14457

A SUCCESSIVE ALGORITHM FOR THE CHINESE POSTMAN PROBLEM

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Noyan Narin

June, 1991

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Associate. Prof. Mustafa Akgül(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Associate. Prof. Osman Oğuz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Associate Prof. Peter Kas

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Associate Prof. Belá Vizvari

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

A SUCCESSIVE ALGORITHM FOR THE CHINESE POSTMAN PROBLEM

Noyan Narin
M.S. in Industrial Engineering
Supervisor: Associate. Prof. Mustafa Akgül
June, 1991

The Chinese Postman Problem being one of the well known problems in combinatorial optimization has many applications in real life problems such as mail delivery, road maintenance and bus scheduling. In this thesis work, we present a successive algorithm for the solution of Chinese Postman Problem. Additionally, we present efficient data structures for the existing algorithms in literature and for the implementation of our successive algorithm.

Keywords: Chinese Postman Problem, Matching, Blossom algorithm, Successive algorithms.

ÖZET

ÇİNLİ POSTACI PROBLEMİ İÇİN SIRALI ALGORİTMA

Noyan Narin
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Mustafa Akgül
Haziran, 1991

Literatürde temel problemlerden biri olarak bilinen Çinli Postacı Probleminin gerçek hayatta da mektup dağıtımı, yol bakımı, otobüs çizelgelemesi gibi birçok uygulaması vardır. Bu çalışmada, Çinli Postacı Problemi için sıralı algoritma anlatılmıştır. Buna ek olarak, hem halahazırda literatürde var olan, hem de bizim geliştirdiğimiz algoritma için uygun olacak veri yapısı sunulmuştur.

Anahtar Kelimeler: Çinli Postacı Problemi, eşleme, blossom algoritmaları, sıralı algoritmalar.

To my family,

ACKNOWLEDGMENT

I am grateful to Assoc. Prof. Mustafa Akgül, for his excellent supervision, guidance, encouragement and patience during the development of this thesis. I am indebted to the members of the thesis committee: Assoc. Prof. Osman Oğuz, Assoc. Prof. Belá Vizvari and Assoc. Prof. Peter Kas for their advice.

I also wish to express my appreciation to Oya Ekin for her invaluable help.

My special thanks are due to Yasemin Eken for her morale support and encouragement.

Contents

Chapter 1

INTRODUCTION

The Chinese Postman Problem (CPP) is one of the oldest problems in combinatorial optimization. It was first introduced by Kwan Mei Ko in 1962 as the problem faced by a postman who had to deliver mails along edges over a given network before returning to his postoffice.

The importance of the problem arises from the fact that many real-life problems such as mail delivery, garbage collection, street cleaning, road maintenance, school bus scheduling, and many others can be modelled as a variation of CPP. Some of the application areas of the problem involves a generalization of it, that is, there is not just one postman but many, and the problem is to assign routes to the postmen with fewest possible number of postmen none of whom has too long tour. The CPP has many applications in VLSI optimization and provides solution to planar multicommodity flows and max cut problems as well.

In this study we present a successive algorithm that can ease the application of the problem to some variations. The successive algorithm is the first step towards parallelization. Beside this we supply an efficient data structure for both the existing algorithms and the new successive algorithm.

Now we give the outline of the thesis. In Chapter 2 we review the related literature. This chapter concentrates on Edmonds' pionering work on this problem. We also describe the primal-dual algorithm by Edmonds and Johnson, and the primal algorithm by Barahona in that chapter. In Chapter 3, we provide efficient data structures which enable us to implement both the existing algorithms and the new successive algorithm with $O(|E||V|\log(|V|))$ time bound. The successive algorithm is presented in Chapter 4. We provide an example in Chapter 5 and make some concluding remarks in Chapter 6.

Chapter 2

LITERATURE REVIEW

The Chinese Postman Problem which was first introduced by Mei-Ko Kwan [21] in 1962 is to find the minimum postman tour over an undirected, connected graph, $G = (V, E)$ with non-negative edge weights, where the postman tour can be defined as a closed walk that traverses each edge at least once.

By simple intuitional appeal, if the degree of all nodes are even then the minimum postman tour is that which traverses each edge exactly once, i.e., wherever the postoffice is, the postman delivers all mails and turn back to his postoffice; he traverses all the edges and never traverses any edge more than once. In such a case, we call the graph **"Eulerian"**. So for Eulerian graphs the solution is trivial.

The problem can be viewed as the transformation of the given graph $G = (V, E)$, with nonnegative edge weights c_e, into an Eulerian graph by duplicating some of its edges having minimum cost and then finding an Euler tour. In this work we do not deal with finding the Euler tour. In [15, 25], two algorithms have been given to find such a tour.

Definition 2.1 *Given $G = (V, E)$ and $T \subseteq V, \mid T \mid$ even, $E' \subseteq E$ is a $T - join$ if in the subgraph $G' = (V, E')$, the degree of any node v is odd if and only if $v \in T$.*

Lemma 2.1 *If $T = \{u : \ u \ is \ odd \ degree, \ u \in V\}$ for $G = (V, E)$ and E' is a $T - join$, then the multigraph obtained from G by duplicating the edges in E' is Eulerian.*

Thus if T is the set of all odd degree nodes of G, then there is a one-to-one correspondence between $T - joins$ in G, and Chinese Postman tours.

Definition 2.2 *An* **improving circuit** *C with respect to a $T_G - join$ E' is a circuit such that the sum of the cost the edges in $C \setminus E'$ is less than the sum of the cost of the edges in $C \cap E'$.*

Kwan's algorithm [21] proceeds by finding an arbitrary $T_G - join$ and improves by finding an improving circuit. Since $C\Delta E'$ is also a $T_G - join$, the cost of the tour can be decreased. Although his idea is appealing, he couldn't give a polynomial time algorithm to find such circuits.

In 1965 Edmonds showed the close relationship between the weighted matching and the postman problem [12]. He was also the first who dubbed "**Chinese**" to the problem in recognition of the Chinese mathematician Mei-Ko Kwan.

The first proposed method is easy. For all the pairs of odd degree nodes of G, shortest paths are computed; and using these values as edge weights of complete graph K_p where p is the number of odd degree nodes in G, a minimum weighted perfect matching problem is solved [15, 12]. If (i,j) is a matched edge in K_p, then duplicate the edges on the shortest path between i and j in the original graph G. Then the new graph obtained is eulerian. The time complexity of this method is $O(\mid V \mid^3)$, however this method does not use the advantage of the sparcity of the original graph in anyway.

In 1973 Edmonds and Johnson [15] gave a polyhedral description of the problem, also proposed a direct algorithm to solve this problem. An algorithmic proof of the polyhedral description was also provided in [15]. Using the data structures proposed by Lawler [22], the algorithm can be implemented in $O(\mid V \mid^3)$ time.

Later in 1982, Barahona adapted the primal algorithm of Cunningham and Marsh [8] for the weighted matching problem to this problem [3, 5].

For many polynomial algorithms that solve the Chinese Postman Problem [15, 3, 5, 8, 19] the essential step is "*blossom shrinking*" operation that is first presented by Edmonds [11].

In 1984, Sebö described a direct combinatorial algorithm to find a minimum cardinality $T_G - join$ through elementary improving steps in polynomial time by generalizing the Kwan's improving circuits [27]. His starting point is the Lovász's interpretation of the matching algorithm [23]. He proved "*structure theorem*" of $T_G - join$ [28] that generalizes the Gallai-Edmonds theorem which plays the same role in Chinese Postman Problem as it does in Lovász's algorithm. Time complexity of this algorithm is $O(\mid V \mid^4)$.

In what follows we will first describe Edmonds' polyhedral description of the problem together with the primal-dual algorithm to solve it. Later we will discuss Barahona's primal algorithm.

2.1 Polyhedral Description of CPP and the Primal-Dual Algorithm

Given a graph $G = (V, E)$ and a set of real weights $c_e : c_e \geq 0, \forall e \in E$, the problem can be formulated as

$$
\begin{array}{llrcl}
P1: & \min & \sum_{e \in E} c_e x_e & & \\
 & s.t. & & & \\
(2.1) & & \sum_{e \in E} a_{ve}(1 + x_e) & \equiv & 0, \quad (\text{mod}2)\ \forall v \in V \\
(2.2) & & x_e & \geq & 0, \quad \forall e \in E \\
(2.3) & & x_e & is & integer, \ \forall e \in E
\end{array}
$$

where

$[a_{ve}]$ is the node-edge incidence matrix of graph G

x_e is the number of extra times that edge e is traversed.

Let b_v for all nodes of G be

$$
b_v = \begin{cases} 0, & \text{if the node v is incident to even number of edges.} \\ 1, & \text{if the node v is incident to odd number of edges.} \end{cases}
$$

Then we can write the congruence equality (2.1) as,

$$
(2.1') \quad \sum_{e \in E} a_{ve} x_e - 2w_v \ = \ b_v, \quad \forall v \in V.
$$

The variable w_v can be thought of nonnegative, integer variable that corresponds to loops constructed for every node where loop is an edge having two ends meeting the same node. Since w_v corresponds to distinct loops for all nodes, we can extend the coefficient matrix $[a_{ve}]$ to include the coefficients of w_v by adding n columns which are all zero except -2 at the $i-th$ row for $i-th$ node.

Resulting coefficient matrix consists of 0,1 and -2 and for all columns the sum of the absolute values of coefficient matrix is less than 2, i.e.,

$$
(2.4) \quad \sum_{v \in V} |\ a_{ve}\ | \ \leq \ 2, \quad \forall e \in E, \text{ also for all e corresponds to loops.}
$$

Indeed (2.1′), (2.2), (2.3) with predefined cost function is a general matching problem [14], knowing (2.4) and using the polyhedral theorem of Edmonds and Johnson [13, 14] we can write the polyhedral description of the CPP. Prior to this, we provide some necessary definitions.

Definition 2.3 *A* **Blossom** *B is a subset of node set V such that $\sum_{v \in B} b_v$ is odd, i.e., B is a subset of V containing odd number of odd degree nodes and any number of even degree nodes. In particular a single node with odd degree is a blossom. Let $\mathcal{B}$ be the set of all blossoms.*

Definition 2.4 *For any subset $S \subseteq V$, the* **coboundary** of S, *$\delta(S)$, is the set of all edges having exactly one end in S. If $S = \{v\}$ consists of a single node, then we will write shortly $\delta(v)$ instead of $\delta(\{v\})$..*

Definition 2.5 *For any subset $S \subseteq V$,* **S induced set of edges**, *$\gamma(S)$, is the set of all edges having both ends in S.*

Let for any subset S of V

$$x(\delta(S)) = \sum_{e \in \delta(S)} c_e x_e$$

$$\begin{array}{llrcll} P2: & \min & \sum_{e \in E} x_e & & & \\ & s.t. & & & & \\ (2.1') & & x(\delta(v)) - 2w_v & = & b_v, & \forall v \in V. \\ (2.5) & & x(\delta(B)) & \geq & 1, & \forall B \in \mathcal{B}. \\ (2.2) & & x_e & \geq & 0, & \forall e \in E. \\ (2.6) & & w_v & \geq & 0, & \forall v \in V. \end{array}$$

Now the aim is to get rid of w_v. From (2.1′) we can say

$$w_v \in \{-\frac{1}{2}, 0, \frac{1}{2}, 1, ...\}$$

Since w_v is integer then it is obvious that w_v is nonnegative. So we can drop (2.6) from the constraint set of the polyhedral description of CPP. Also, once we find a solution $\bar{\mathbf{x}}$ satisfying (2.5) and (2.2), (2.1′) is used in order to determine w_v for all nodes v. Thus in polyhedral description we don't need (2.1′).

Consequently, the claim is that, the optimum solution of P3 is optimum for P2 and hence for P1 where

$$\begin{array}{llrcll} P3: & & \min & \sum_{e\in E} c_e x_e & & \\ & & s.t. & & & \\ & (2.5) & x(\delta(B)) & \geq & 1, & \forall B \in \mathcal{B}. \\ & (2.2) & x_e & \geq & 0, & \forall e \in E. \end{array}$$

The algorithmic proof for this claim is given in [15].

The dual of P3 is

$$\begin{array}{llrcll} D3: & & \max & \sum_{B\in\mathcal{B}} y_B & & \\ & & s.t. & & & \\ & (2.7) & \sum_{B\in\mathcal{B}}\{y_B : \ e \in \delta(B)\} & \leq & c_e, & \forall e \in E. \\ & (2.8) & y_B & \geq & 0, & \forall B \in \mathcal{B}. \end{array}$$

For an optimal pair ($\bar{\mathrm{x}}$,$\bar{\mathrm{y}}$), complementarary slackness conditions should be satisfied; that is,

$$CS \quad (2.9) \qquad \bar{\mathrm{x}}_e(c_e - \sum_{B\in\mathcal{B}}\{\bar{\mathrm{y}}_B : \ e \in \delta(B)\}) = 0, \quad \forall e \in E.$$

$$(2.10) \qquad \bar{\mathrm{y}}_B(1 - \sum_{e\in E}\{\bar{\mathrm{x}}_e : \ e \in \delta(B)\}) = 0, \quad \forall B \in \mathcal{B}.$$

Let $c'_e{=}c_e - \sum\{y_B : \ e \in \delta(B)\}$ be the **reduced cost** of edge e.

The primal-dual algorithm for CPP presented in [15, 5] is an adaptation of Edmonds' weighted matching algorithm. Major difference arises from the shrinking operation because in Chinese Postman Problem, there are four types of blossoms that will be described in the course of the algorithm.

Since the algorithm is very similar to matching algorithms we will use the notations that are common almost to all matching algorithms.

Definition 2.6 *Given an undirected graph $G = (V, E)$, a* **matching** *$M \subseteq E$ is a subset of edges no two of which are incident to a common vertex. Clearly, for the case $c_e \geq 0$, $\forall e \in E$, the optimum solution to P3 is a binary vector.*

The edge with $x_e = 1$ is called a **matched** edge and it means the edge e should be duplicated in order to make the graph Eulerian, and the edge with $x_e = 0$ is an **unmatched** edge.

The primal-dual algorithm works on the so called **surface graph** G_s which consists of the blossoms and even degree nodes that are not contained in any blossom, i.e., in G_s any blossom is maximal.

Initially surface graph is equal to G, except all odd degree nodes correspond to **pseudonodes(blossoms)**. Even degree nodes are called **even nodes**. Initial dual variables associated with pseudonodes are zero, and even nodes do not have dual variables. At the beginning of the algorithm, x_e values are zero for all edges, i.e., M is empty.

Definition 2.7 *If a blossom is incident to no matched edge then it is called a* **free(exposed)** *blossom. Otherwise it is called* **saturated**. *For a free blossom for all $e \in \delta(B)$, we have $x_e = 0$.*

Definition 2.8 *An* **alternating path** *with respect to M is a simple path whose edges are alternately in M and not in M. An* **augmenting path** *is an alternating path whose end nodes are free. An* **alternating cycle** *is an alternating path with the same starting and end point. Obviously an alternating cycle has even number of edges.*

Definition 2.9 *An* **alternating tree** *is a tree $T = (V(T), (E(T))$ rooted at free node r with the properties that the paths from r to each vertex in T are alternating paths and $E(T) \cap M$ is a perfect matching with respect to $V(T) \setminus \{r\}$.*

Theorem 2.1 (Berge) *A matching M in G is* **maximum** *if and only if there is no augmenting path with respect to M.*

The algorithm consists of successive stages in each of which it tries to find an augmenting path between free blossoms. Each stage ends up with an augmentation, so at the end of each stage the number of free blossoms in the surface graph is decreased by two.

Any stage of the algorithm begins with a dual feasible solution y and corresponding to that y let $G_s(y)$ be the **equality subgraph** of G_s. That is $G_s(y)$ consists of all nodes of G_s and those edges of G_s for which the reduced costs are zero. Moreover, the matching M at hand at the beginning of the stage contains only the edges of $G_s(y)$ so that M and y fulfill the complementary slackness conditions $((2,9),(2.10))$. That is to say, in equality subgraph every matched edge meets at least one pseudonode and every pseudonode is incident to at most one matched edge. Only the dual variables corresponding to saturated pseudonodes can be positive and the dual variables corresponding to free pseudonodes are all zero. If the matching M is perfect in $G_s(y)$, i.e., if there is no free blossoms in $G_s(y)$ then we are done, we found the optimum solution.

Otherwise, starting from the free pseudonodes the algorithm grows alternating trees rooted at those distinct, unmatched pseudonodes by using the edges in $G_s(y)$. We call the collection of those alternating trees as **planted forest**. The nodes on the planted forest are all pseudonodes. The roots of the alternating trees are labelled by + and all the other pseudonodes in planted forest are alternately labelled by + and −. For any pseudonode p, + label indicates the existence of an even length alternating path from p to the root of the alternating tree containing p, whereas − label indicates the existence of an odd length alternating path from p to the root of the alternating tree containing p. The pseudonodes that are not contained in the planted forest are indicated by 0. Each − labelled pseudonode of the planted forest is incident to two edges one of which is a matched edge in the planted forest. The + labelled pseudonodes are incident to any number of unmatched edges and to one matched edge except the roots of the alternating trees. Number of + labelled pseudonodes in planted forest is greater than the number of − labelled pseudonodes. Thus, in order to result in an augmentation, we should add some edges of G_s into the equality subgraph by changing the dual variables.

The dual changes should be done in a way that, the dual feasibility is not violated and there is no edge that leaves the planted forest even there is none that is adjoined to it. To satisfy these we employ a special procedure FIND-MIN. FIND-MIN determines the maximum possible dual variable change. After the dual update, four cases may occur.

a. An edge of G_s, which links an unlabelled pseudonode to a + labelled pseudonode of an alternating tree, is adjoined to the equality subgraph. In that case we grow the tree.

b. i- An edge of G_s, which joins two + labelled pseudonodes of an alternating tree, is adjoined to the equality subgraph.

   ii- An edge of G_s that joins an even node to a + labelled pseudonode of G_s is adjoined.

   In both case, we employ a special SHRINK operation to preserve the properties of the planted forest and not to ignore the possibility of having an augmenting path that passes through a pseudonode that has − label before SHRINK. Given an odd set of nodes $S \subseteq V_s$, we shrink S by updating the surface graph,

$$G'_s = (V'_s, E'_s) \qquad \textit{where} \quad \begin{aligned} V'_s &= V_s \setminus S \cup \{v'\} \\ E'_s &= E_s \setminus \{(i,j) : \ (i,j) \in \gamma(S)\} \end{aligned}$$

   and v' is called the **pseudonode induced by S.**

These structures were first investigated by Edmonds for CPP. Their importance comes from the fact that whenever a perfect matching in G_s can be found it can be extended to the solution of the CPP over the original graph G. The odd sets that are shrunk are nested, and whenever the odd set S is saturated by an edge in $\delta(S)$, the pseudonodes in S can be saturated by using the edges in $\gamma(S)$.

c. No edge is adjoined to the equality subgraph but the dual variable corresponding to a $-$ labelled pseudonode drops to zero. In that case we call expand.

   As we mention before, in any stage, blossom algorithm looks for a minimum cost augmenting path with respect to given matching between free pseudonodes over the current surface graph G_s. Needless to say that the minimum cost augmenting path in G may not be induced by any augmenting path in G_s. In such cases, the algorithm detects it and overcome this situation by expanding certain pseudonodes and updating the surface graph.

d. An edge of G_s that meets two $+$ labelled pseudonodes on different alternating trees is adjoined to the equality subgraph. It means we find an augmenting path P. Then the matching M is changed by reversing the role of matched and unmatched edges of P. This will be the end of a stage and a new stage begins with the augmented M and the updated dual vector y. They satisfy the complementary slackness and the additional conditions.

Definition 2.10 *The nodes in the current surface graph, i.e., the nodes that are not contained in any pseudonode are called* **exterior nodes**, *and all nodes that are contained in a pseudonode are called* **interior nodes**. *For any node v,* **b(v)** *indicates the exterior node that contains v, and for any exterior node k,* **REAL(k)** *denotes the set of* **real nodes**, *i.e., the nodes of the original graph G.*

Knowing the necessary definitions and the conceptual description of the algorithm, the more compact form of the generic primal-dual algorithm can be described. