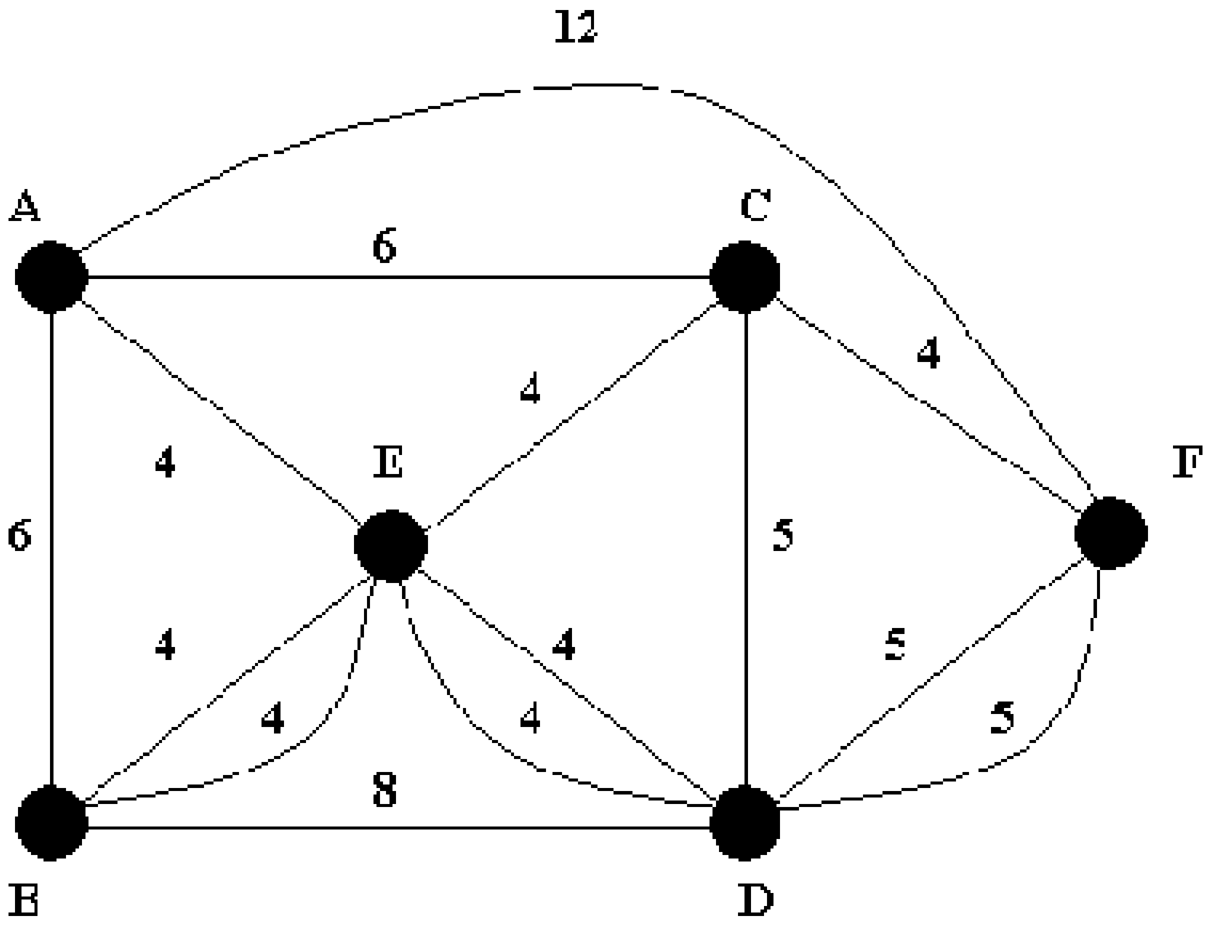
גרף אקוויולנטי למפה

מסלולי ומעגלי אויילר

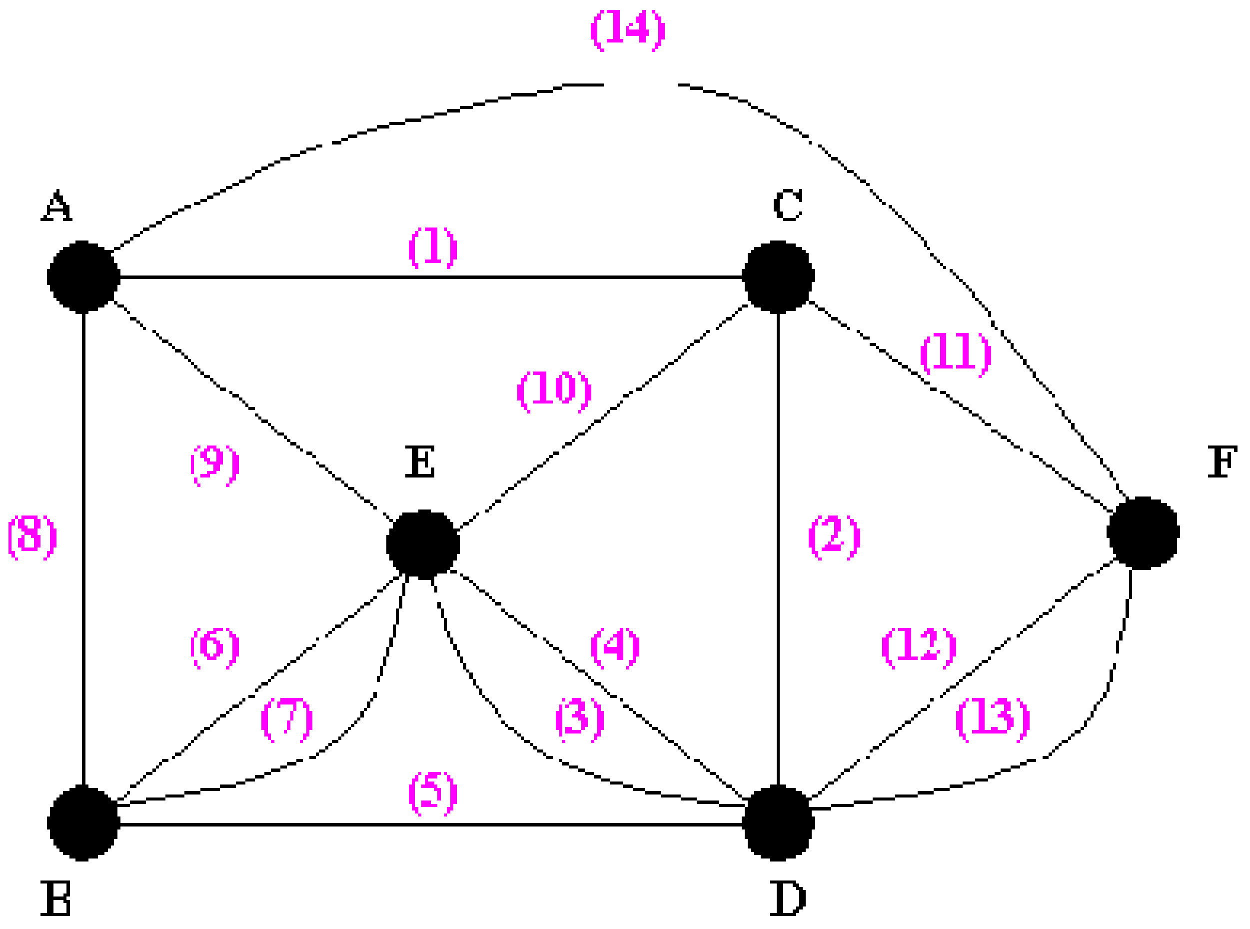
מסלול/מעגל אוילר: מסלול/מעגל בגרף בילתי מכוון, העובר בכל הקשתות של הגרף, בכל קשת בו בדיוק פעם אחת.

תנאי קיום: מסלול אוילר – בדיוק שני קודקודים בעלי דרגה אי זוגית בגרף. מעגל אוילר - כל קודקודי הגרף בעלי דרגה זוגית.

מסלולי ומעגלי אויילר – דוגמא



מסלולי ומעגלי אויילר – דוגמא



מסלול אויילר- תכנות לינארי

1. בחר קודקוד התחלה v
2. מהקודקוד בחר קשת כלשהי שאינה גשר (גשר הינה קשת שאם נסיר אותה מגרף קשיר, הגרף יהפוך ללא קשיר), אלא אם אין קשת כזו, נלך לאורכה לקודקוד השכן
3. לאחר שעברו בקשת, יש למחוק אותה מהגרף
4. חזור על שלבים 2 ו-3 כל עוד ישנן קשתות בגרף

- בכדי שייתקיים פתרון תנאי הקיום צריכים להתקיים.
- אם אנו מעוניינים במסלול קודקוד ההתחלה צריך להיות בעל דרגה אי-זוגית.

אלגוריתם למציאת מסלול אויילר Fleury's Algorithm

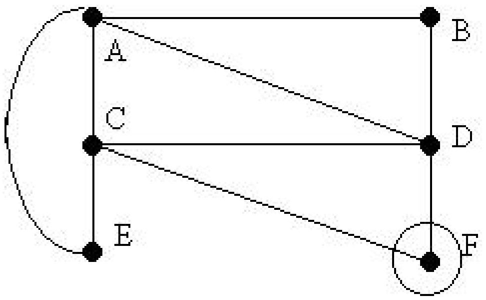
1. בחר קודקוד התחלה v
2. מהקודקוד בחר קשת כלשהי שאינה גשר (גשר הינה קשת שאם נסיר אותה מגרף קשיר, הגרף יהפוך ללא קשיר), אלא אם אין קשת כזו, נלך לאורכה לקודקוד השכן
3. לאחר שעברו בקשת, יש למחוק אותה מהגרף
4. חזור על שלבים 2 ו-3 כל עוד ישנן קשתות בגרף

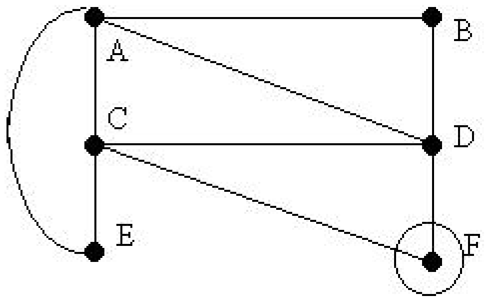
- בכדי שייתקיים פתרון תנאי הקיום צריכים להתקיים.
- אם אנו מעוניינים במסלול קודקוד ההתחלה צריך להיות בעל דרגה אי-זוגית.

דוגמא - Fleury's Algorithm

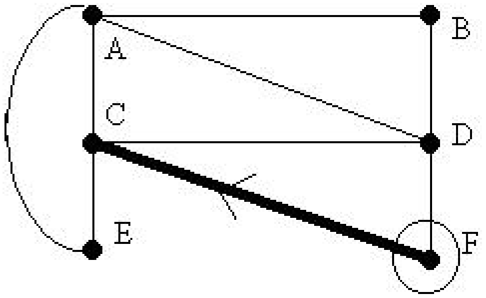
הקשתות הנותרות הרשת המקורית

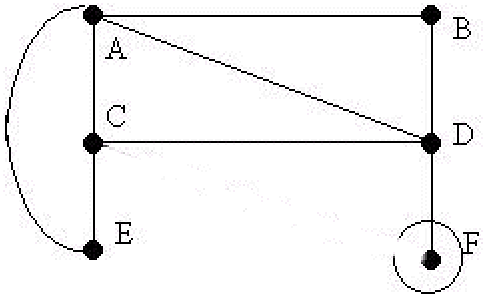
שלב 1



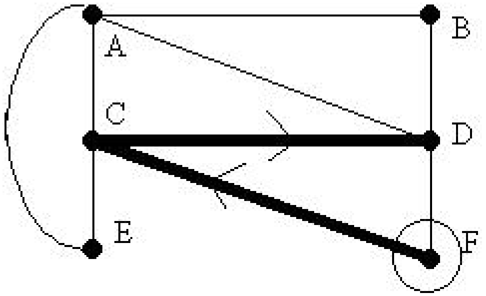


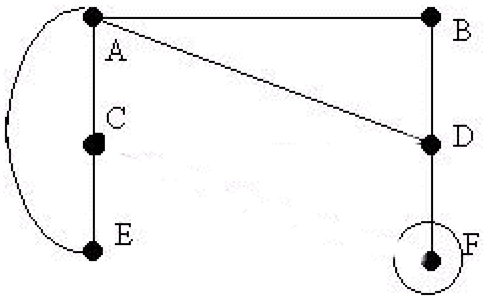
שלב 2



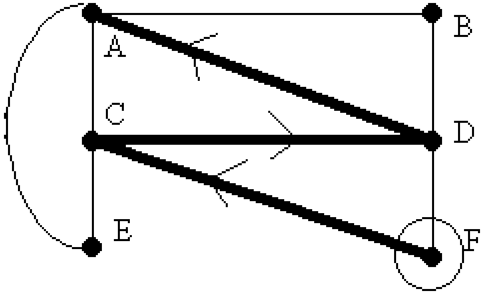
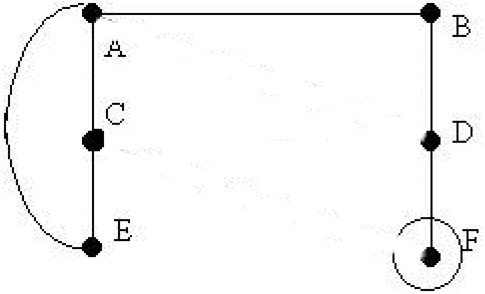
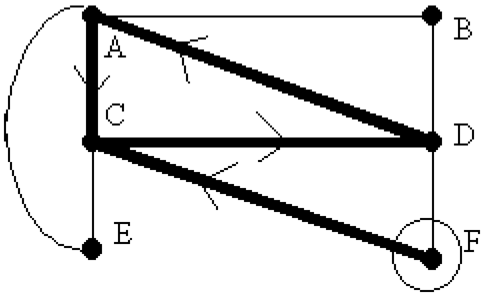
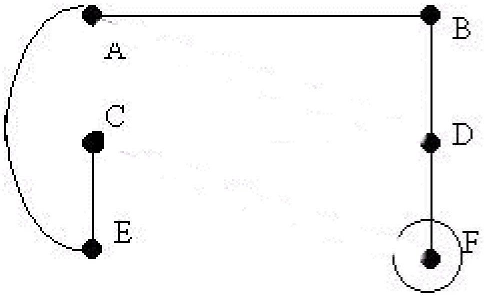


שלב 3





דוגמא - Fleury's Algorithm

	הרשת המקורית	הקשתות הנותרות
שלב 4		
שלב 5		

F-C-D-A-C-E-A-B-D-F

בעיית מחלק הדואר הסיני – Chinese Postman Problem

הבעיה: מציאת מעגל בגרף שעובר בכל קשת לפחות פעם אחת בעל משקל מינימאלי (אם קיים מעגל אוילר, זה הפתרון)

שימושים: חלוקת דואר, איסוף אשפה, תכנון מסלול הליכה במוזאון

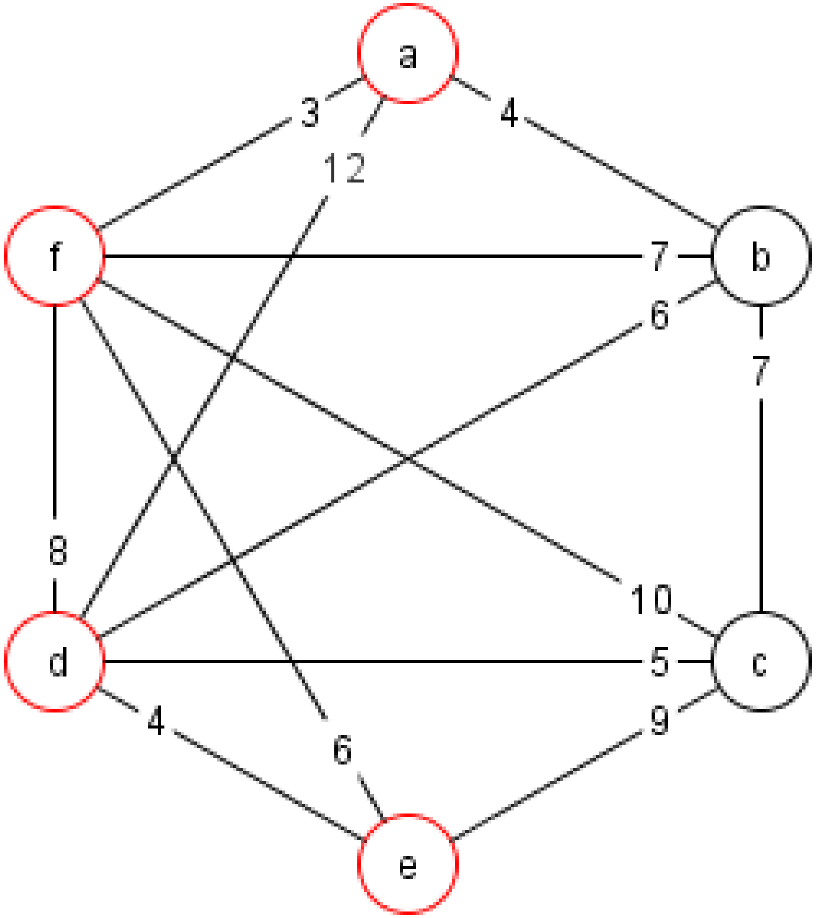
אלגוריתם לפתרון Chinese Postman Problem

1. יש למצוא ולהכין רשימה של הקודקודים בעלי הדרגה האי-זוגית.
2. על בסיס הרשימה שיצרנו בשלב 1, יש לבנות את הזיווגים האפשריים.
3. יש לחשב את המסלול הקצר ביותר בין כל זיווג שמצאנו בסעיף 2.
4. יש לבחור את קבוצת הזיווגים שסכום המסלולים הקצרים ביותר שלה הוא הנמוך ביותר. על הקשתות שבין הזיווגים הללו יש לחזור יותר מפעם אחת בפתרון בעיית מחלק הדואר הסיני.

דוגמא

הקודקודים בעלי הדרגה האי-זוגית הם:

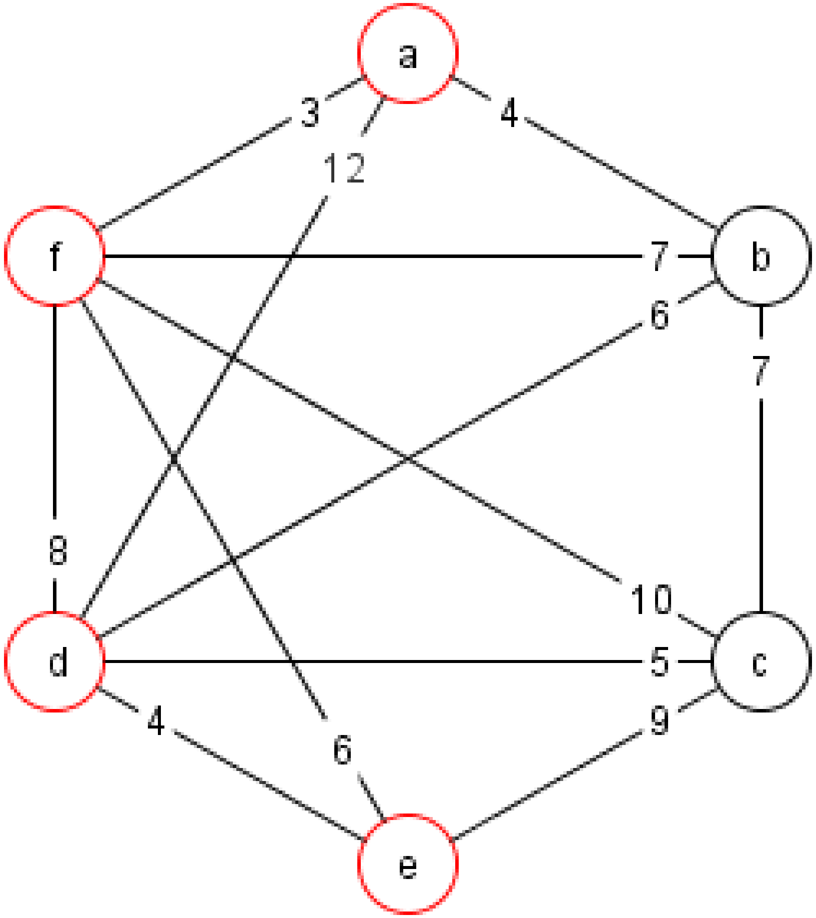
- ❑ A (3)
- ❑ D (5)
- ❑ E (3)
- ❑ F (5)



דוגמא

ניצור את כל הזוגות האפשריים ונחשב את המסלול הקצר ביותר בניהם:

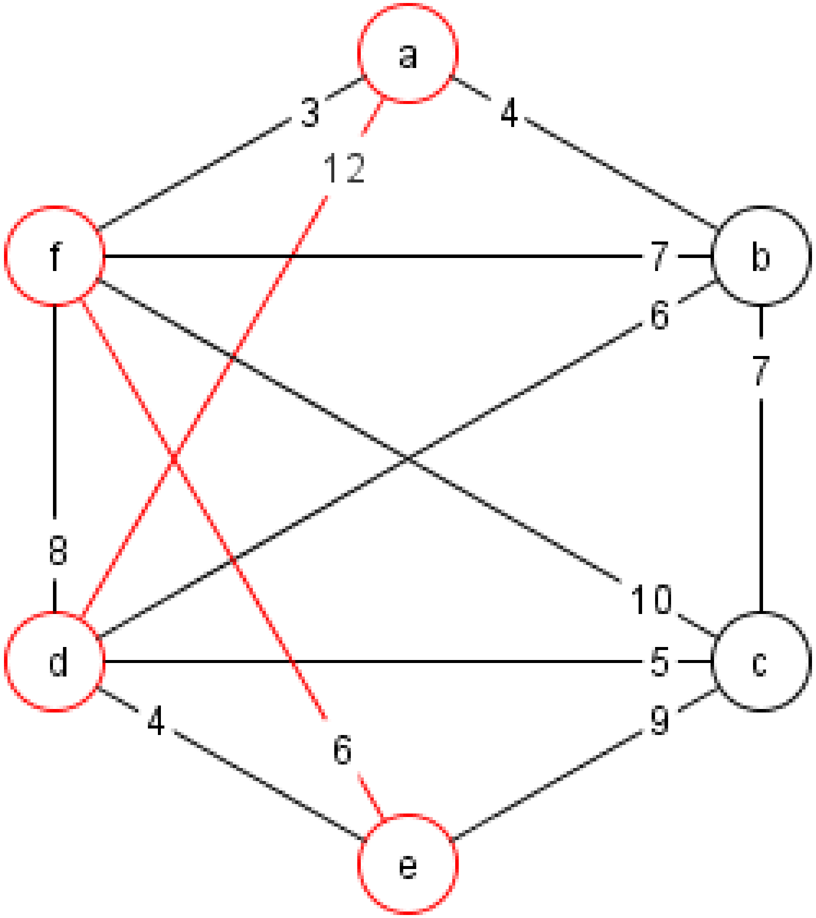
- ❑ A-D ו-E-F
- ❑ A-E ו-D-F
- ❑ A-F ו-D-E



דוגמא

ניצור את כל הזוגות האפשריים ונחשב את המסלול הקצר ביותר בניהם:

- D-A ו-F-E – 16=10+6
- E-A ו-F-D
- F-A ו-E-D

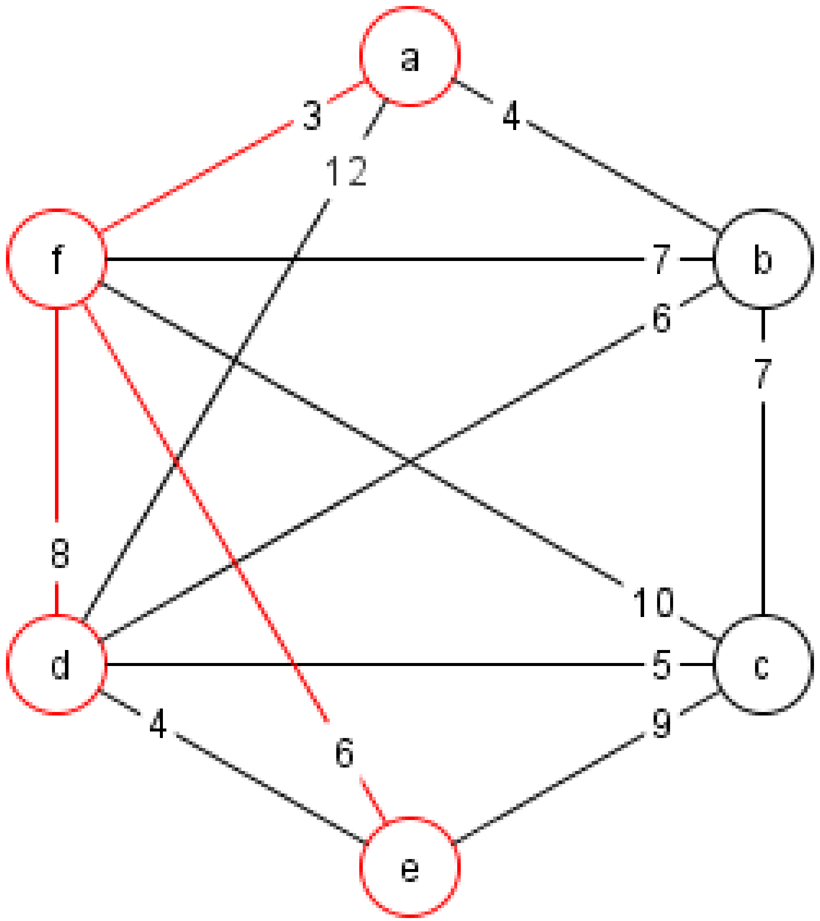


A-D = A-B-D (10)

דוגמא

ניצור את כל הזוגות האפשריים ונחשב את המסלול הקצר ביותר בניהם:

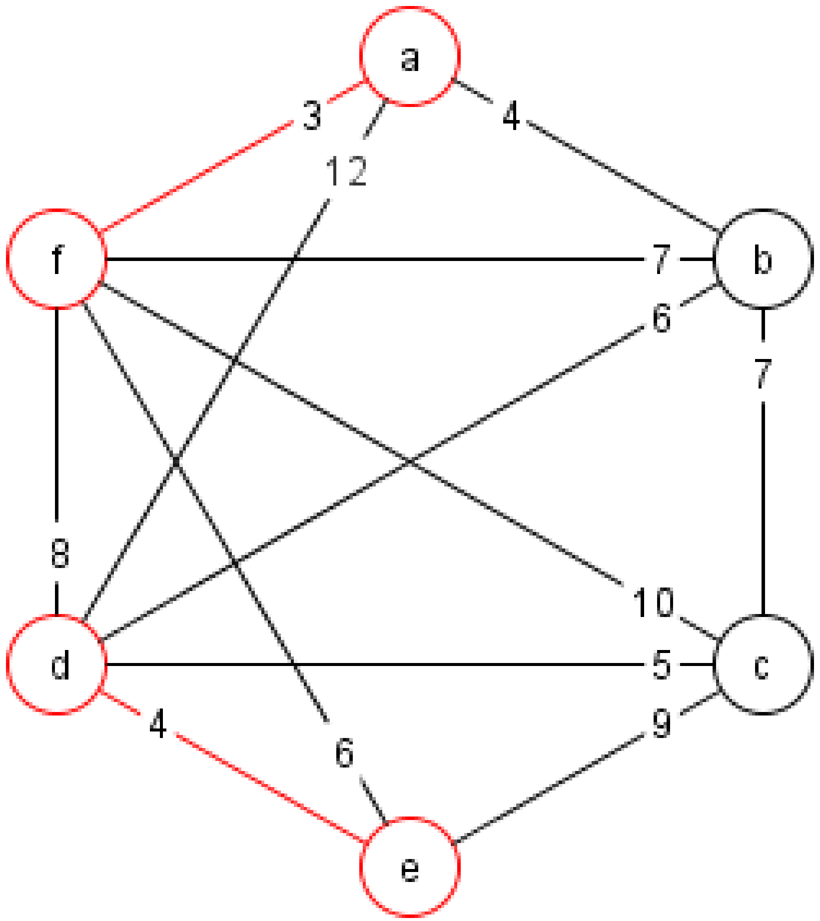
- ❑ D-A ו-F-E – 16=10+6
- ❑ E-A ו-F-D – 17=9+8
- ❑ F-A ו-E-D



דוגמא

ניצור את כל הזוגות האפשריים ונחשב את המסלול הקצר ביותר בניהם:

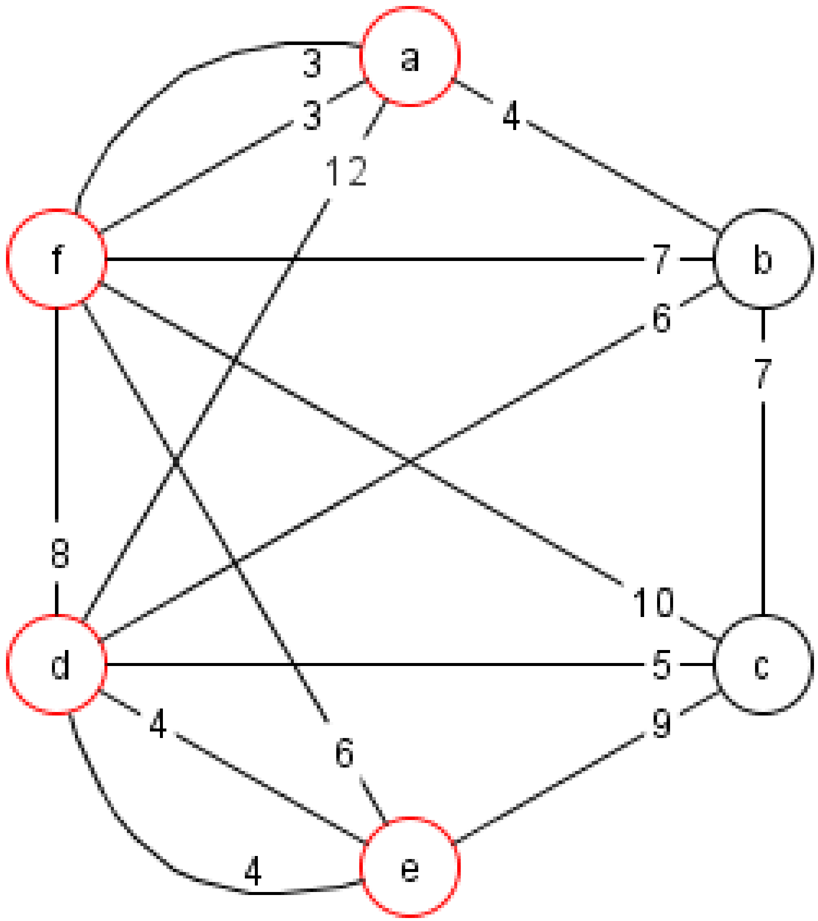
- ❑ D-A ו-F-E – 16=10+6
- ❑ E-A ו-F-D – 17=9(A-F-E)+8(F-D)
- ❑ F-A ו-E-D – 7=3+4



דוגמא

ניצור את כל הזוגות האפשריים ונחשב את המסלול הקצר ביותר בניהם:

- D-A ו-F-E – 18=12+6
- E-A ו-F-D – 17=9+8
- F-A ו-E-D – 7=3+4



כעת נפתור בעיית מציאת מעגל אויילר

בעיית הסוכן הנוסע – Traveling Salesman Problem

מסלול המילטוני: מסלול בגרף בילתי מכוון העובר דרך כל קודקודי הגרף, בכל קודקוד פעם אחת בלבד.

מעגל המילטוני: מסלול מעגלי בגרף בילתי מכוון העובר דרך כל קודקודי הגרף, בכל קודקוד פעם אחת פרט לקודקוד ממנו יצא (שבו הוא עובר בדיוק פעמיים - בהתחלה ובסוף).

בעיית סוכן נוסע: מציאת מעגל המילטוני ברשת בעל מישקל מינימאלי.

שימושים: סוכן נוסע, ביקור מהנדס שרות

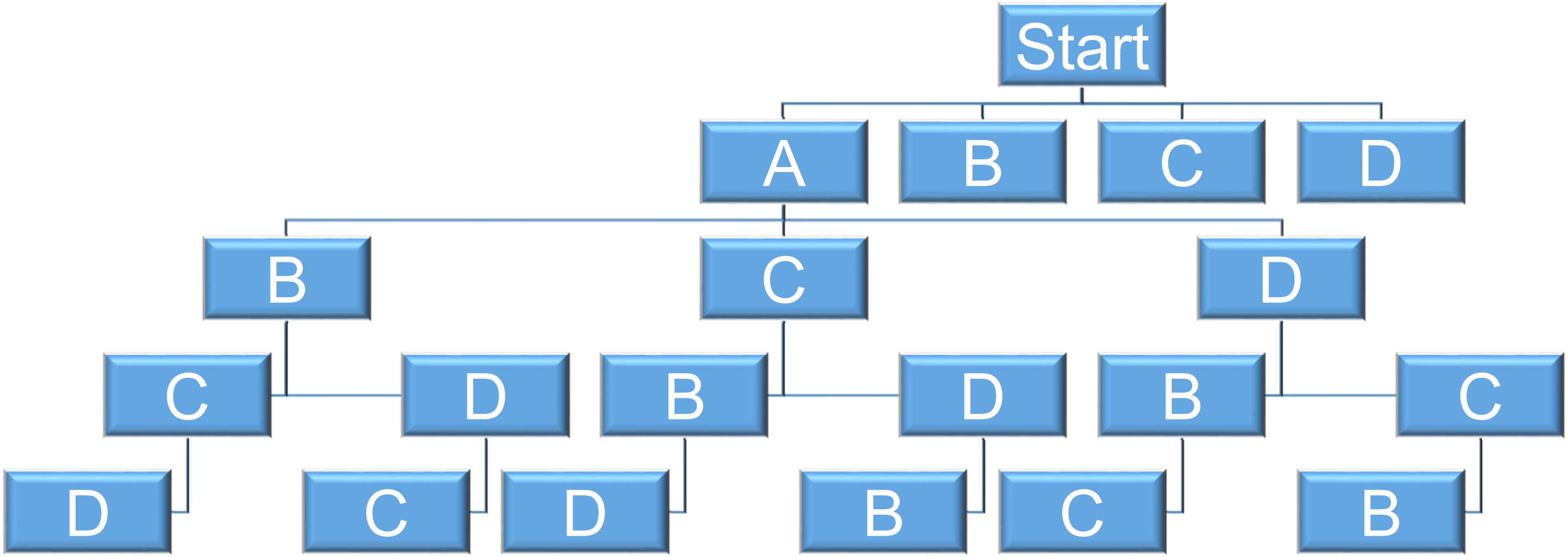
- הבעייה מוגדרת עבור גרף שלם
- סיבוכיות גבוהה מאד (NP-HARD - בילתי פתיר אופטימאלית)
- אלגוריתמים מקורבים

סוכן נוסע – אלגוריתמים מדוייקים

- הפתרון הפשוט ביותר הוא לבנות את כל הקומבינציות האפשריות, לחשב את אורכן ולבחור את זו בעלת האורך הקצר ביותר.

סוכן נוסע – אלגוריתם אלגוריתמים מדוייקים

- ❑ הפתרון הפשוט ביותר הוא לבנות את כל הקומבינציות האפשריות, לחשב את אורכן ולבחור את זו בעלת האורך הקצר ביותר.

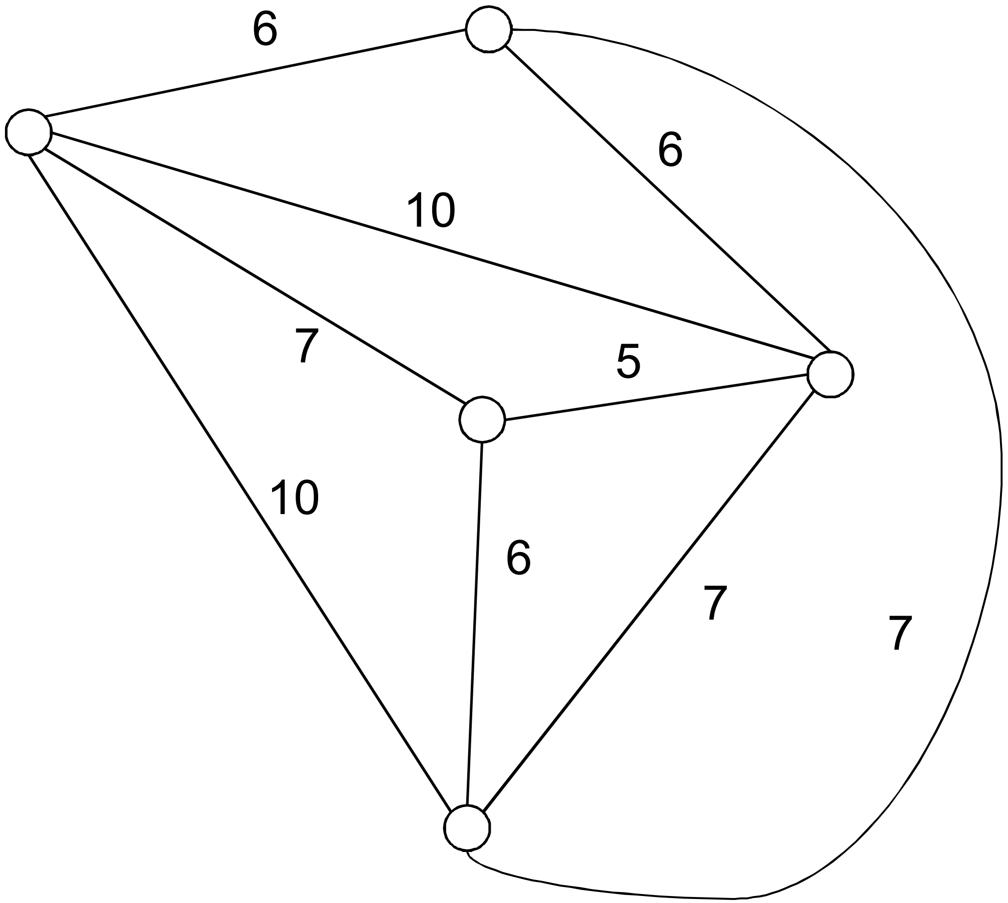


סוכן נוסע – אלגוריתם שכן הקרוב ביותר (חמדני)

