

יסודות מערכות תובלה ושינוע

מודלים וכלים – ניתוב רכבים

בעיית מחלק הדואר הסיני – Chinese Postman Problem

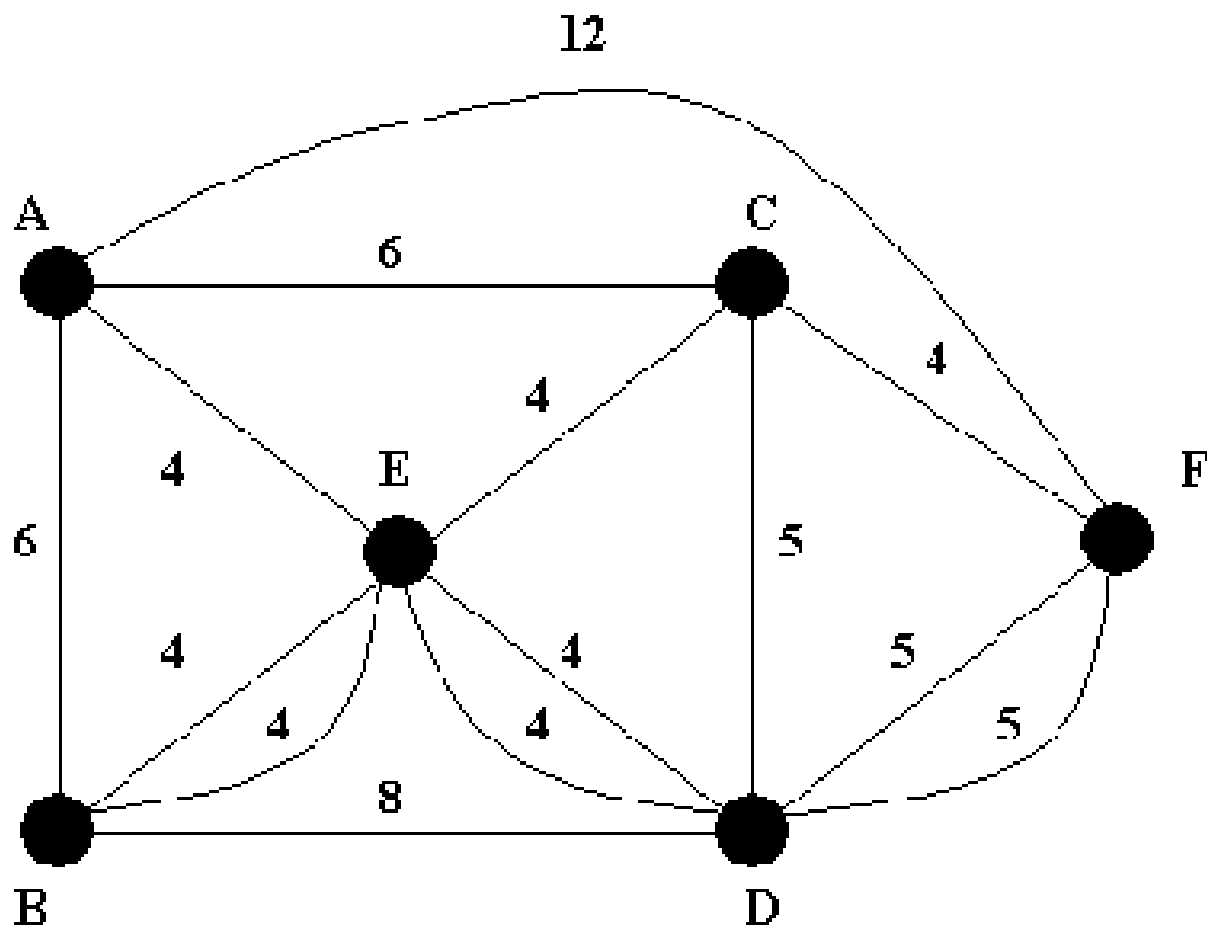
מסלול/מעגל אוילר: מסלול/מעגל בגרף בילתי מכוון, העובר בכל הקשתות של הגרף, בכל קשת בו בדיוק פעם אחת.

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

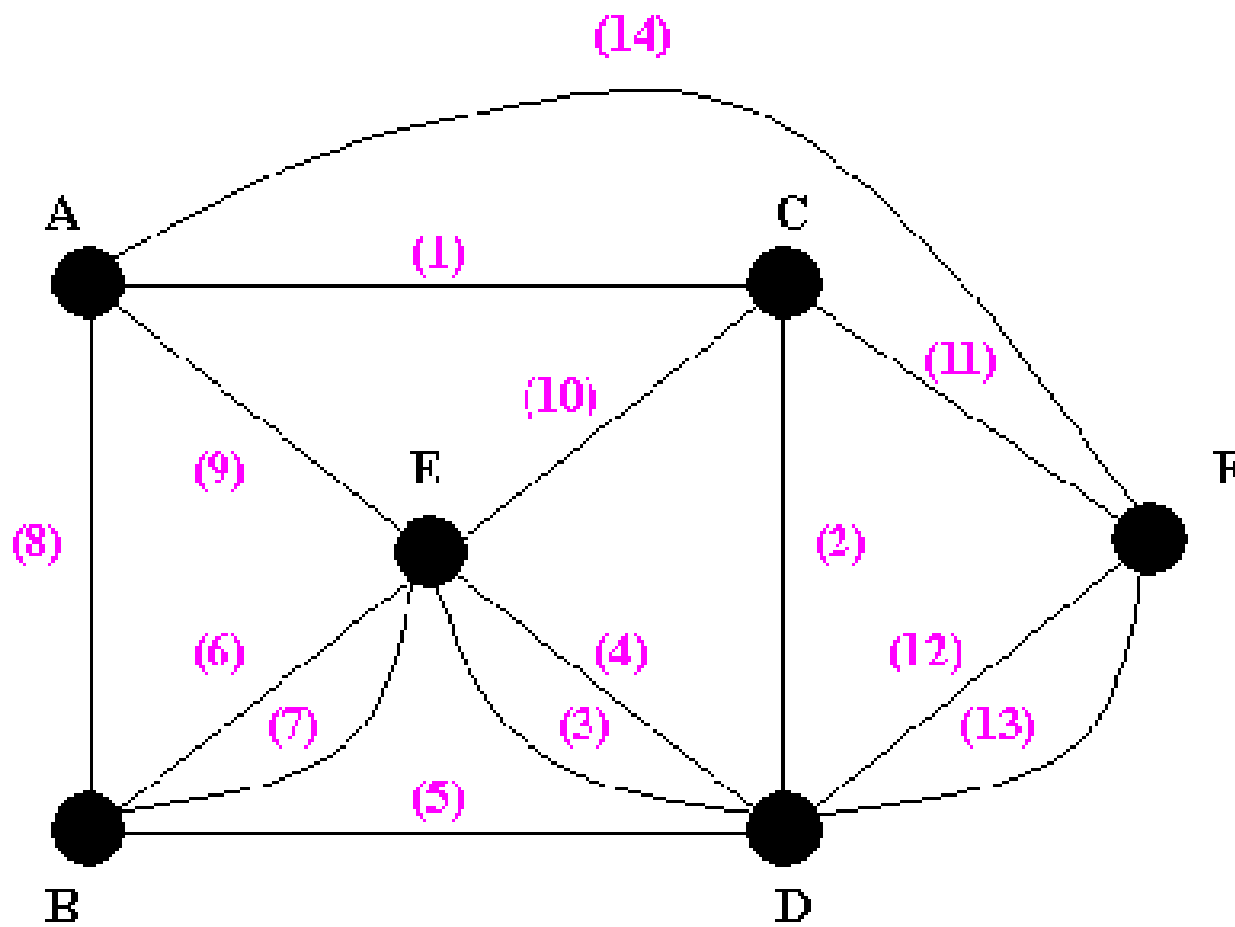
הבעיה: מציאת מעגל בגרף שעובר בכל קשת לפחות פעם אחת בעל מישקל מינימאלי (אם קיים מעגל אוילר, זה הפתרון)

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

בעיית מחלק הדואר הסיני – דוגמא



בעיית מחלק הדואר הסיני – דוגמא



Fleury's Algorithm

אלגוריתם למציאת מסלול אויילר

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, ולך 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 לפתרון Chinese Postman Problem

1. יש למצוא את כל הקודקודים בעלי הדרגה האי-זוגית
2. יש לייצר רשימה של כל הזוגות של הקודקודים שנמצאו בשלב 1
3. על בסיס הרשימה שיצרנו בשלב 2, יש לבנות את הזיווגים האפשריים.
4. יש לחשב את המסלול הקצר ביותר בין כל זיווג שמצאנו בסעיף 3.
5. 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

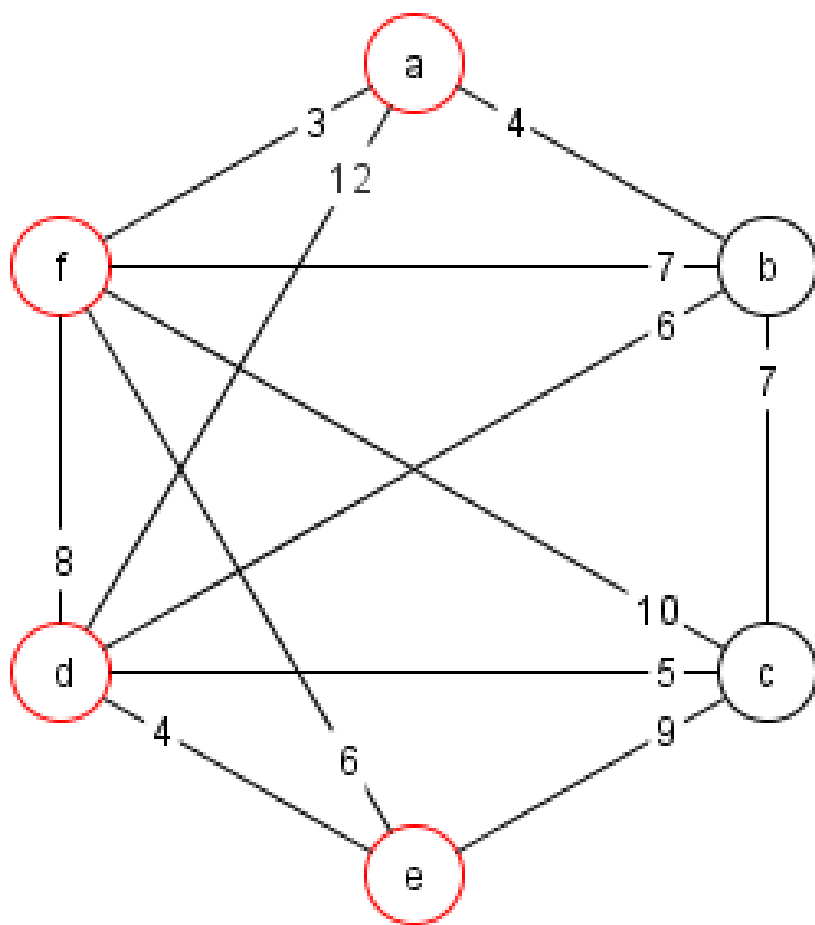
הקודקודים בעלי הדרגה האי-זוגית הם:

(3) A

(5) D

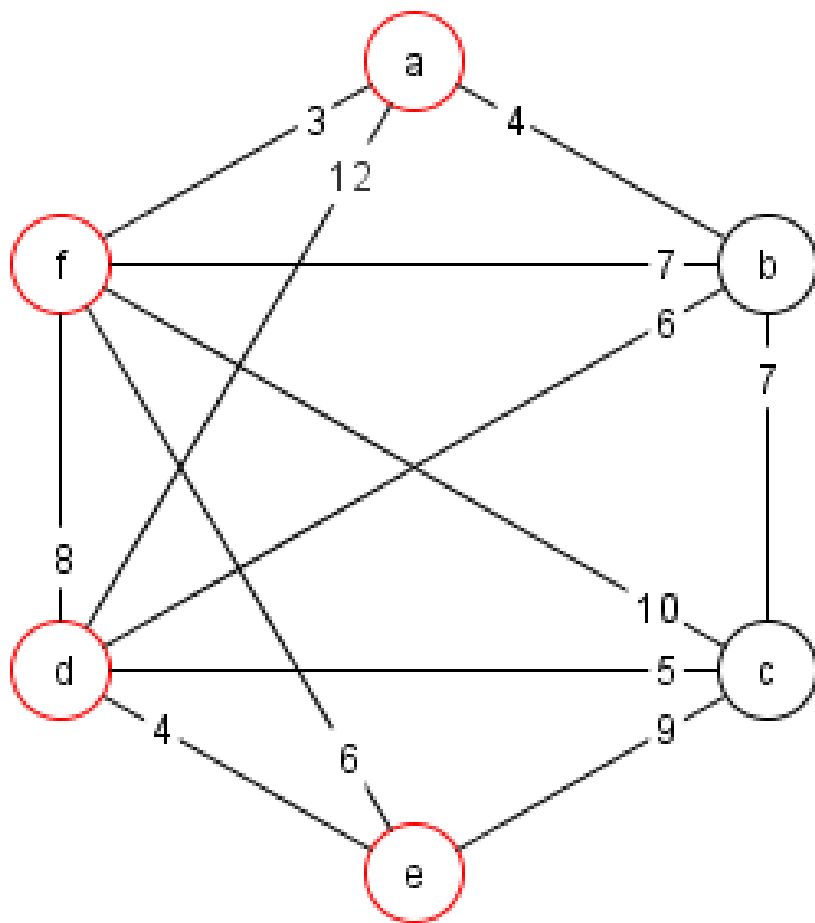
(3) E

(5) F



דוגמא

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



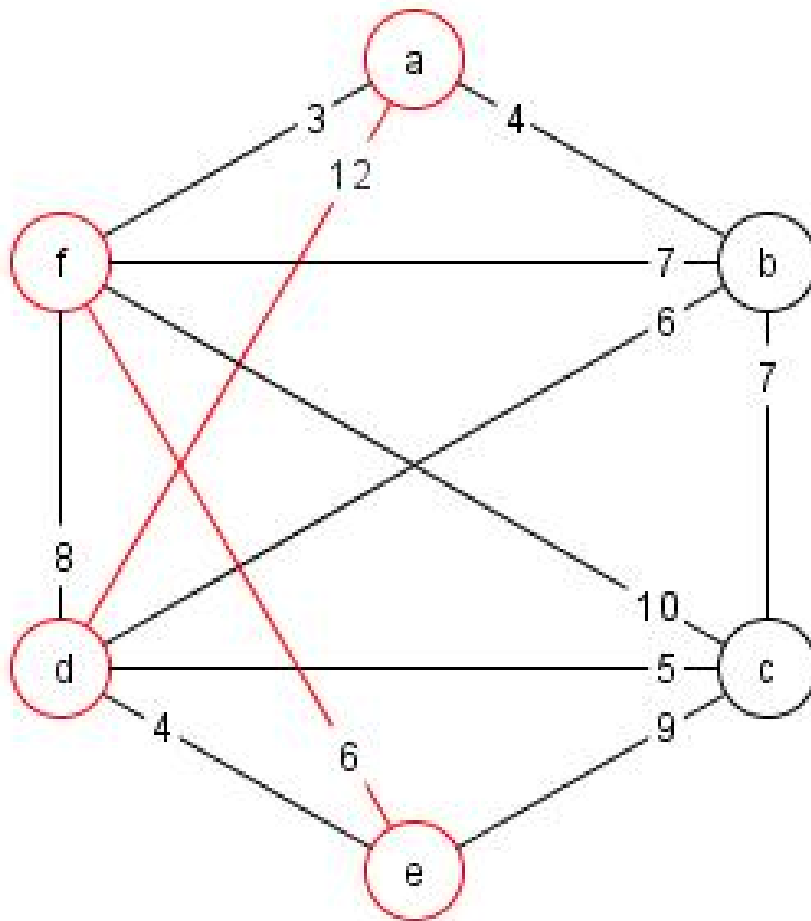
F-E-ו D-A

F-D-ו E-A

E-D-ו F-A

דוגמא

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



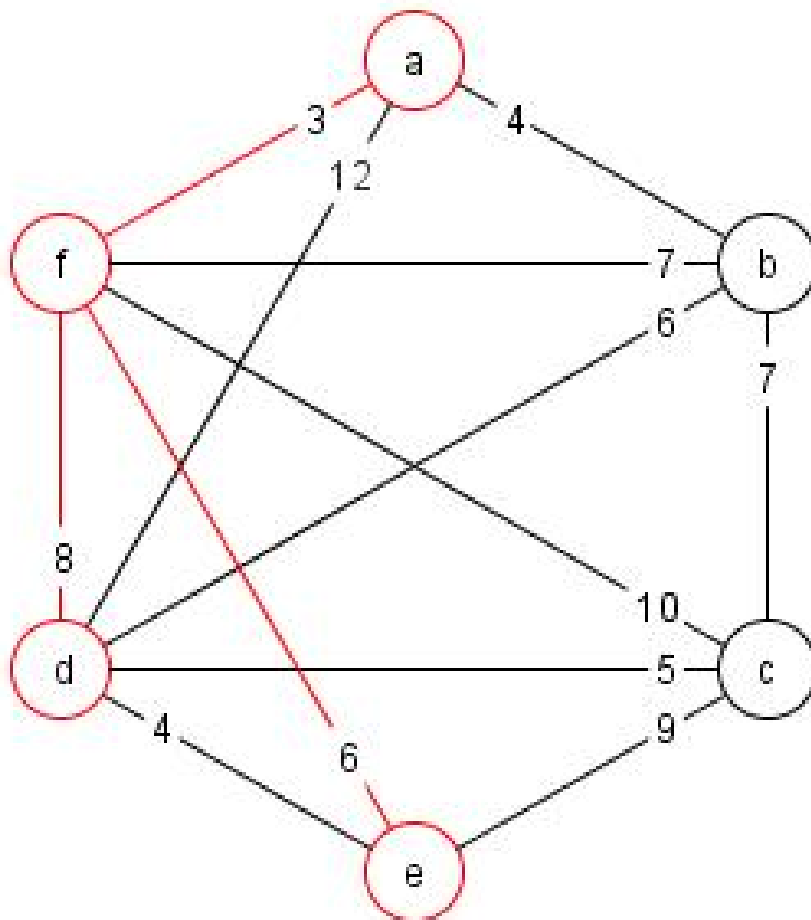
$$18 = 12 + 6 - F-E - D-A \quad \square$$

$$F-D - E-A \quad \square$$

