§Approximation Algorithms

Up to now, the best algorithm for solving an NP-complete problem requires exponential time in the worst case. It is too time-consuming.

To reduce the time required for solving a problem, we can relax the problem, and obtain a feasible solution "close" to an optimal solution.

- **An approximation algorithm for convex hulls**

A convex hull of n points in the plane can be computed in O(nlogn) time in the worst case.

An approximation algorithm:

Step1:Find the leftmost and rightmost points.

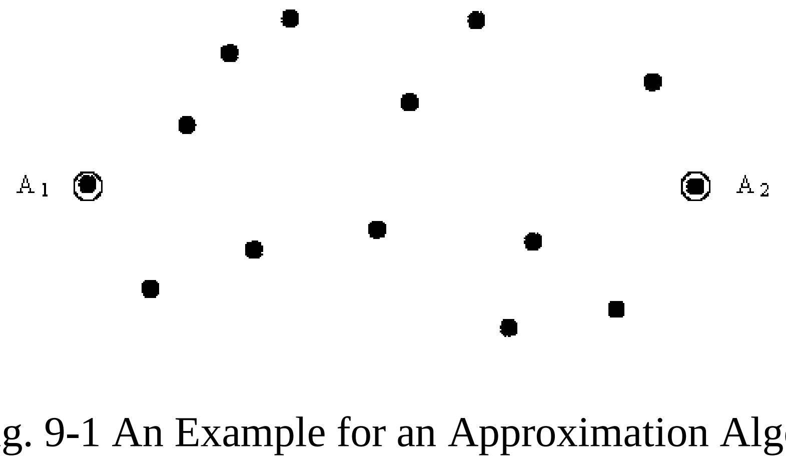


Fig. 9-1 An Example for an Approximation Algorithm for Convex Hulls

Step2:Divide the points into K strips. Find the highest and lowest points in each strip.

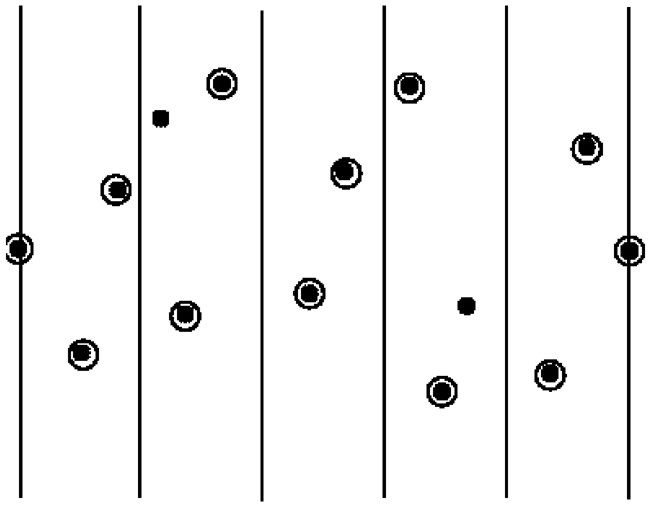


Fig. 9-2 Dividing Points into Strips

Step3:Apply the Graham scan to those highest and lowest points to construct an approximate convex hull. (The highest and lowest points are already sorted by their x-coordinates.)

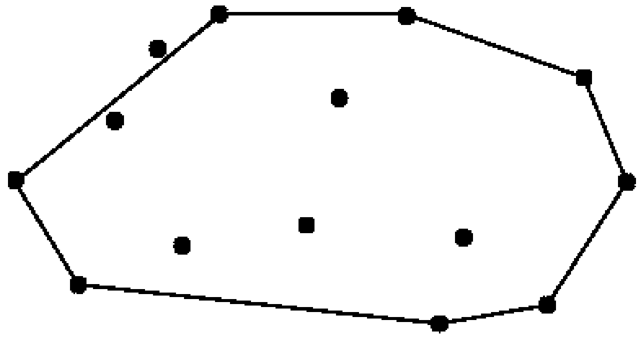


Fig. 9-3 An Approximation Convex Hull

Algorithm 9-1 An approximation Algorithm for Convex Hull.

Input: A set of n points.

Output: An approximate convex hull of S.

Step 1: Find the leftmost and right most points of S, denoted as A_1 and A_2, respectively. (with minimum and maximum x-coordinates respectively).

Step 2: Divide the area bounded by A_1 and A_2 into k equally spaced strips and for each strip, select the points with the minimum and maximum y-coordinates. Denote the set of points selected in this step together with A_1 and A_2 as set P.

Step 3: Construct the convex hull of P and use that as the approximate convex hull of S.

time complexity: O(n+k)

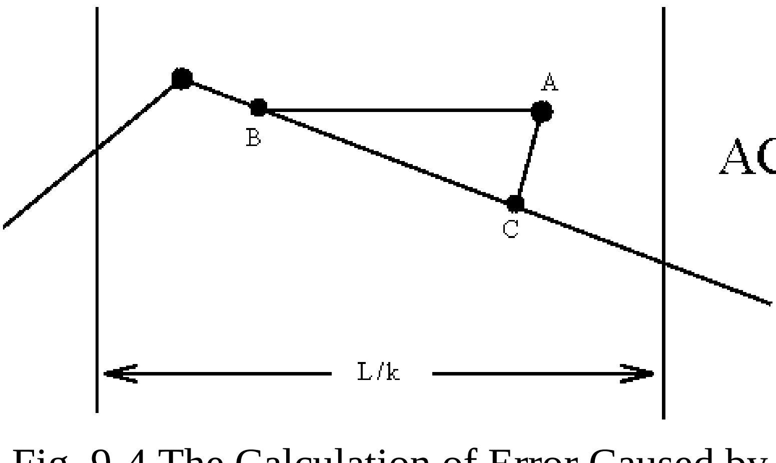
Step 1: O(n)

Step 2: O(n)

Step 3: O(k)

How far away the points outside are from the approximate convex hull? L/K.

L: the distance between the leftmost and rightmost points.



$$AC \le AB \le \frac{L}{K}$$

Fig. 9-4 The Calculation of Error Caused by the Approximation

- **An approximation algorithm for Euclidean**

traveling salesperson problem (ETSP).

The ETSP is to find a shortest closed path through a set S of n points in the plane.

The ETSP is NP-hard.

Algorithm 9-2 An Approximation Algorithm for ETSP

Input: A set S of n points in the plane.

Output: An approximate traveling salesperson tour of S.

Step 1: Find a minimal spanning tree T of S.

Step 2: Find a minimal Euclidean weighted matching M on the set of vertices of odd degrees in T. Let $G=M\cup T$.

Step 3: Find an Eulerian cycle of G and then traverse it to find a Hamiltonian cycle as an approximate tour of ETSP by bypassing all previously visited vertices.

e.g.

Step1:

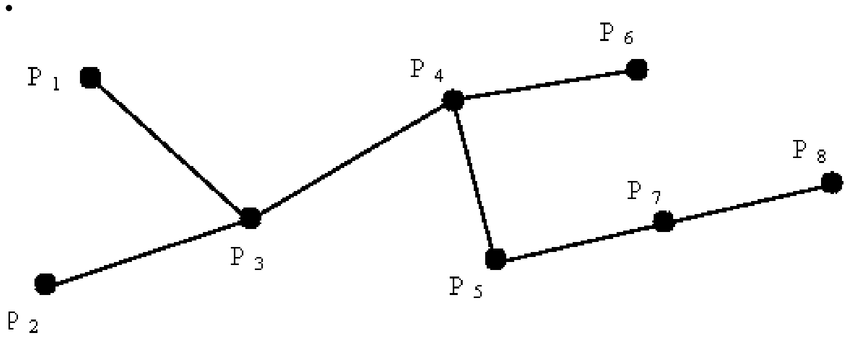


Fig. 9-6 A Minimal Spanning Tree of Eight Points

Step2:The number of points with odd degrees must be even. $\because \sum_{i=1}^{n} d_i = 2|E|$, even

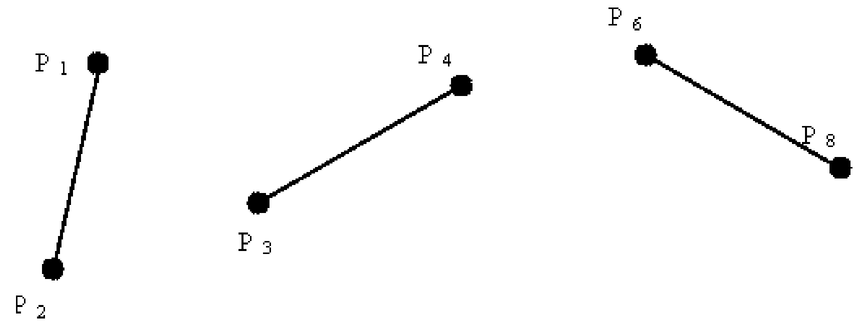


Fig. 9-7 A Minimal Weighted Matching of Six Vertices.

Step3:

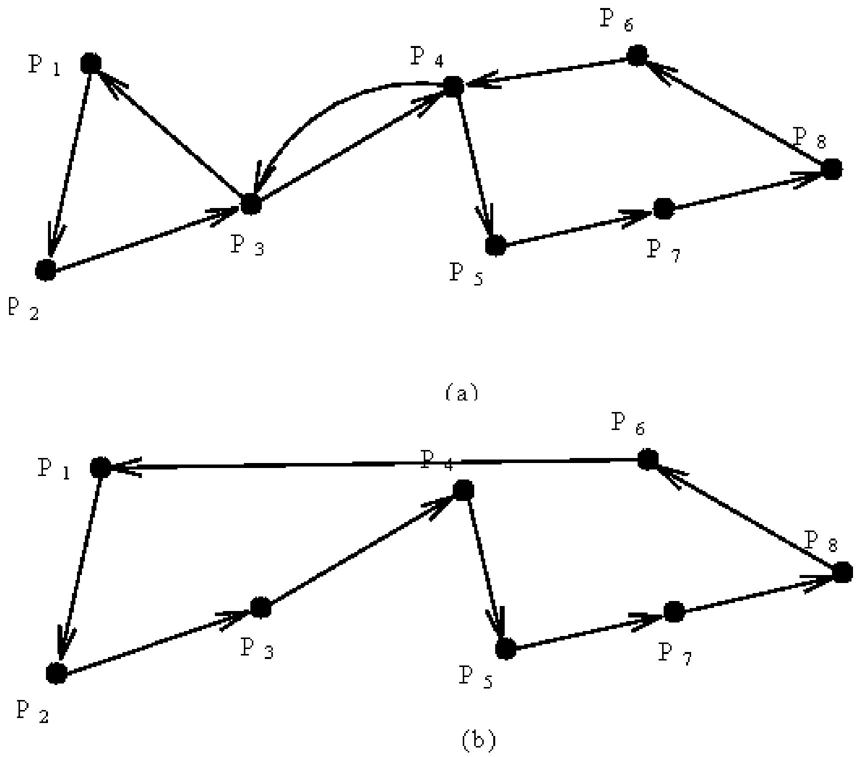


Fig 9-8. An Eulerian Cycle and the Resulting Approximate Tour

time complexity: $O(n^3)$

Step 1: $O(n\log n)$

Step 2: $O(n^3)$

Step 3: $O(n)$

How close the approximate solution to an optimal solution?

The approximate tour is within 3/2 of an optimal one.

Reasoning:

L: optimal tour

$j_1 \cdots i_1 j_2 \cdots i_2 j_3 \cdots i_{2m}$

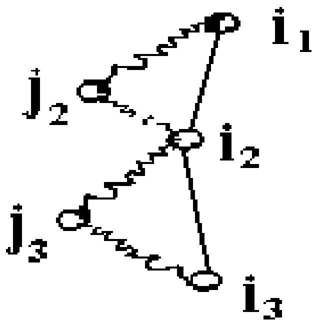
$\{i_1, i_2, \cdots, i_{2m}\}$: the set of odd degree vertices in T.

2 matchings: $M_1=\{[i_1,i_2],[i_3,i_4],\cdots,[i_{2m-1},i_{2m}]\}$

$M_2=\{[i_2,i_3],[i_4,i_5],\cdots,[i_{2m},i_1]\}$

$\text{length}(L) \geq \text{length}(M_1) + \text{length}(M_2)$ (triangular inequality)

$\geq 2\ \text{length}(M)$



$\Rightarrow \text{length}(M) \leq 1/2\ \text{length}(L)$

$G = T \cup M$

$\Rightarrow \text{length}(G) = \text{length}(T) + \text{length}(M)$

$\leq \text{length}(L) + 1/2\ \text{length}(L)$

$= 3/2\ \text{length}(L)$

- An approximation algorithm for the bottleneck traveling salesperson problem
  - minimize the longest edge of a tour.
  - This is a mini-max problem.
  - This problem is NP-complete.
  - The input data for this problem fulfill the following assumptions:

(i) The graph is a complete graph.
(ii)All edges obey the triangular inequality rule.

- An algorithm for finding an optimal solution:

Step1: Sort all edges in G = (V,E) into a nondecresing sequence $|e_1| \leq |e_2| \leq \cdots \leq |e_m|$. Let $G(e_i)$ denote the subgraph obtained from G by deleting all edges longer than e_i.
Step2: $i \leftarrow 1$
Step3: If there exists a Hamiltonian cycle in $G(e_i)$, then this cycle is the solution and stop.
Step4: $i \leftarrow i+1$. Go to Step 3.

e.g.

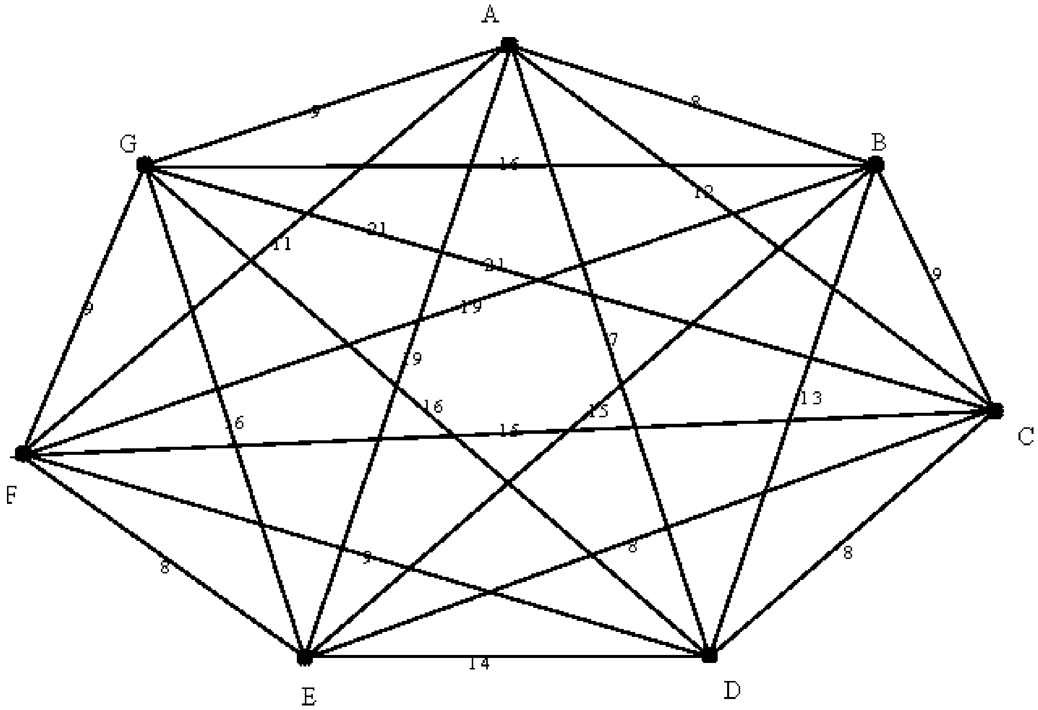


Fig. 9-9 A Complete Graph

e.g.

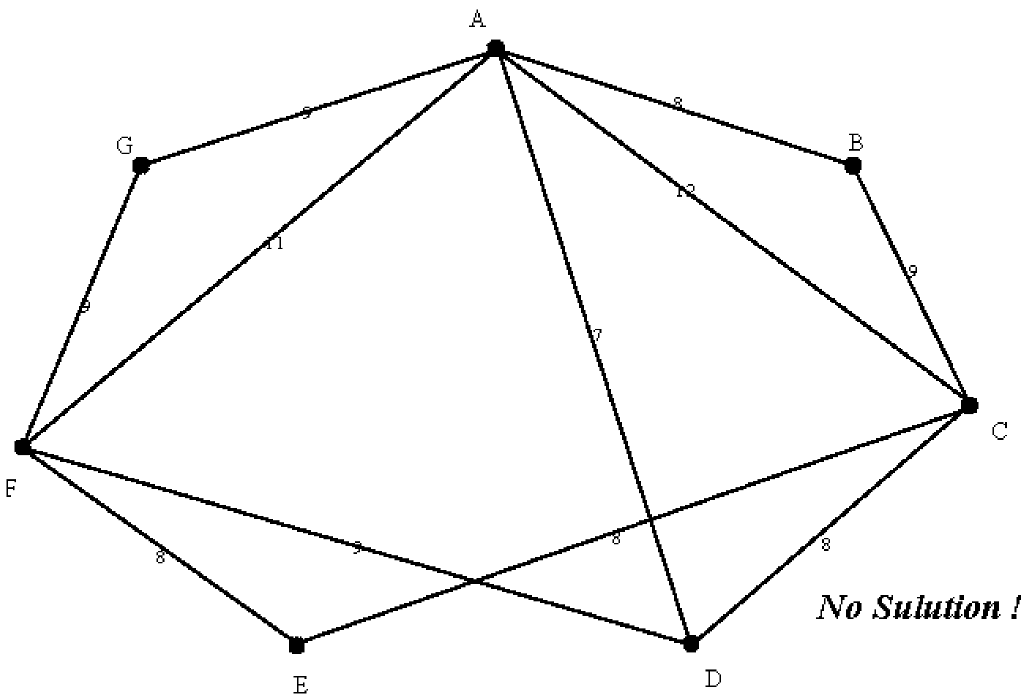


Fig. 9-10 G(AC) of the Graph in Fig 9-9

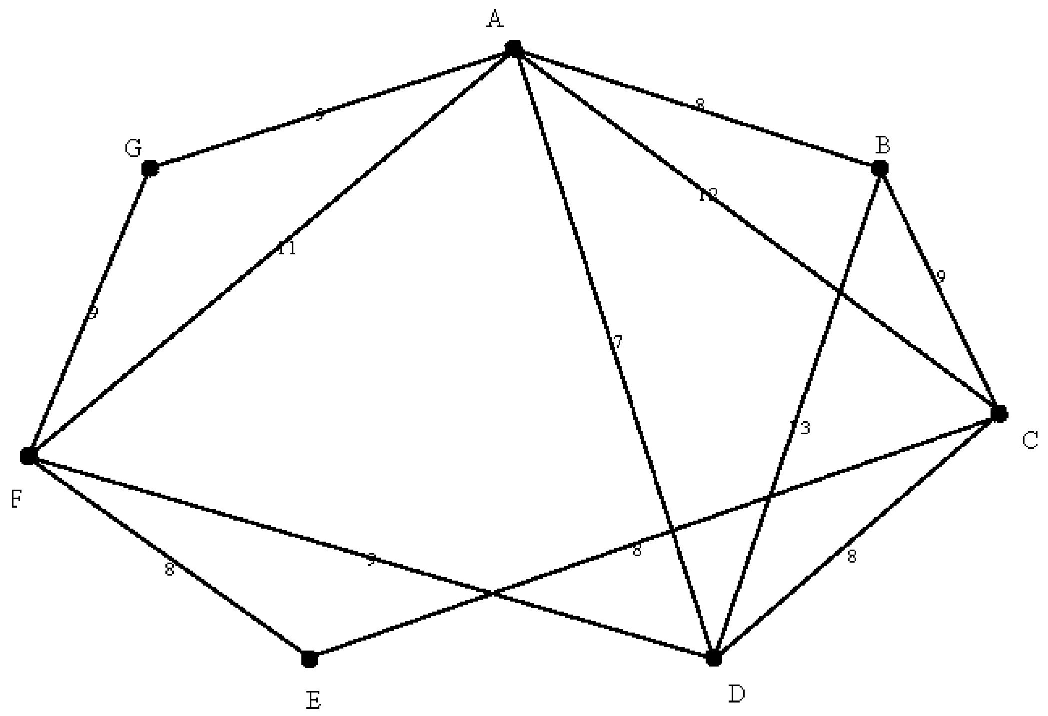


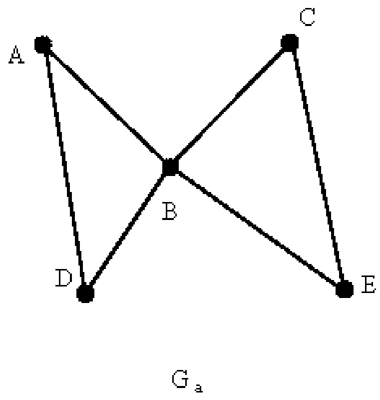
Fig. 9-11 G(BD) of Graph in Fig 9-9

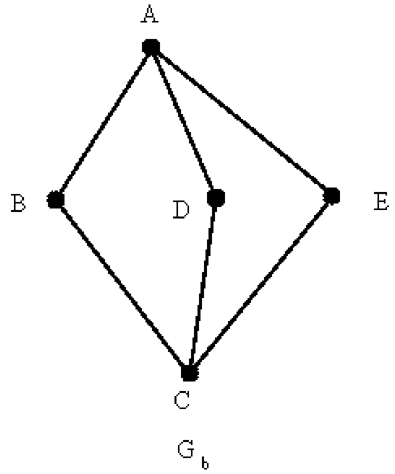
There is a Hamiltonian cycle, A-B-D-C-E-F-G-A, in G(BD).

The optimal solution is 13.

- Def: The t-th power of G=(V,E), denoted as $G^t=(V,E^t)$, is a graph that an edge $(u,v) \in E^t$ if there is a path from u to v with at most t edges in G.
- If a graph G is bi-connected, then G^2 has a Hamiltonian cycle.

e.g.





not bi-connected bi-connected

e.g.

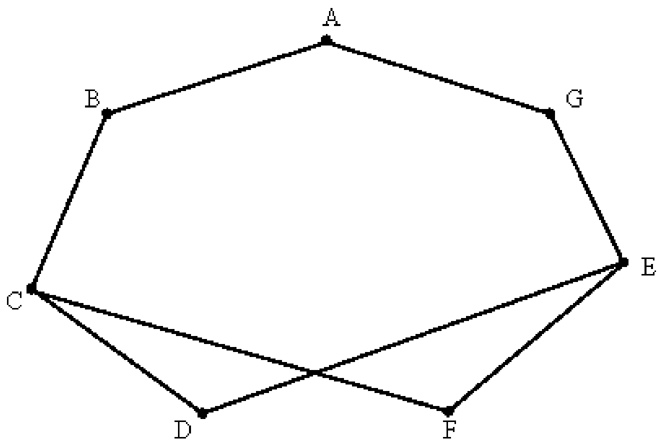
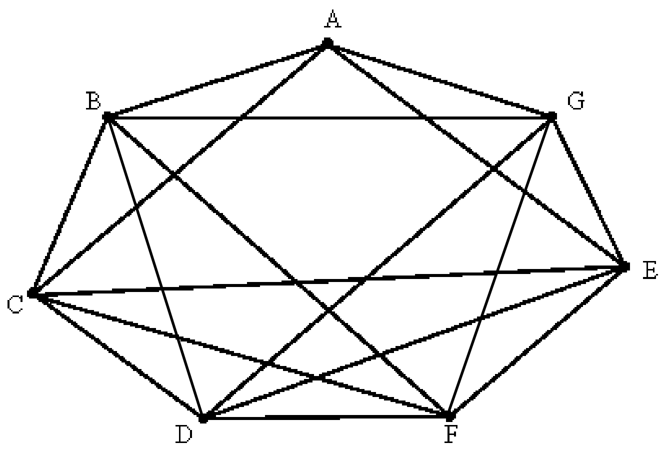


Fig. 9-13 A Bi-Connected Graph

e.g.



A Hamiltonian cycle

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

Fig. 9-14 G^2 of the Graph in Fig 9-13.

Algorithm 9-3 An Approximation Algorithm to Solve the Special Bottleneck Traveling Salesperson Problem.

Input: A complete graph $G=(V,E)$ where all edges satisfy triangular inequality.

Output: A tour in G whose longest edges is not greater than twice of the value of an optimal solution to the special bottleneck traveling salesperson problem of G.

Step 1: Sort the edges into $|e_1| \le |e_2| \le \cdots \le |e_m|$.

Step 2: $i := 1$.

Step 3: If $G(e_i)$ is bi-connected, construct $G(e_i)^2$, find a Hamiltonian cycle in $G(e_i)^2$ and return this as the output, otherwise, go to Step 4.

Step 4: $i := i + 1$. Go to Step 3.

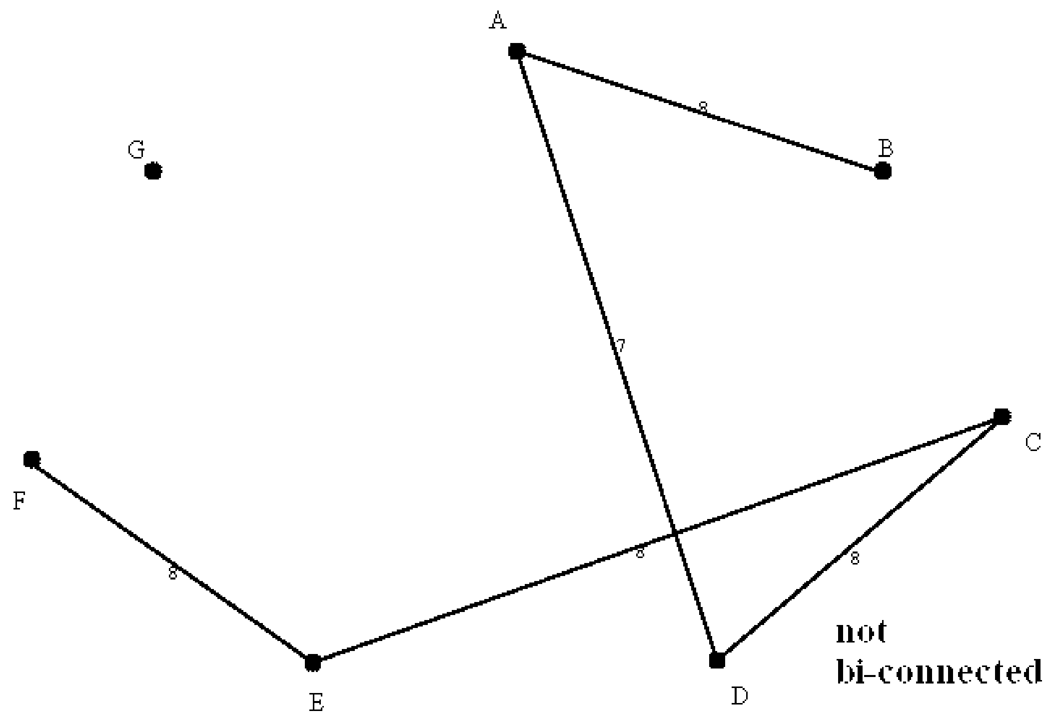


Fig. 9-15 G(FE) of the Graph in Fig 9-9

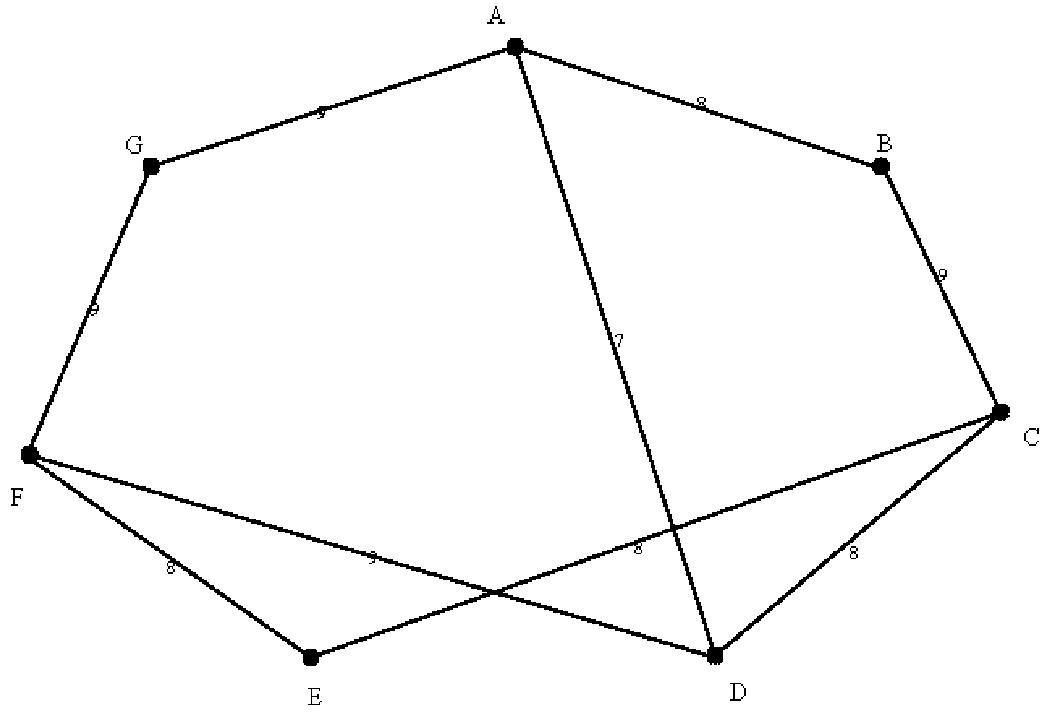


Fig. 9-16 G(FG) of the Graph in Fig 9-9

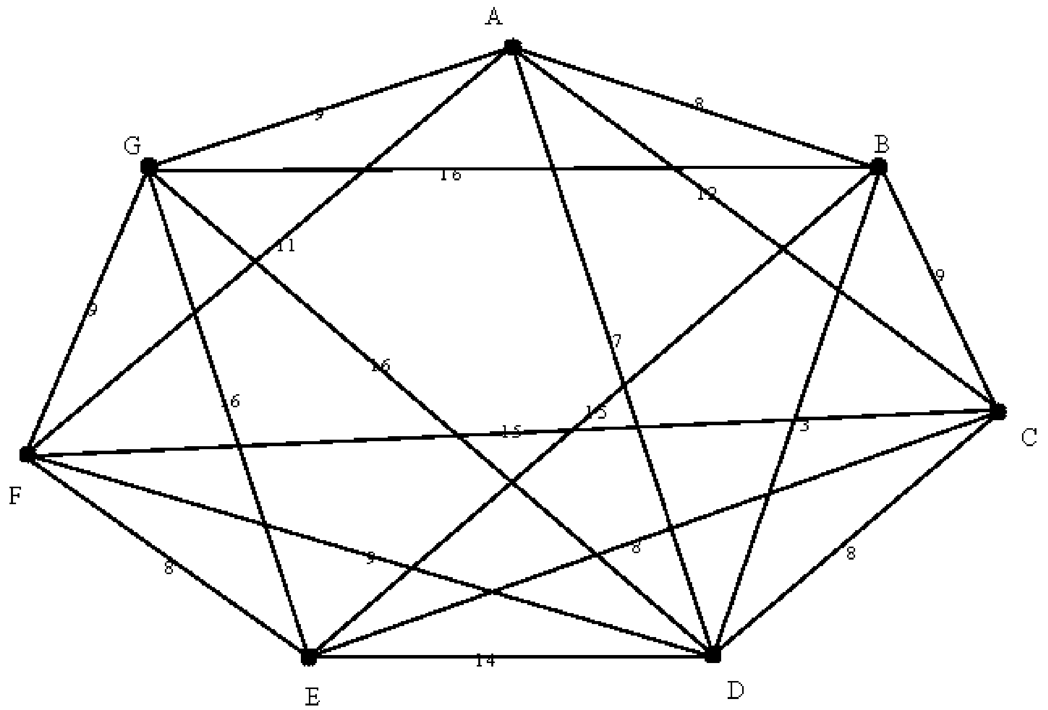


Fig. 9-17 $G(FG)^2$

A Hamiltonian cycle: A-G-F-E-D-C-B-A.

the longest edge: 16

time complexity:

polynomial time

The approximate solution is bounded by two times an optimal solution.

Reasoning:

A Hamiltonian cycle is bi-connected.

e_{op}: the longest edge of an optimal solution

$G(e_i)$: the first bi-connected graph

$|e_i| \leq |e_{op}|$

The length of the longest edge in $G(e_i)^2 \leq 2|e_i|$

(triangular inequality) $\leq 2|e_{op}|$

If there is a polynomial approximation algorithm which produces a bound less than two, then NP=P.
(The Hamiltonian cycle decision problem reduces to this problem.)

Proof:

For an arbitrary graph G=(V,E), we expand G to a complete G_c:

$C_{ij} = 1$ if $(i,j) \in E$

$C_{ij} = 2$ if otherwise

(The definition of C_{ij} satisfies the triangular inequality.)

Let V^* denote the value of an optimal solution of the bottleneck TSP of G_c.

$V^* = 1 \Leftrightarrow$ G has a Hamiltonian cycle

Because there are only two kinds of edges, 1 and 2 in G_c, if we can produce an approximate solution whose value is less than $2V^*$, then we can also solve the Hamiltonian cycle decision problem.

- **An approximation algorithm for the bin packing problem**

n items $a_1, a_2, \cdots, a_n$, $0 < a_i \le 1, 1 \le i \le n$
to determine the minimum number of bins of unit capacity to accomodate all n items.

e.g. n = 5, {0.3, 0.5, 0.8, 0.2 0.4}

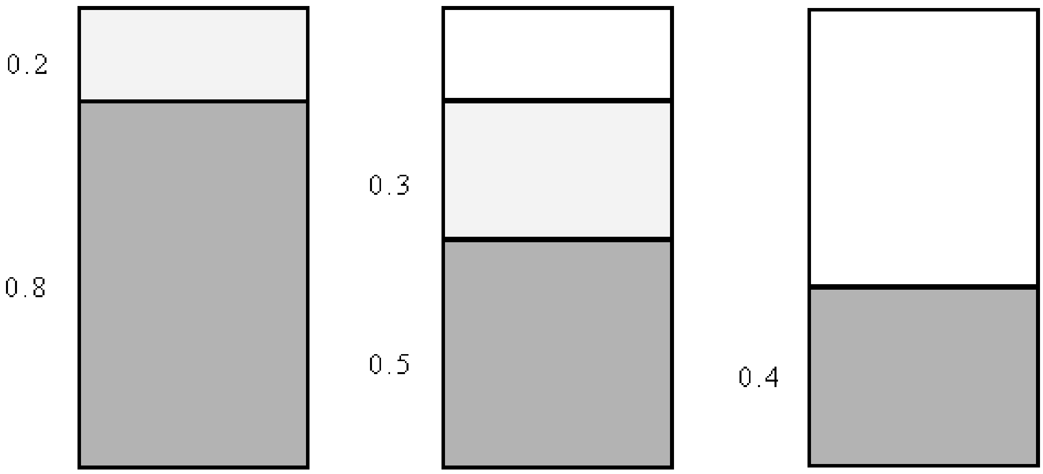


Fig. 9-27 An Example of the Bin-Packing Problem

- The bin packing problem is NP-hard.

An approximation algorithm: (first-fit)

place a_i into the lowest-indexed bin which can accommodate a_i.

$S(a_i)$: the size of a_i

$OPT(I)$: the size of an optimal solution of an instance I

$FF(I)$: the size of bins in the first-fit algorithm

$C(B_i)$: the sum of the sizes of a_j's packed in bin B_i in the first-fit algorithm

$$OPT(I) \geq \sum_{i=1}^{n} S(a_i)$$

$C(B_i) + C(B_{i+1}) > 1$

m nonempty bins:

$C(B_1)+C(B_2)+\cdots+C(B_m) > m/2$

$$\Rightarrow FF(I) = m < 2\sum_{i=1}^{m} C(B_i) = 2\sum_{i=1}^{n} S(a_i) \leq 2\ OPT(I)$$

$FF(I) < 2\ OPT(I)$

An approximation algorithm for the rectilinear m-center problem

- The sides of a rectilinear square are parallel or perpendicular to the x-axis of the Euclidean plane.
- The problem is to find m rectilinear squares covering all of the n given points such that the maximum side length of these squares is minimized.

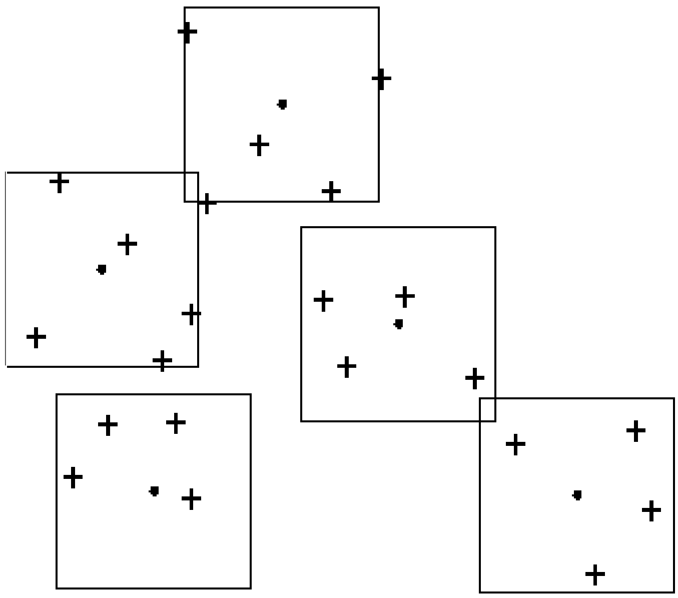


Fig. 9-28 A Rectilinear 5-center Problem Instance

- This problem is NP-complete.
- This problem for the solution with error ratio < 2 is also NP-complete.

input: P={P1, P2, ⋯, Pn}

The size of an an optimal solution must be equal to one of the L_∞(Pi,Pj), $1 \le i < j \le n$, where L_∞((x1,y1),(x2,y2)) = max{|x1-x2|,|y1-y2|}.

Algorithm 9-5 Approximation Algorithm Rectilinear Center

Input: A set P of n points, number of centers: m

Output: SQ[1], ⋯, SQ[m]: A feasible solution of the rectilinear m-center problem with size less than or equal to twice of the size of an optimal solution.

Step 1: Compute rectilinear distances of all pairs of two points and sort them together with 0 into an ascending sequence D[0]=0, D[1], ⋯, D[n(n-1)/2].

Step 2: LEFT := -1, RIGHT := n(n-1)/2.

Step 3: i := $\lceil$(LEFT + RIGHT)/2$\rceil$.

Step 4: If Test(m, P, D[i]) is not “failure” then
        RIGHT := i
    else
        LEFT := i

Step 5: If RIGHT = LEFT + 1 then
        return Test(m, P, D[RIGHT])
    else
        go to Step 3.

Algorithm 9-6 Algorithm Test(m, P, r):

Input: point set: P, number of centers: m, size: r.

Output: "failure", or SQ[1], ⋯, SQ[m] m squares of size 2r covering P.

Step 1: PS := P

Step 2: For i := 1 to m do

    If PS ≠ ∅ then

        p := the point is PS with the smallest x-value

        SQ[i] := the square of size 2r with center at p

      PS := PS - {points covered by SQ[i]}

    else

      SQ[i] := SQ[i-1].

Step 3: IF PS = ∅ then

      return SQ[1], ⋯, SQ[m]

    else

      return "failure".

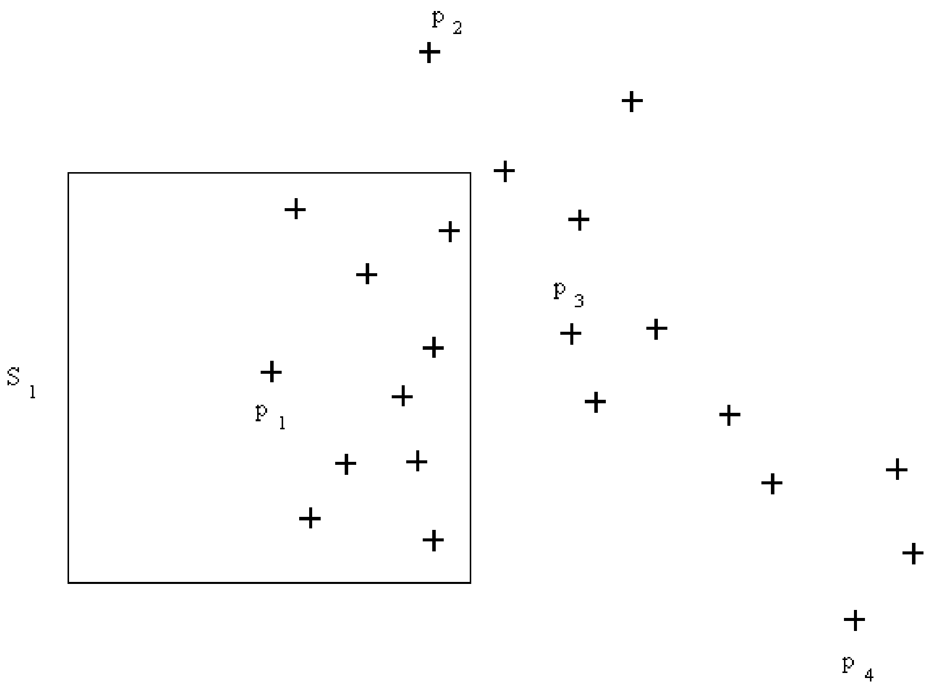


Fig. 9-29 The First Application of the Relaxed Test Subroutine.

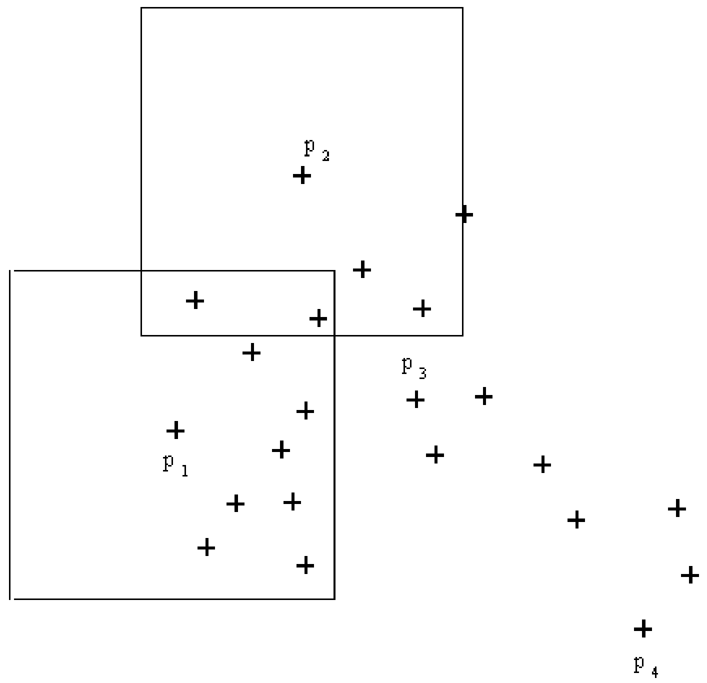


Fig. 9-30 The Second Application of the Test Subroutine.

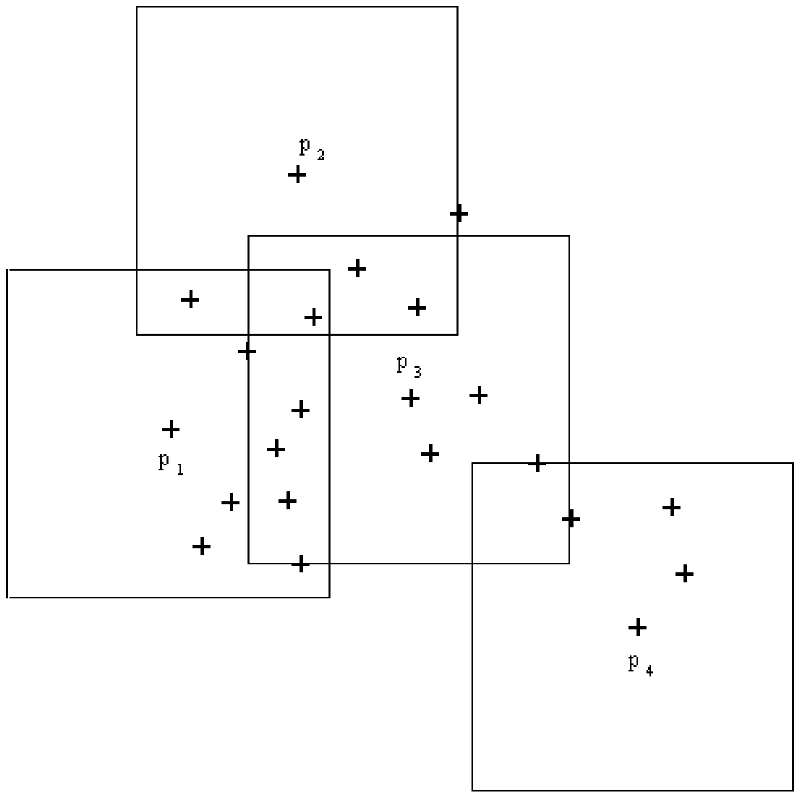


Fig. 9-31 A Feasible Solution of the Rectilinear 5-center Problem

time complexity: $O(n^2 \log n)$

Step 1: $O(n)$

Step 2: $O(1)$

$$\left.\begin{array}{l}\text{Step 3:}\\ \quad\int\\ \text{Step 5:}\end{array}\right\} \; O(\log n) * O(mn) = O(n^2 \log n)$$

The approximation algorithm is of error ratio 2.

If r is feasible, then Test(m, P, r) returns a feasible solution of size 2r.

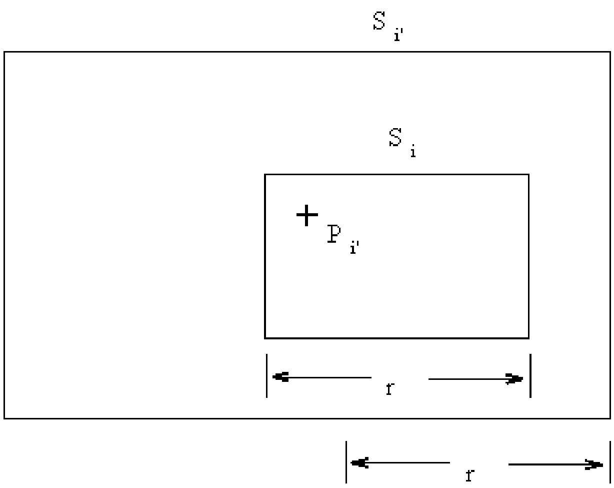


Fig. 9-32 The Explanation of $S_i \subset S_i{}'$