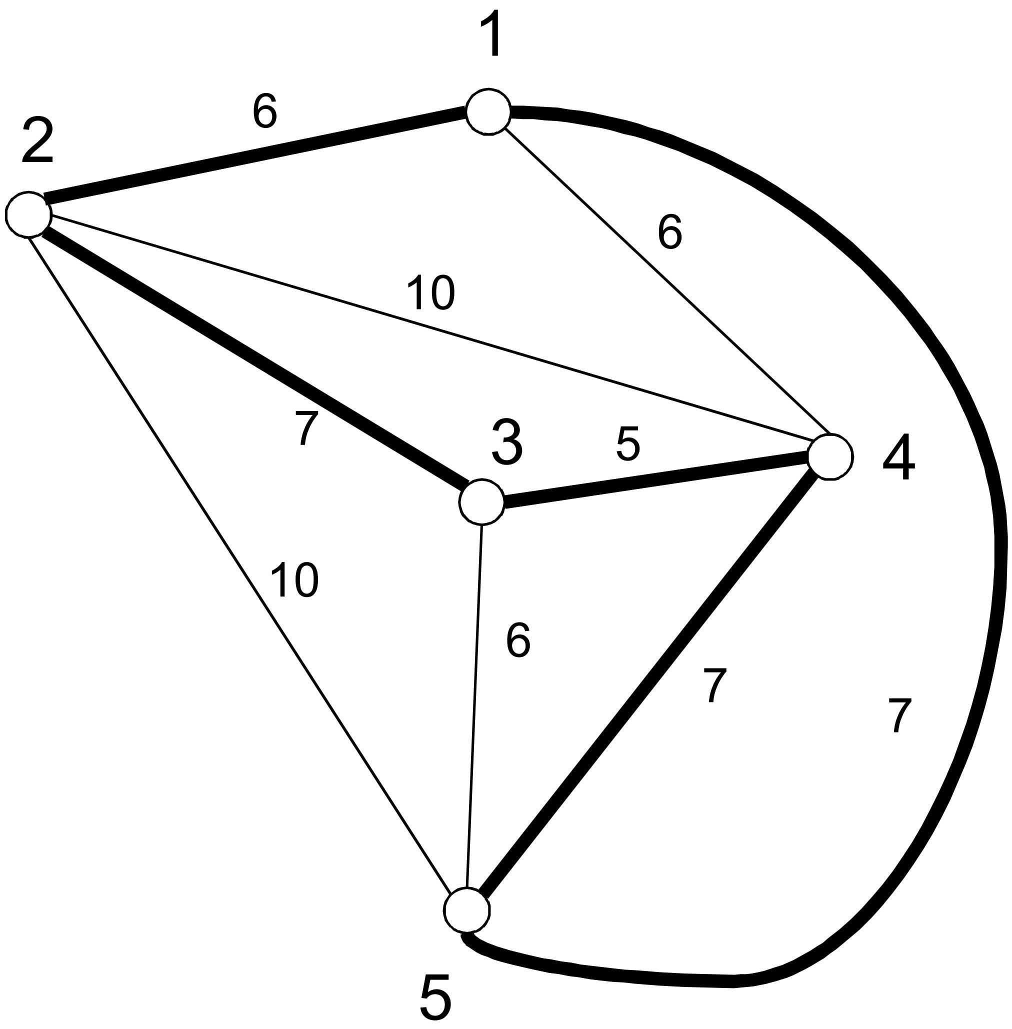
1. נבחר קודקוד באופן אקראי
2. נחפש את הקודקוד הקרוב ביותר אליו, ונחבר בניהם את הקשת עם המשקל המינימאלי
3. באופן דומה, אנו נחפש את הקודקוד הקרוב ביותר אל הקודקוד שמצאנו ב-(2), וגם בניהם נחבר את הקשת עם המשקל המינימאלי.
4. נמשיך את התהליך עבור שאר הקודקודים.

- נסמן את תוצאת האלגוריתם כ-S_{Alg}, תוצאת האלגוריתם ביחס לפתרון האופטימאלי הוא $2S_{Opt} \geq S_{Alg} \geq S_{Opt}$

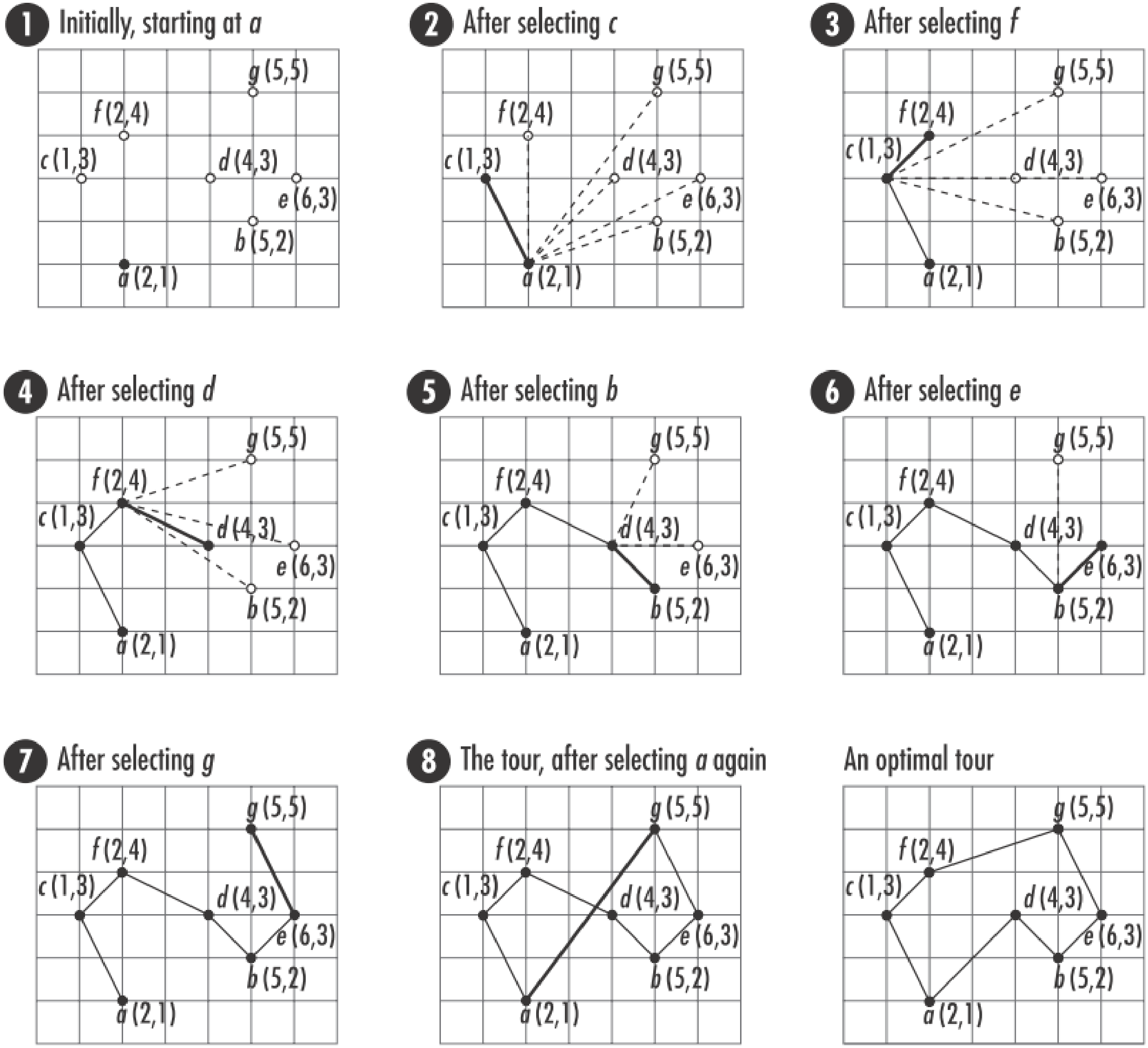
בעיית הסוכן הנוסע – דוגמא



בעיית הסוכן הנוסע – דוגמא



בעיית הסוכן הנוסע – דוגמא

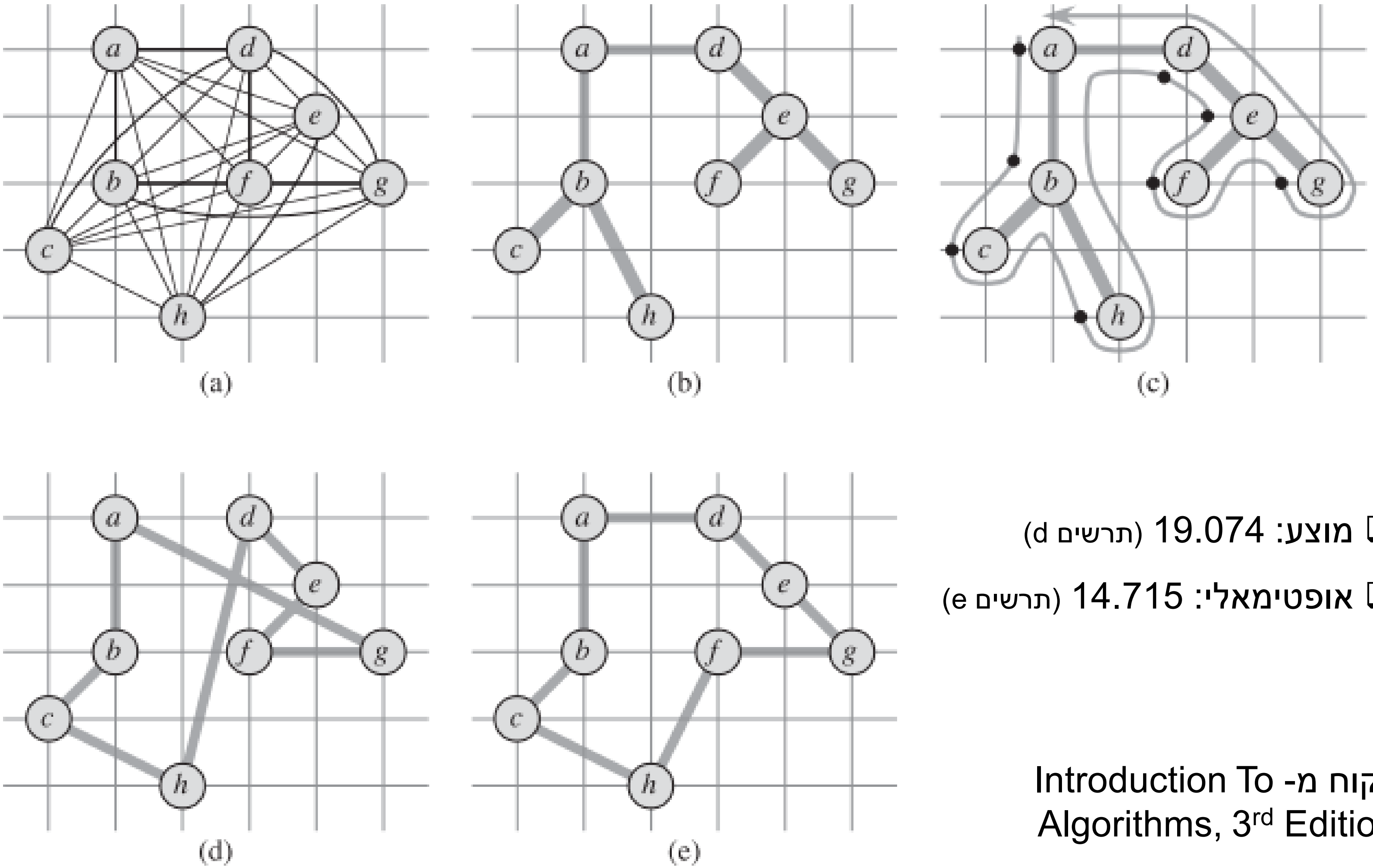


סוכן נוסע – האלגוריתם של קריסטופידס (מבוסס על עץ פורש מינימאלי)

1. נבנה עץ פורש מינימאלי של הגרף.
2. נבצע הליכה לעומק על העץ. ההליכה לעומק קובעת את סדר המעבר בין הקודקודים.
3. חוזרים לגרף המקורי, ובונים את המסלול על סמך סדר הקודקודים שנקבע בסעיף 2.

- נסמן את תוצאת האלגוריתם כ-S_{Alg}, תוצאת האלגוריתם ביחס לפתרון האופטימאלי הוא $2S_{Opt} \geq S_{Alg} \geq S_{Opt}$

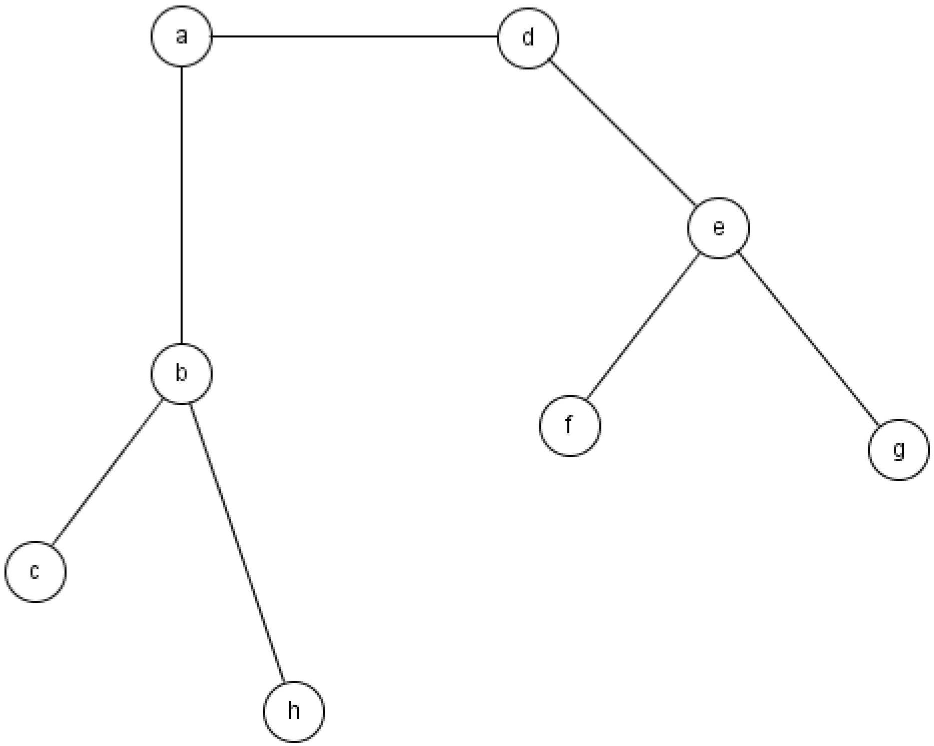
בעיית הסוכן הנוסע – דוגמא



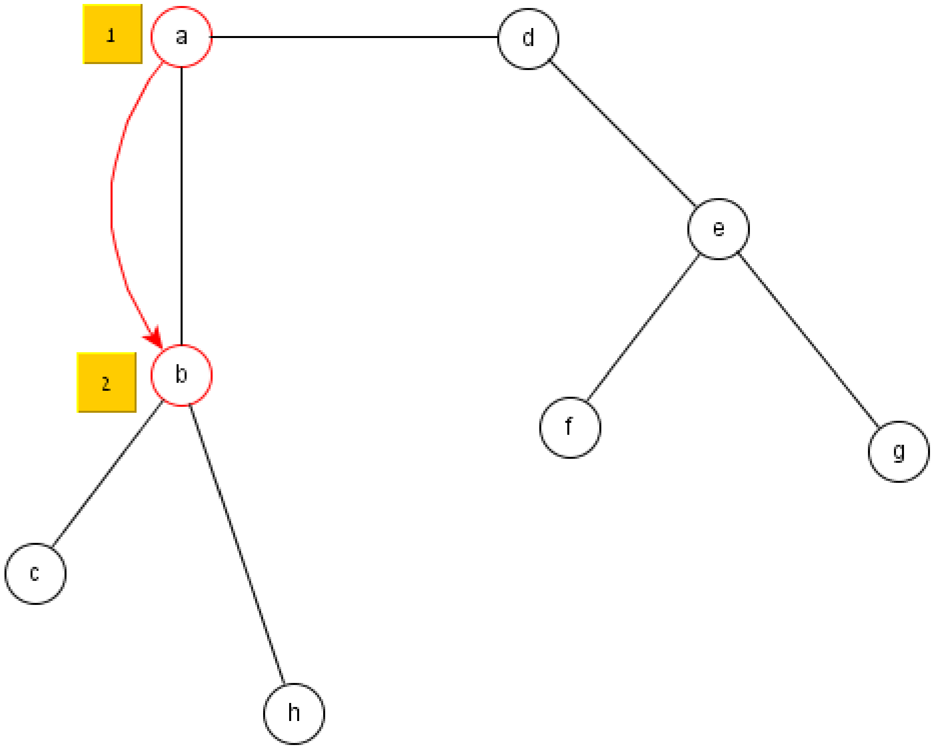
- מוצע: 19.074 (תרשים d)
- אופטימאלי: 14.715 (תרשים e)

לקוח מ- Introduction To Algorithms, 3rd Edition

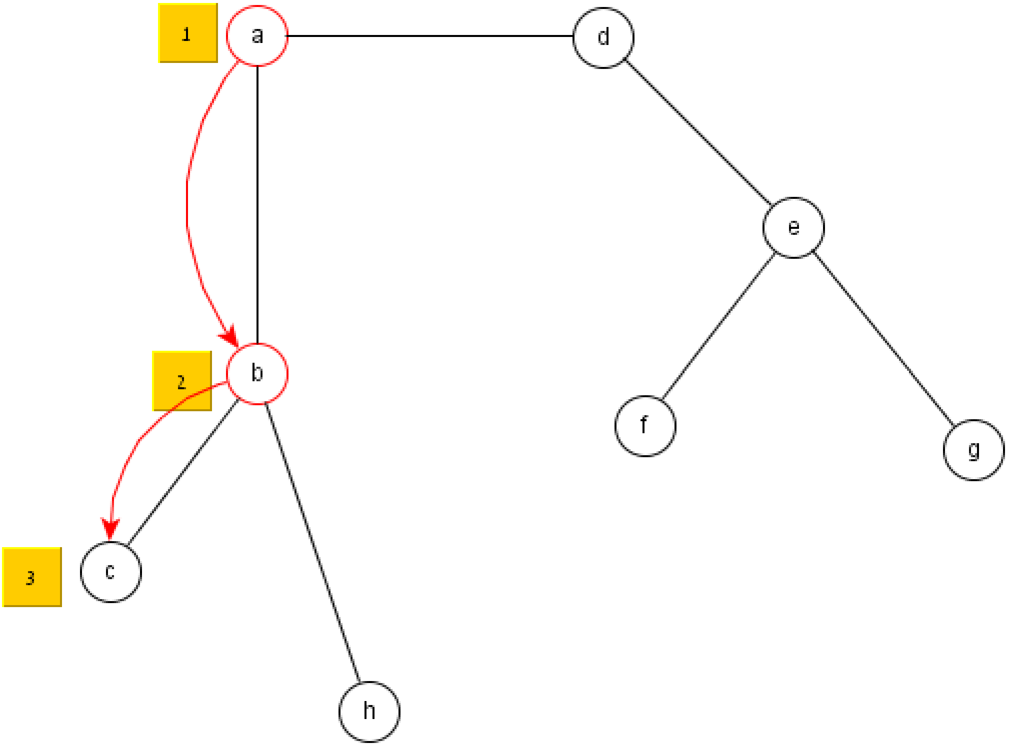
בעיית הסוכן הנוסע – דוגמא



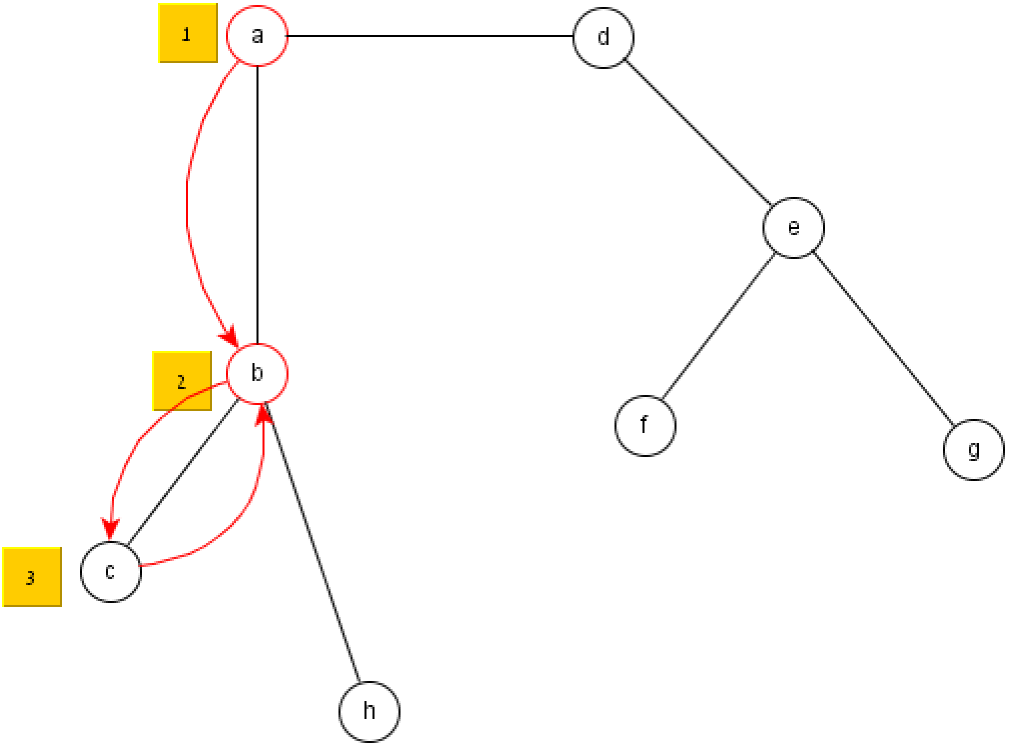
בעיית הסוכן הנוסע – דוגמא



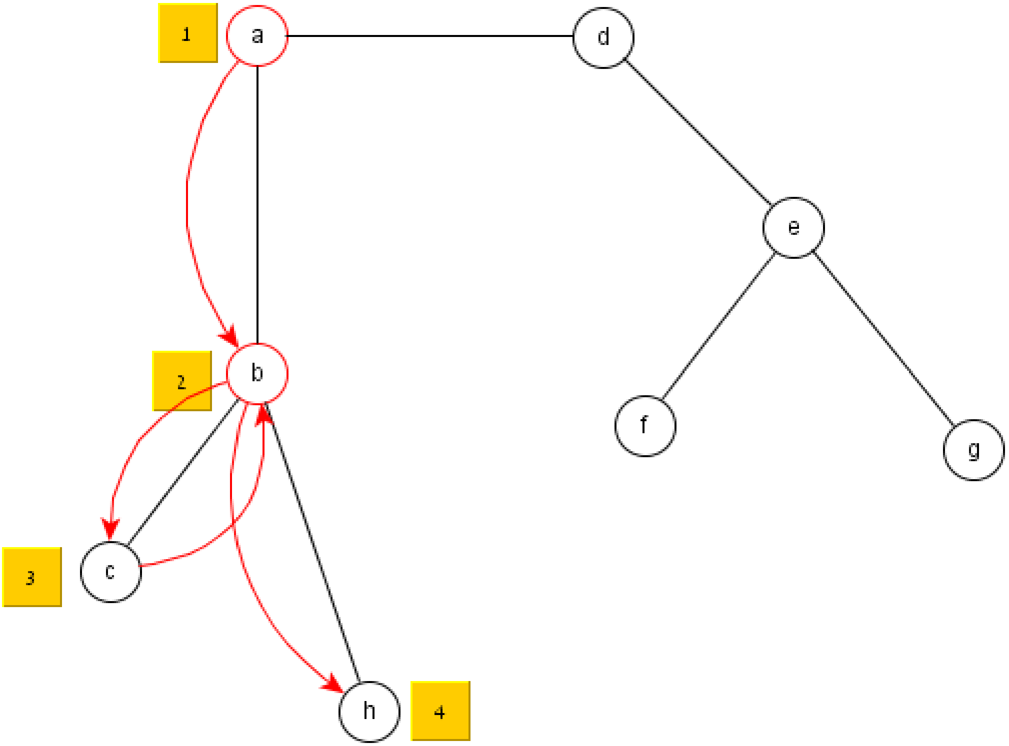
בעיית הסוכן הנוסע – דוגמא



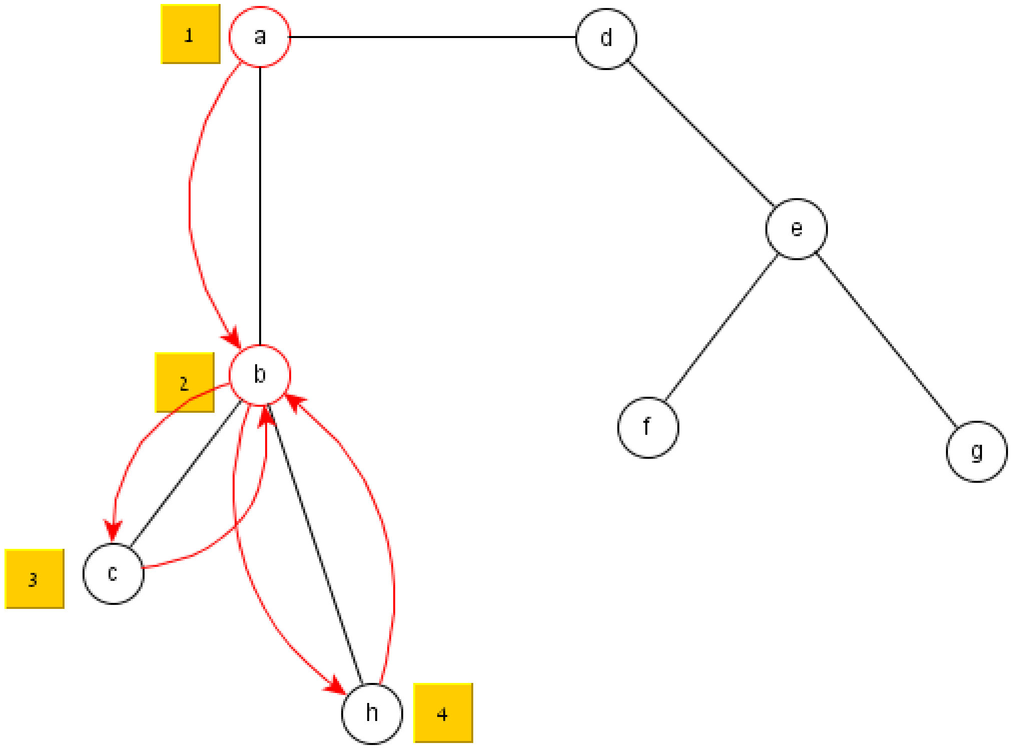
בעיית הסוכן הנוסע – דוגמא



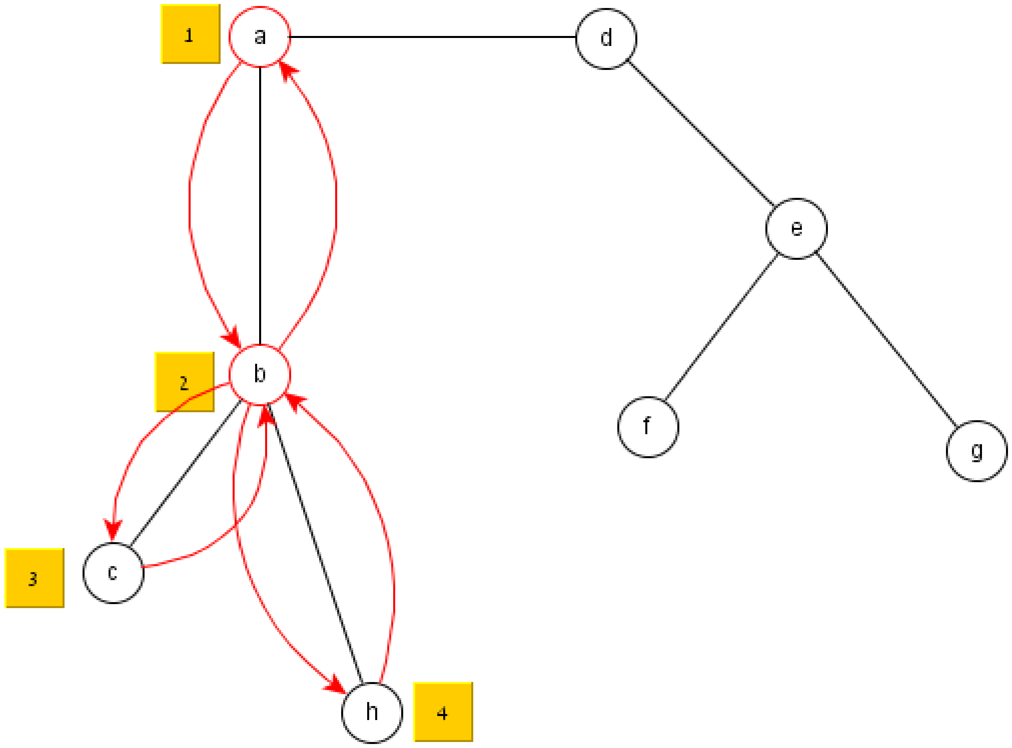
בעיית הסוכן הנוסע – דוגמא



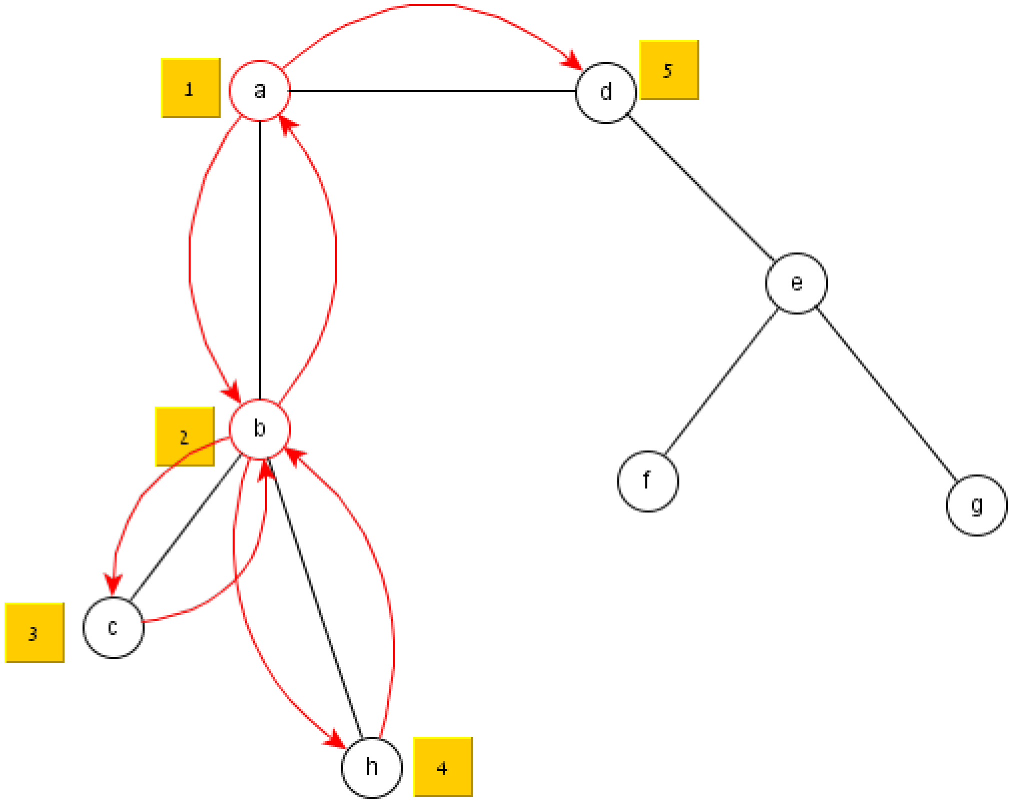
בעיית הסוכן הנוסע – דוגמא



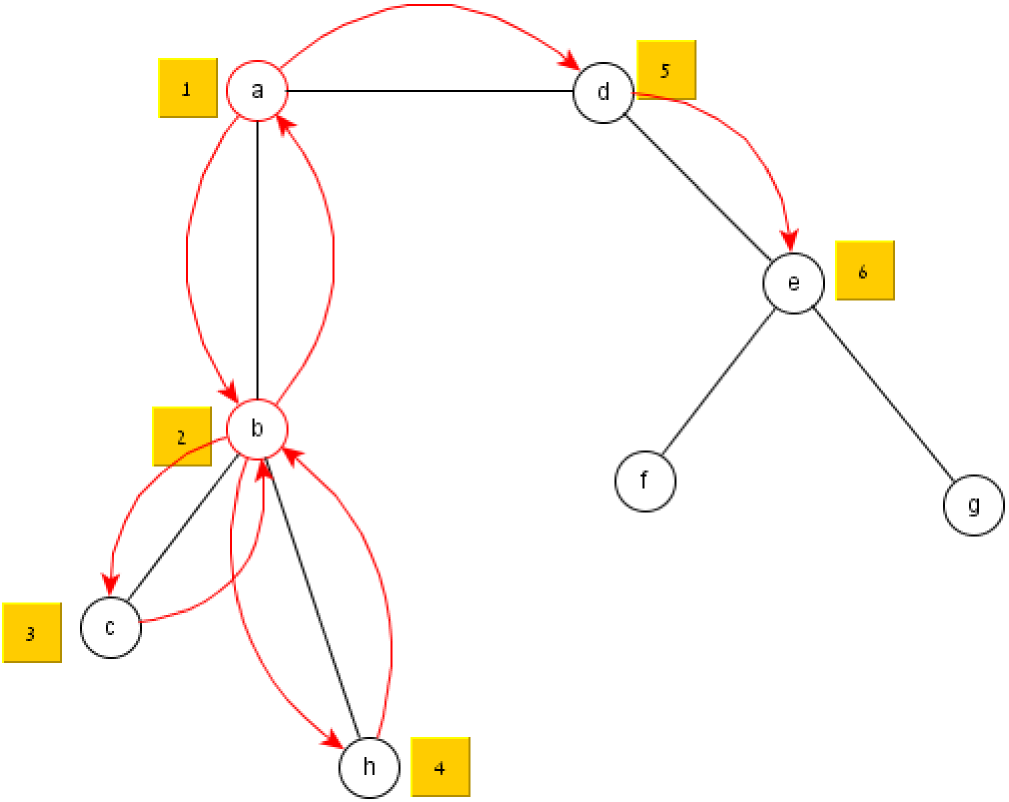
בעיית הסוכן הנוסע – דוגמא



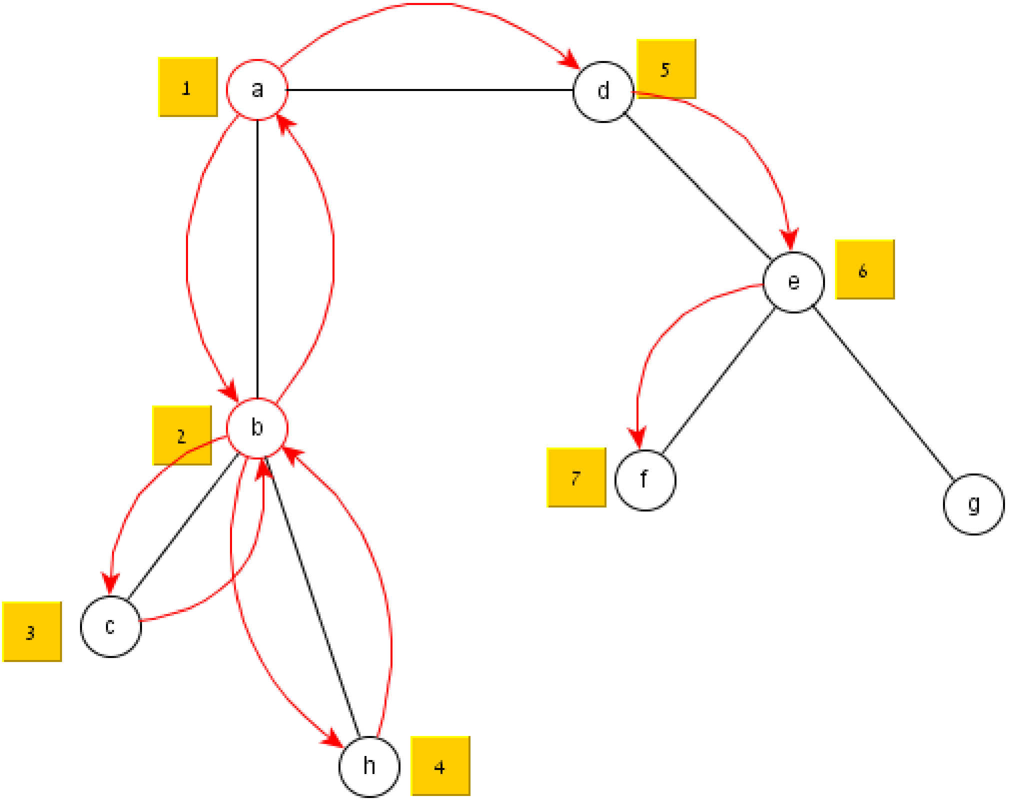
בעיית הסוכן הנוסע – דוגמא



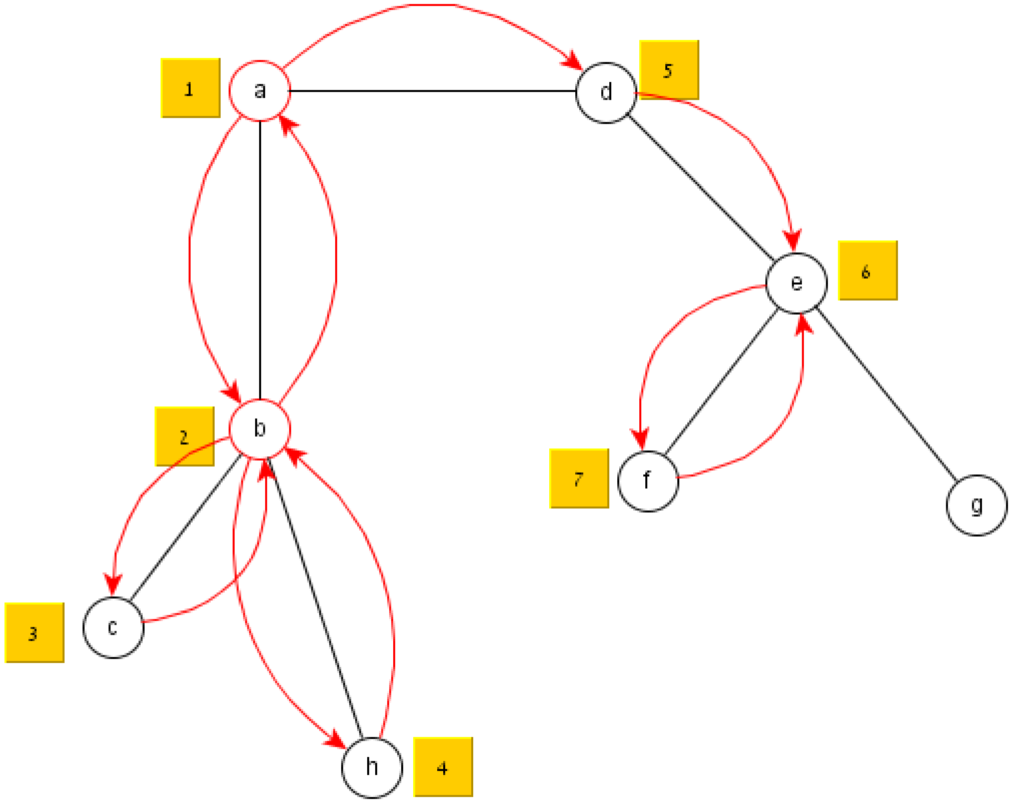
בעיית הסוכן הנוסע – דוגמא



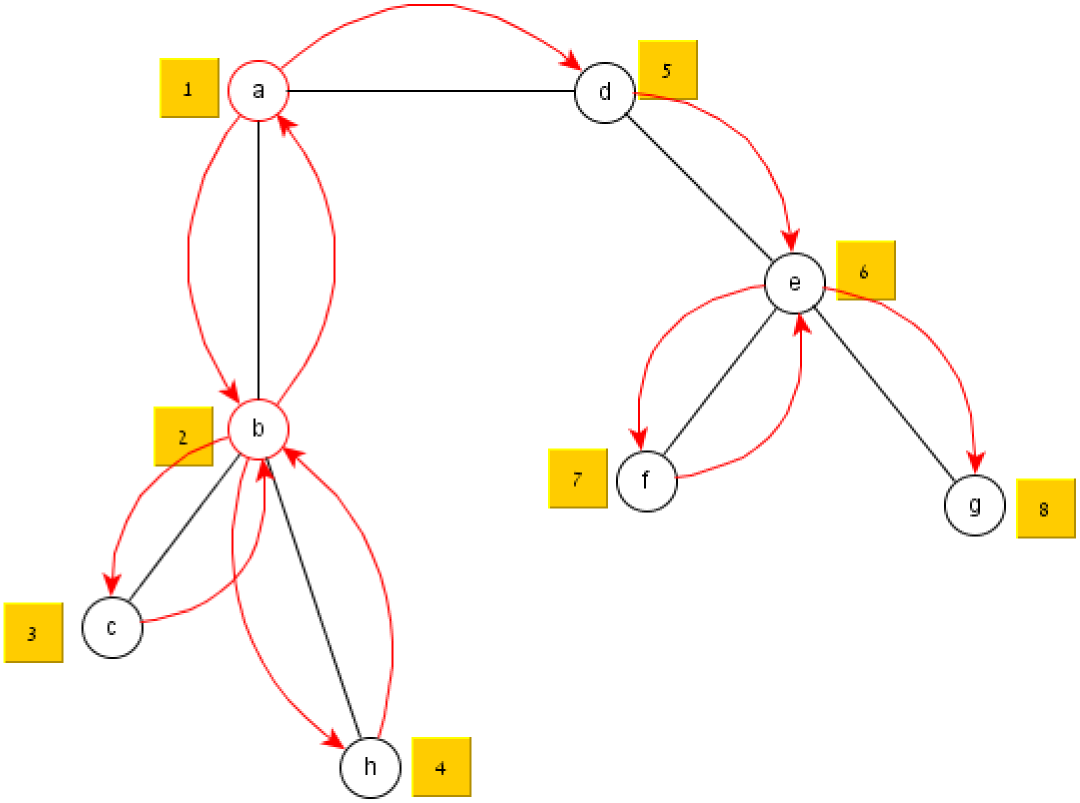
בעיית הסוכן הנוסע – דוגמא



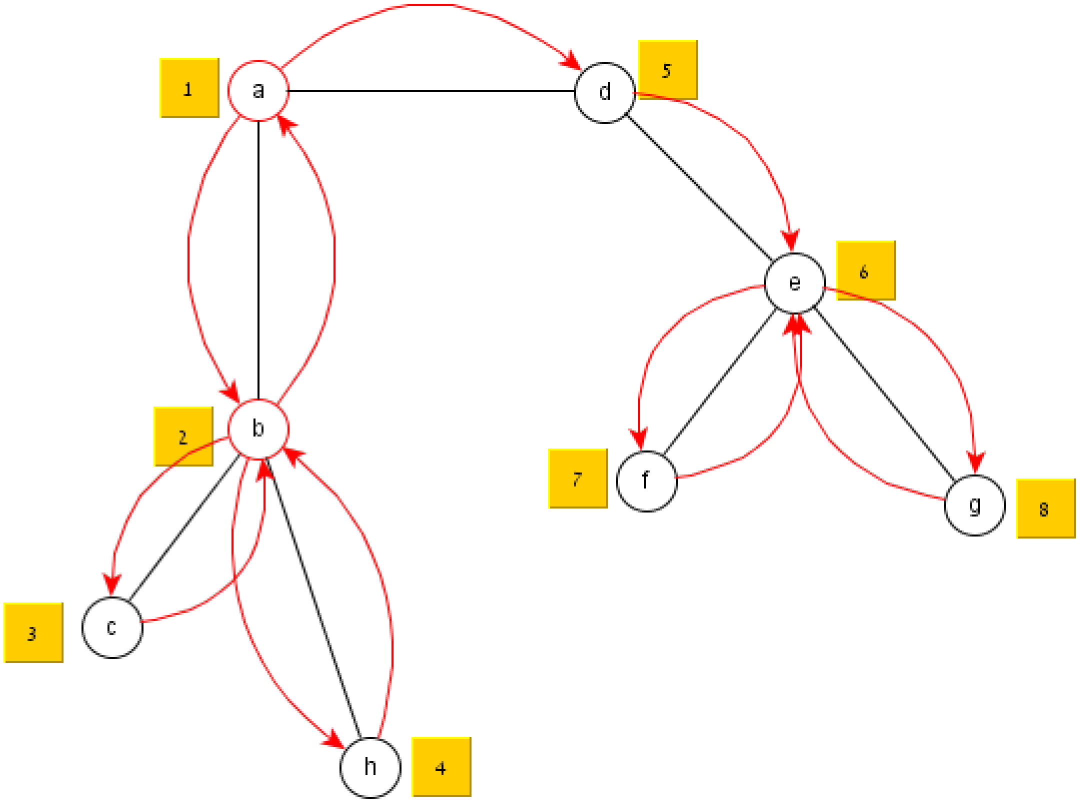
בעיית הסוכן הנוסע – דוגמא



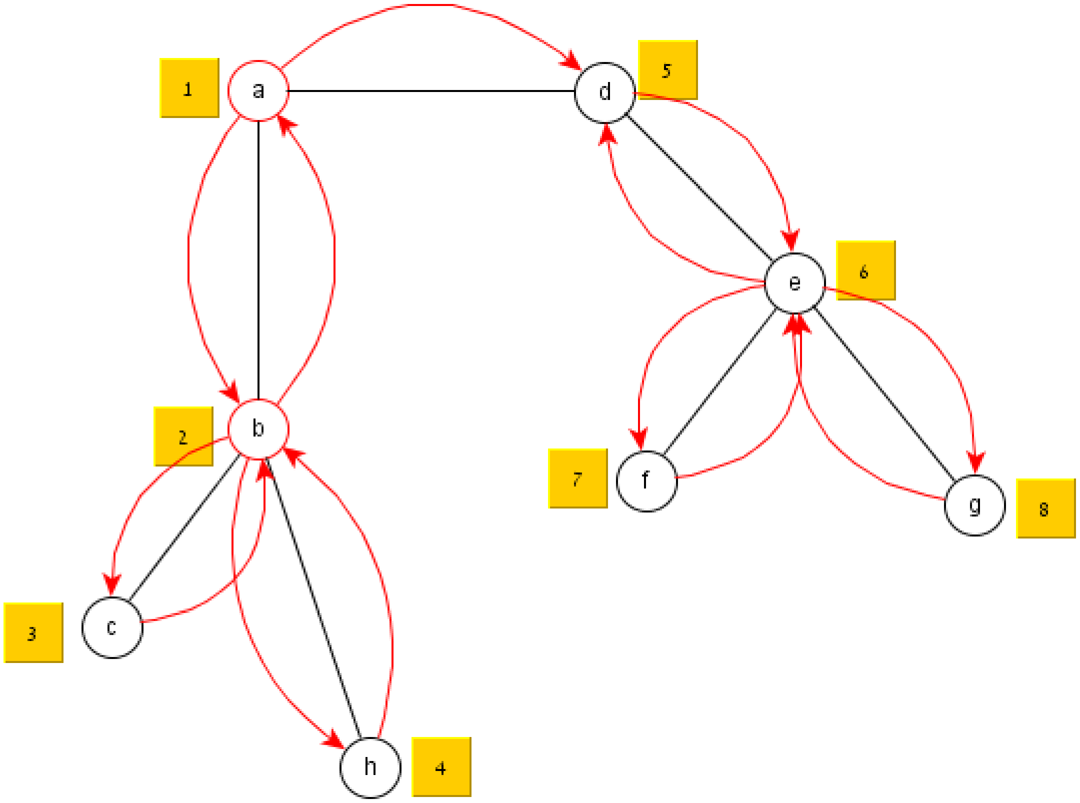
בעיית הסוכן הנוסע – דוגמא



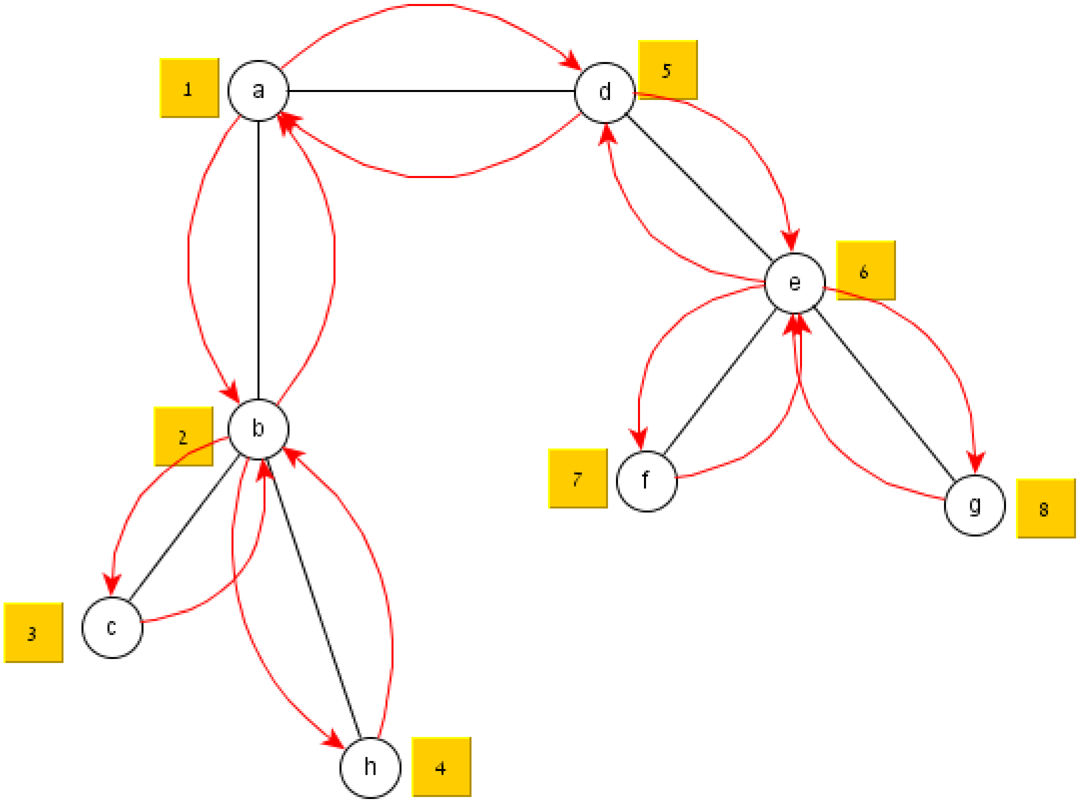
בעיית הסוכן הנוסע – דוגמא



בעיית הסוכן הנוסע – דוגמא



בעיית הסוכן הנוסע – דוגמא



סוכן נוסע – Hill Climbing

Algorithm 2.4.1: Pseudocode for Stochastic Hill Climbing.

Input: $Iter_{max}$, ProblemSize
Output: Current

```
Current ← RandomSolution(ProblemSize);
foreach iter_i ∈ Iter_max do
    Candidate ← RandomNeighbor(Current);
    if Cost(Candidate) ≥ Cost(Current) then
        Current ← Candidate;
    end
end
return Current;
```

סוכן נוסע – Simulated Annealing

Algorithm 4.2.1: Pseudocode for Simulated Annealing.

```
Input: ProblemSize, iterations_max, temp_max
Output: S_best
S_current ← CreateInitialSolution(ProblemSize);
S_best ← S_current;
for i = 1 to iterations_max do
    S_i ← CreateNeighborSolution(S_current);
    temp_curr ← CalculateTemperature(i, temp_max);
    if Cost(S_i) ≤ Cost(S_current) then
        S_current ← S_i;
        if Cost(S_i) ≤ Cost(S_best) then
            S_best ← S_i;
        end
    else if Exp( (Cost(S_current) − Cost(S_i)) / temp_curr ) > Rand() then
        S_current ← S_i;
    end
end
return S_best;
```

סוכן נוסע – TABU Search

Algorithm 2.10.1: Pseudocode for Tabu Search.

```
Input: TabuList_size
Output: S_best
S_best ← ConstructInitialSolution();
TabuList ← ∅;
while ¬ StopCondition() do
    CandidateList ← ∅;
    for S_candidate ∈ Sbest_neighborhood do
        if ¬ ContainsAnyFeatures(S_candidate, TabuList) then
            CandidateList ← S_candidate;
        end
    end
    S_candidate ← LocateBestCandidate(CandidateList);
    if Cost(S_candidate) ≤ Cost(S_best) then
        S_best ← S_candidate;
        TabuList ← FeatureDifferences(S_candidate, S_best);
        while TabuList > TabuList_size do
            DeleteFeature(TabuList);
        end
    end
end
return S_best;
```

סוכן נוסע – אלגוריתם גנטי

Algorithm 3.2.1: Pseudocode for the Genetic Algorithm.

Input: $Population_{size}$, $Problem_{size}$, $P_{crossover}$, $P_{mutation}$
Output: S_{best}

```
Population ← InitializePopulation(Population_size, Problem_size);
EvaluatePopulation(Population);
S_best ← GetBestSolution(Population);
while ¬StopCondition() do
    Parents ← SelectParents(Population, Population_size);
    Children ← ∅;
    foreach Parent_1, Parent_2 ∈ Parents do
        Child_1, Child_2 ← Crossover(Parent_1, Parent_2, P_crossover);
        Children ← Mutate(Child_1, P_mutation);
        Children ← Mutate(Child_2, P_mutation);
    end
    EvaluatePopulation(Children);
    S_best ← GetBestSolution(Children);
    Population ← Replace(Population, Children);
end
return S_best;
```

בעיית ניתוב רכבים – Vehicle Routing Problem

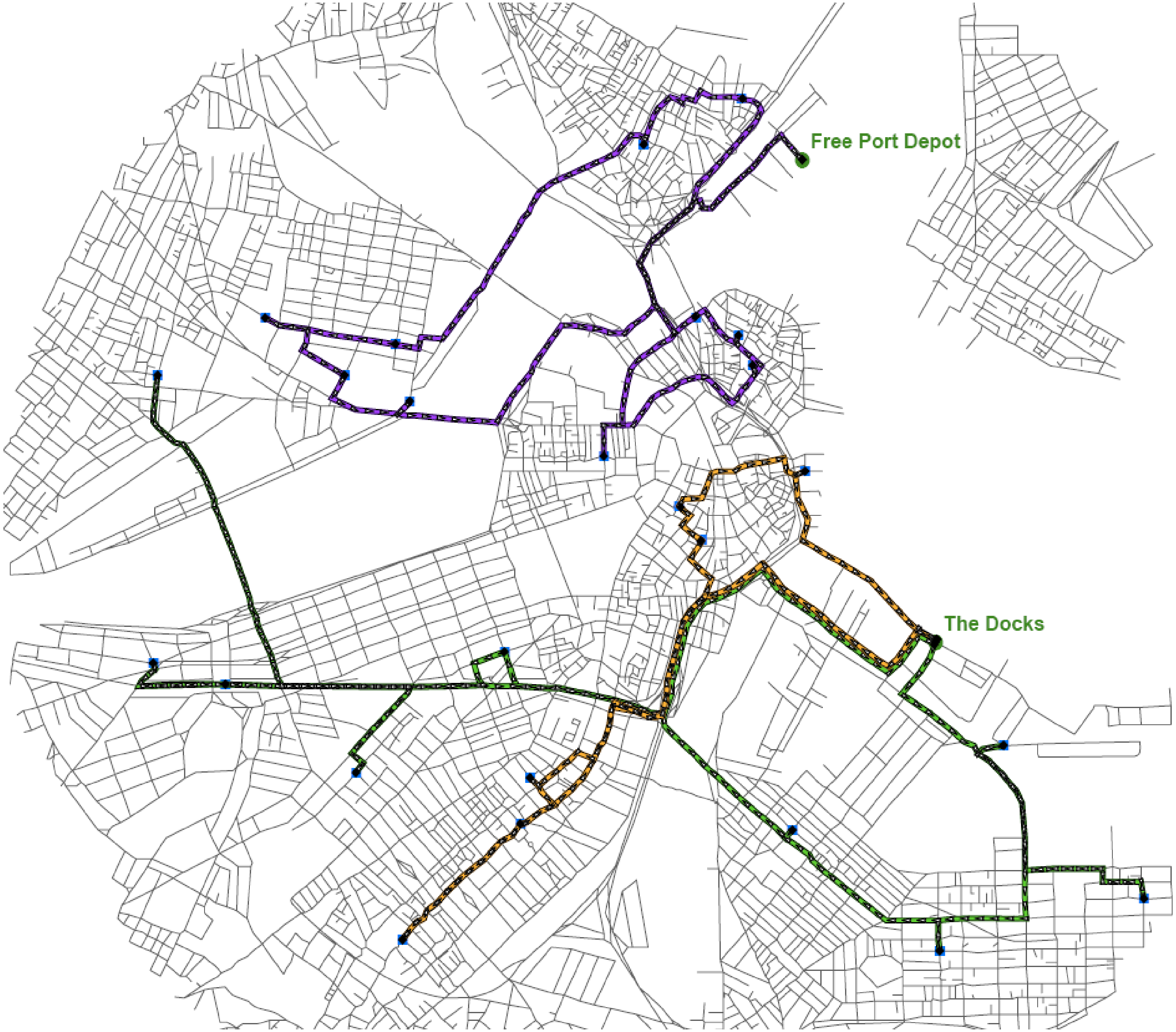
הבעיה: שיבוץ רכבים למשימות בהינתן אילוצים שונים ובמטרה להקטין עלויות / זמנים / מרחקים

שימושים: בניית רשת הפצה, שיבוץ אנשי שירות, איסוף והורדת נוסעים

- סיבוכיות גבוהה מאד – $O(n^2 3^n)$ עבור ניתוב רכב בודד
- אלגוריתמים בדרך כלל ייעודיים לבעיה, חלקם היוריסטים או מקורבים

בעיית ניתוב רכבים – דוגמא

בעיית ניתוב רכבים – דוגמא



בעיית ניתוב רכבים - ניסוח LP

$$Min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{v=1}^{NV} c_{ij}x_{ij}^{v}$$

$subject\ to$:

$$\sum_{j=1}^{n}\sum_{v=1}^{NV} x_{ij}^{v} = 1 \quad \forall i = 2,\dots,n$$

$$\sum_{i=1}^{n}\sum_{v=1}^{NV} x_{ij}^{v} = 1 \quad \forall j = 2,\dots,n$$

$$\sum_{i=1}^{n} x_{ip}^{v} - \sum_{j=1}^{n} x_{pj}^{v} = 0 \ \ \forall v = 1,\dots,NV, p = 1,\dots,n$$

$$\sum_{i=1}^{n} d_i\left(\sum_{j=1}^{n} x_{ij}^{v}\right) \le K_v \quad \forall v = 1,\dots,NV$$

$$\sum_{i=1}^{n} t_i^{v}\sum_{j=1}^{n} x_{ij}^{v} + \sum_{i=1}^{n} t_{ij}^{v}\sum_{j=1}^{n} x_{ij}^{v} \le T^{v} \quad \forall v = 1,\dots,NV$$

$$\sum_{j=2}^{n} x_{1j}^{v} \le 1 \quad \forall v = 1,\dots,NV$$

$$\sum_{i=2}^{n} x_{i1}^{v} \le 1 \quad \forall v = 1,\dots,NV$$

$$x_{ij}^{v} \in \{0,1\}$$

אלגוריתם Savings

1. בונים פתרון התחלתי. הפתרון ההתחלתי הוא אוסף מסלולים, כשכל מסלול מתחיל במחסן, מגיע ללקוח אחד בלבד וחוזר למחסן. עבור כל לקוח קיים מסלול שמגיע אליו. אין שני מסלולים שמגיעים לאותו הלקוח.
2. עבור כל זוג מסלולים אנו מחשבים את החיסכון
3. את זוג המסלולים עם החיסכון הגדול ביותר, שחיבורם אינו יגרום לכך שאחד האילוצים שלנו לא יתקיים, אנו מחברים למסלול אחד.
4. במידה ובשלב 3, אכן חיברנו זוג מסלולים, חוזרים לשלב 2, אחרת סיימנו.

אלגוריתם Swap

- פתרון הבעיה בשני שלבים
  - חלוקת הלקוחות לקבוצות
  - בניית מסלול אופטימאלי עבור כל קבוצת לקוחות (פתרון בעיית סוכן נוסע עבור כל קבוצה)

בעיית ניתוב רכבים - הרחבות

- סטטי
- דינאמי
- חלונות זמן (Time Window)
- Dial A Ride (דינאמי + חלונות זמן)
- מספר פונקציות מטרה – MOP (Multi-Objectives Programming)

בעיית ניתוב רכבים - מאפיינים

מאפיינים	אפשרויות לבעיה
גודל צי	❑ רכב בודד ❑ צי רכבים
סוג צי	❑ הומוגני ❑ הטרוגני ❑ רכבים מיוחדים
מיקום הרכבים	❑ מרכז אחד ❑ מספר מרכזים
אופי הביקוש	❑ דטרמיניסטי ❑ סטוכסטי ❑ מימוש חלקי של הביקוש
מיקום הביקוש	❑ בקשתות (בכולן או חלקן) ❑ בקודקודים (בכולם או חלקם) ❑ מעורב

בעיית ניתוב רכבים – מאפיינים (המשך)

מאפיינים	אפשרויות לבעיה
סוג הרשת	❑ מכוונת ❑ לא מכוונת ❑ מעורבת ❑ אאוקלידית
מגבלות קיבולת רכב	❑ אחיד ❑ סוגים שונים ❑ ללא מגבלה
מגבלות זמן או מרחק	❑ אחידות ❑ שונות למסלולים/רכבים/נהגים ❑ ללא מגבלה
תפעול	❑ איסוף ❑ חלוקה ❑ מעורב ❑ הספקות חלקיות

בעיית ניתוב רכבים – מאפיינים (המשך)

מאפיינים	אפשרויות לבעיה
עלויות	❑ משתנות (נסיעה) ❑ קבועות (תפעול קבוע, רכש ופחת) ❑ קנסות
פונקציות מטרה	❑ מינימום עלות ניתוב ❑ מינימום סכום עלויות כולל ❑ מינימום כלי רכב דרושים ❑ מקסימום תועלת מצד המפעיל ❑ מקסימום תועלת מצד הלקוח

בעיית ניתוב רכבים – הקשר לתחבורה ציבורית

- תכנון רשת
- קביעת תדירויות
- קביעת לוח זמנים
- שיבוץ רכבים
- שיבוץ נהגים