$$E-D - F-A \quad \square$$

דוגמא

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



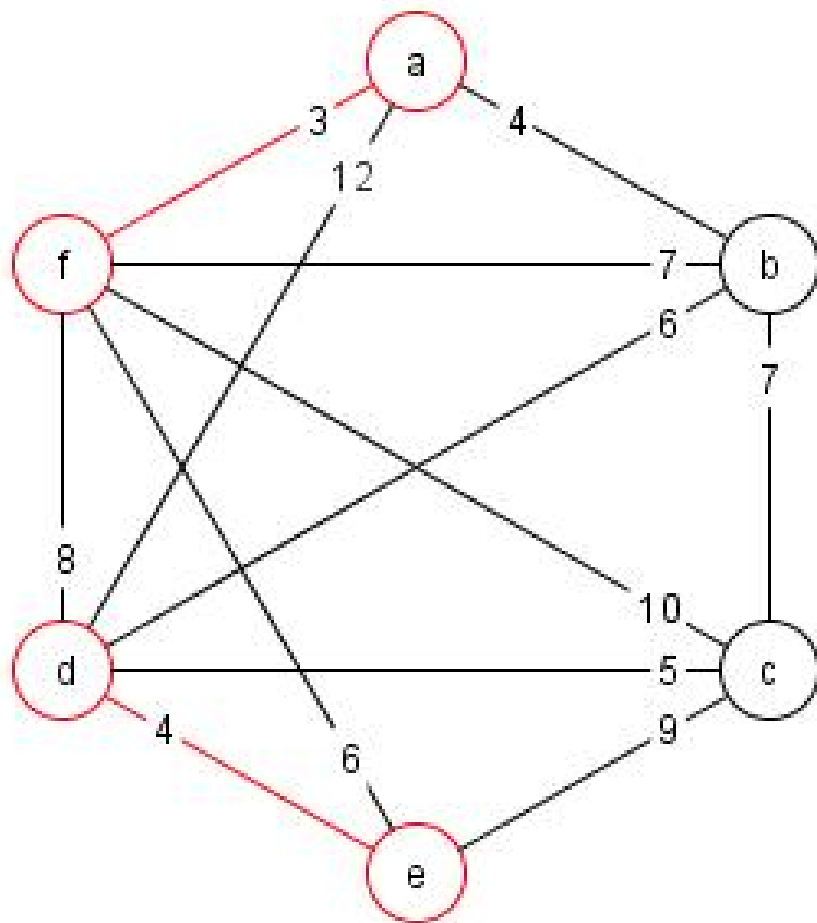
$$18 = 12 + 6 - F-E - D-A \quad \square$$

$$17 = 9 + 8 - F-D - E-A \quad \square$$

$$E-D - F-A \quad \square$$

דוגמא

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



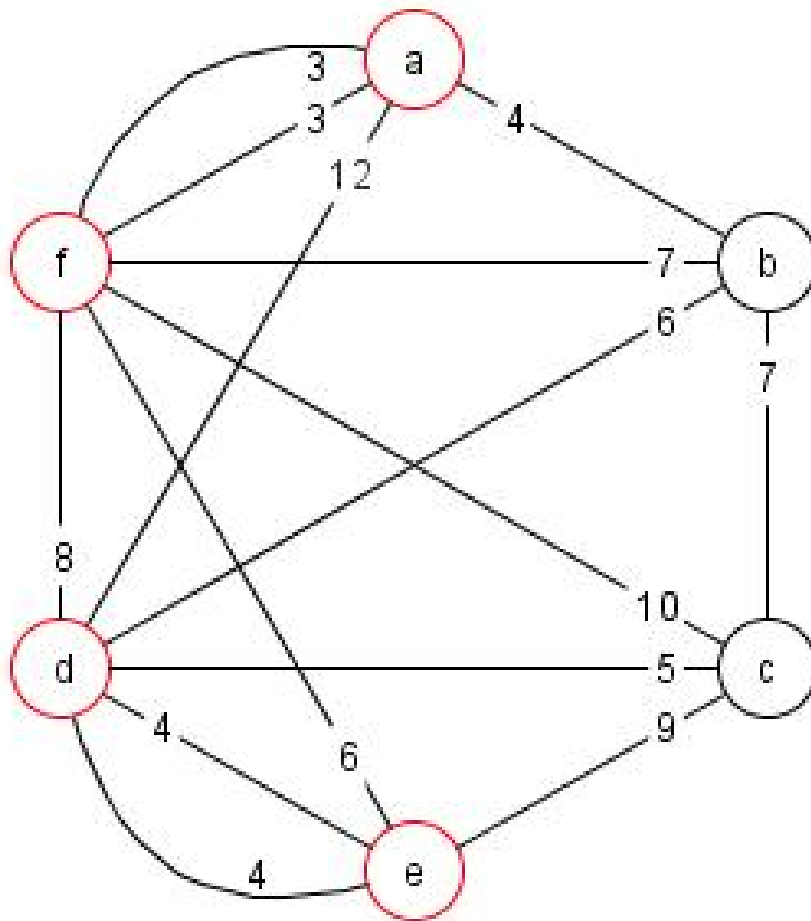
$$18 = 12 + 6 - F-E - D-A \quad \square$$

$$17 = 9 + 8 - F-D - E-A \quad \square$$

$$7 = 3 + 4 - E-D - F-A \quad \square$$

דוגמא

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



$$18 = 12 + 6 - F-E - D-A \quad \square$$

$$17 = 9 + 8 - F-D - E-A \quad \square$$

$$7 = 3 + 4 - E-D - F-A \quad \square$$

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 (בילתי פתיר אופטימאלית)

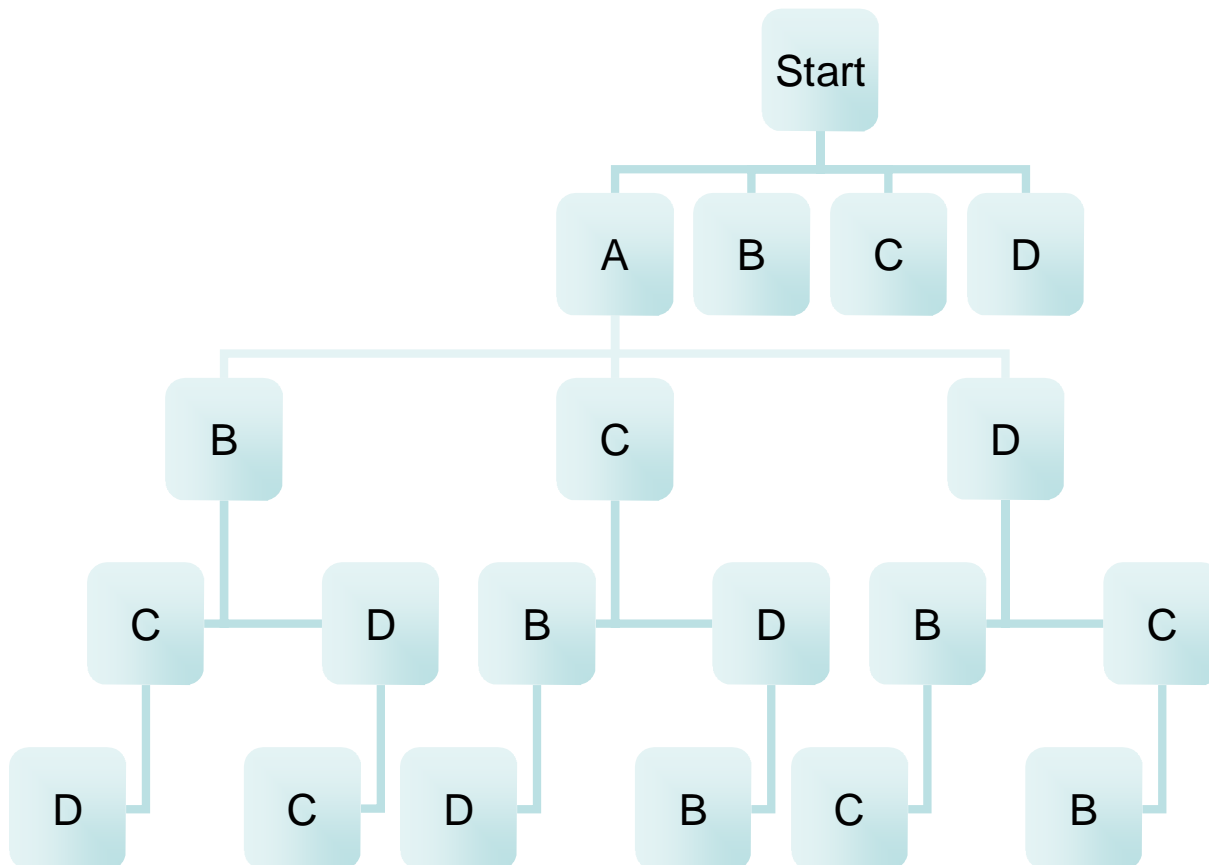
אלגוריתמים מקורבים

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 אלגוריתמים מדוייקים

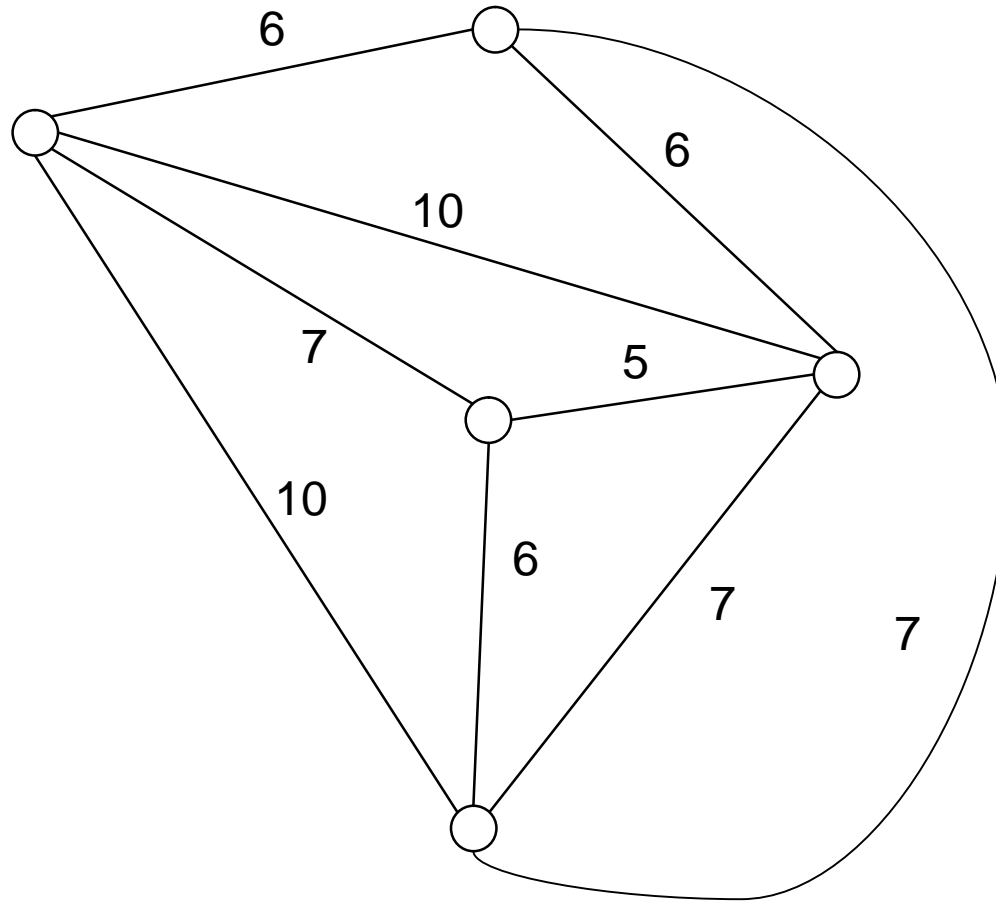
□ הפתרון הפשוט ביותר הוא לבנות את כל הקומבינציות האפשריות, לחשב את אורכן ולבחור את זו בעלת האורך הקצר ביותר.



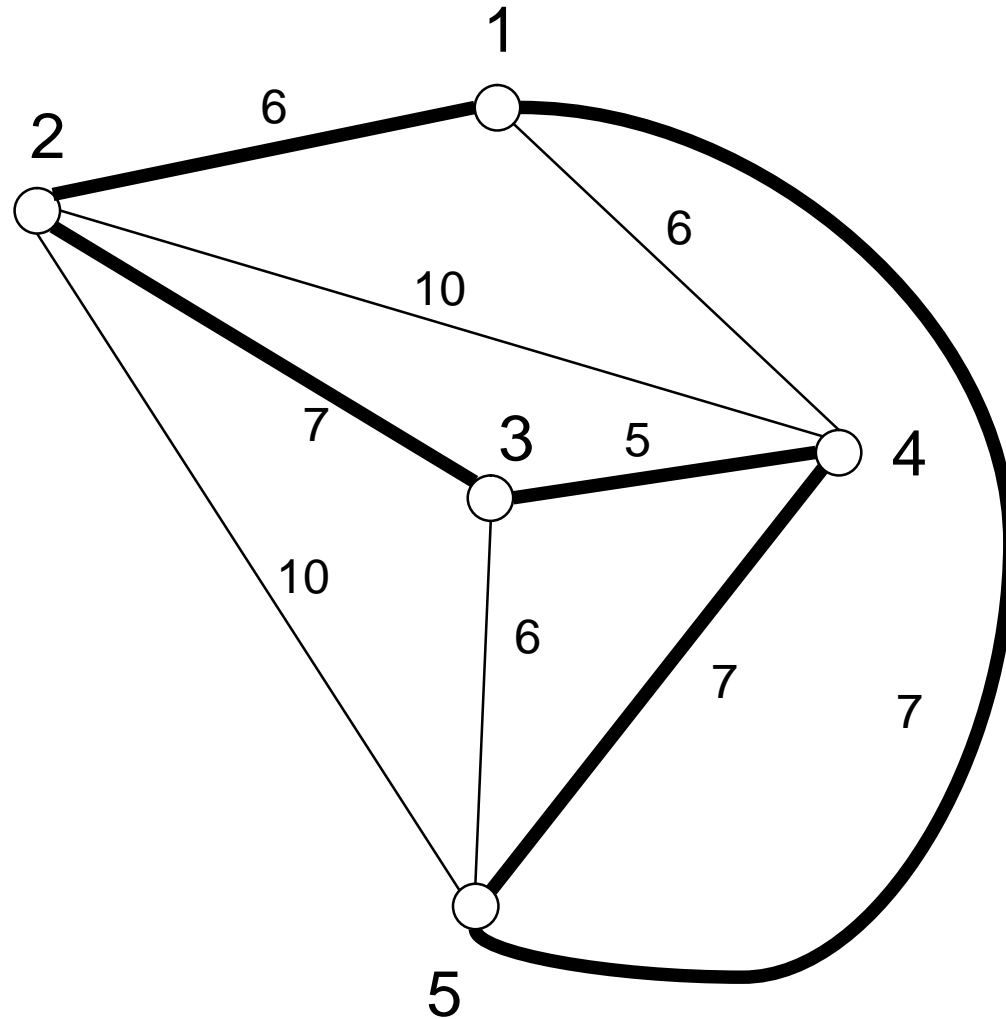
סוכן נוסע – אלגוריתם חמדני

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 $S_{Opt} \leq 2S_{Alg}$

בעיית הסוכן הנוסע – דוגמא



בעיית הסוכן הנוסע – דוגמא



סוכן נוסע – אלגוריתם מבוסס על עץ פורש

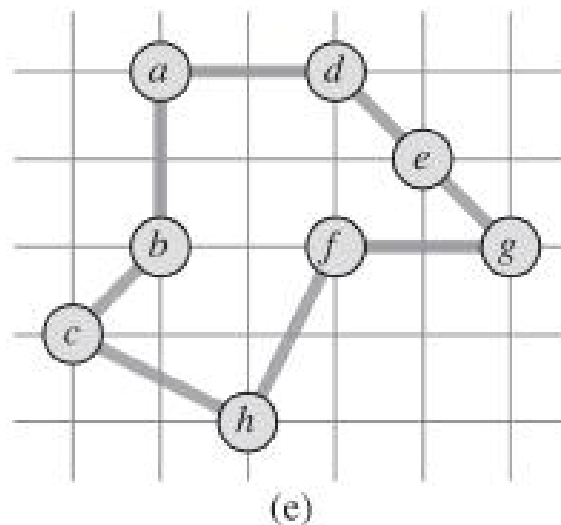
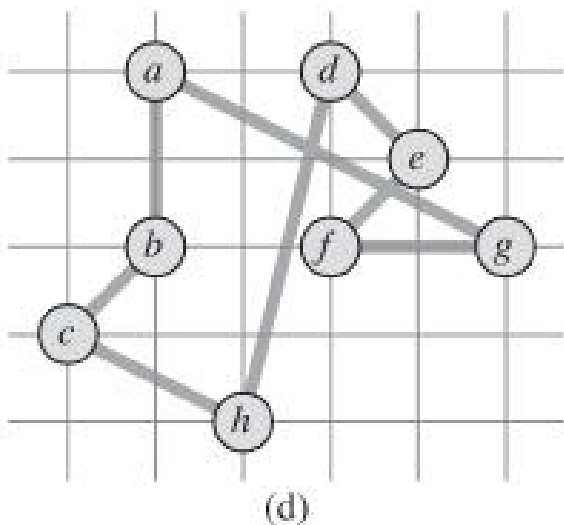
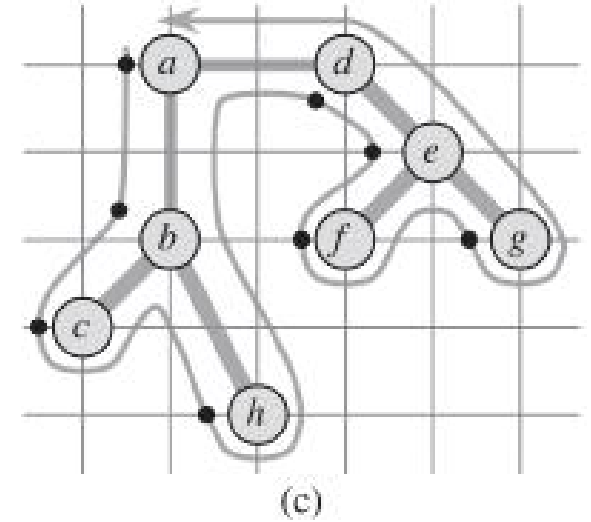
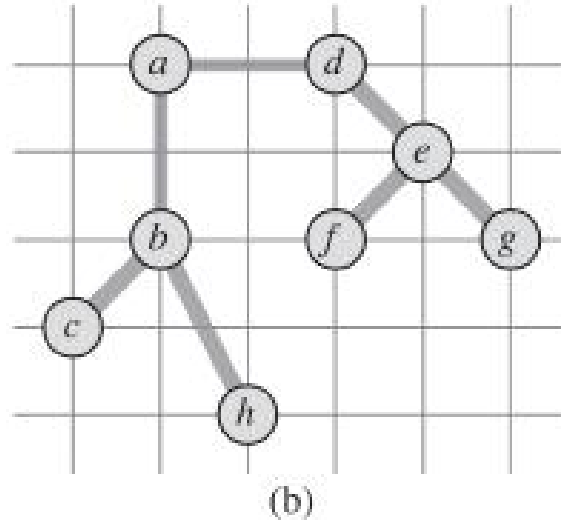
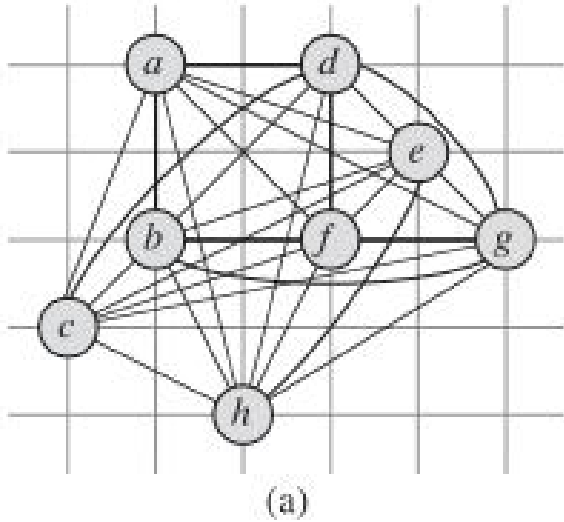
1. נבנה עץ פורש מינימאלי של הגרף

2. נבצע הליכה על הגרף

3. אם במהלך "ההליכה" אנו חוזרים לקודקוד שעברנו בו, אנו מתקנים את המסלול ע"י ביצוע קיצור דרך.

□ 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

אופטימאלי: 14.715

סוכן נוסע – Hill Climbing

Algorithm 2.4.1: Pseudocode for Stochastic Hill Climbing.

Input: $Iter_{max}$, ProblemSize

Output: Current

```
1 Current ← RandomSolution(ProblemSize);
2 foreach  $iter_i \in Iter_{max}$  do
3   | Candidate ← RandomNeighbor(Current);
4   | if Cost(Candidate)  $\geq$  Cost(Current) then
5   |   | Current ← Candidate;
6   | end
7 end
8 return Current;
```

סוכן נוסע – Simulated Annealing

Algorithm 4.2.1: Pseudocode for Simulated Annealing.

Input: ProblemSize, $iterations_{max}$, $temp_{max}$

Output: S_{best}

```

1  $S_{current} \leftarrow \text{CreateInitialSolution}(\text{ProblemSize});$ 
2  $S_{best} \leftarrow S_{current};$ 
3 for  $i = 1$  to  $iterations_{max}$  do
4    $S_i \leftarrow \text{CreateNeighborSolution}(S_{current});$ 
5    $temp_{curr} \leftarrow \text{CalculateTemperature}(i, temp_{max});$ 
6   if  $\text{Cost}(S_i) \leq \text{Cost}(S_{current})$  then
7      $S_{current} \leftarrow S_i;$ 
8     if  $\text{Cost}(S_i) \leq \text{Cost}(S_{best})$  then
9        $S_{best} \leftarrow S_i;$ 
10    end
11  else if  $\text{Exp}(\frac{\text{Cost}(S_{current}) - \text{Cost}(S_i)}{temp_{curr}}) > \text{Rand}()$  then
12     $S_{current} \leftarrow S_i;$ 
13  end
14 end
15 return  $S_{best};$ 

```

TABU Search – סוכן נוסע

Algorithm 2.10.1: Pseudocode for Tabu Search.

```

Input:  $TabuList_{size}$ 
Output:  $S_{best}$ 
1  $S_{best} \leftarrow \text{ConstructInitialSolution}();$ 
2  $TabuList \leftarrow \emptyset;$ 
3 while  $\neg \text{StopCondition}()$  do
4    $CandidateList \leftarrow \emptyset;$ 
5   for  $S_{candidate} \in S_{best_{neighborhood}}$  do
6     if  $\neg \text{ContainsAnyFeatures}(S_{candidate}, TabuList)$  then
7        $CandidateList \leftarrow S_{candidate};$ 
8     end
9   end
10   $S_{candidate} \leftarrow \text{LocateBestCandidate}(CandidateList);$ 
11  if  $\text{Cost}(S_{candidate}) \leq \text{Cost}(S_{best})$  then
12     $S_{best} \leftarrow S_{candidate};$ 
13     $TabuList \leftarrow \text{FeatureDifferences}(S_{candidate}, S_{best});$ 
14    while  $TabuList > TabuList_{size}$  do
15       $\text{DeleteFeature}(TabuList);$ 
16    end
17  end
18 end
19 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}

```
1 Population ← InitializePopulation( $Population_{size}$ ,  
    $Problem_{size}$ );  
2 EvaluatePopulation(Population);  
3  $S_{best}$  ← GetBestSolution(Population);  
4 while ¬StopCondition() do  
5     Parents ← SelectParents(Population,  $Population_{size}$ );  
6     Children ←  $\emptyset$ ;  
7     foreach  $Parent_1, Parent_2 \in$  Parents do  
8          $Child_1, Child_2$  ← Crossover( $Parent_1, Parent_2, P_{crossover}$ );  
9         Children ← Mutate( $Child_1, P_{mutation}$ );  
10        Children ← Mutate( $Child_2, P_{mutation}$ );  
11    end  
12    EvaluatePopulation(Children);  
13     $S_{best}$  ← GetBestSolution(Children);  
14    Population ← Replace(Population, Children);  
15 end  
16 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 היוריסטיים או מקורבים

בעיית ניתוב רכבים – דוגמא



בעיית ניתוב רכבים – דוגמא



בעיית ניתוב רכבים – ניסוח LP

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

subject to

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

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

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

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

$$\sum_{i=1}^n t_i^v \sum_{j=1}^n x_{ij}^v + \sum_{i=1}^n \sum_{j=1}^n t_{ij}^v x_{ij}^v \leq T_v \quad \forall v = 1, \dots, NV \quad \text{Time limit}$$

$$\sum_{j=2}^n x_{1j}^v \leq 1 \quad \forall v = 1, \dots, NV \quad \text{Vehicle availability}$$

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

$$x_{ij}^v \in \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right\}$$

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	<input type="checkbox"/> רכב בודד <input type="checkbox"/> צי רכבים
סוג צי	<input type="checkbox"/> הומוגני <input type="checkbox"/> הטרוגני <input type="checkbox"/> רכבים מיוחדים
מיקום הרכבים	<input type="checkbox"/> מרכז אחד <input type="checkbox"/> מספר מרכזים
אופי הביקוש	<input type="checkbox"/> דטרמיניסטי <input type="checkbox"/> סטוכסטי <input type="checkbox"/> מימוש חלקי של הביקוש
מיקום הביקוש	<input type="checkbox"/> בקשתות (בכולן או חלקן) <input type="checkbox"/> בקודקודים (בכולם או חלקם) <input type="checkbox"/> מעורב

בעיית ניתוב רכבים – מאפיינים (המשך)

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

בעיית ניתוב רכבים – מאפיינים (המשך)

אפשרויות לבעיה	מאפיינים
<ul style="list-style-type: none"> <input type="checkbox"/> משתנות (נסיעה) <input type="checkbox"/> קבועות (תפעול קבוע, רכש ופחת) <input type="checkbox"/> קנסות 	<p>עלויות</p>
<ul style="list-style-type: none"> <input type="checkbox"/> מינימום עלות ניתוב <input type="checkbox"/> מינימום סכום עלויות כולל <input type="checkbox"/> מינימום כלי רכב דרושים <input type="checkbox"/> מקסימום תועלת מצד המפעיל <input type="checkbox"/> מקסימום תועלת מצד הלקוח 	<p>פונקציות מטרה</p>

בעיית ניתוב רכבים – אלגוריתם Savings

1. בונים פיתרון 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.

בעיית ניתוב רכבים – הקשר לתחבורה ציבורית

תכנון רשת

קביעת תדירויות

קביעת לוח זמנים

שיבוץ רכבים

שיבוץ נהגים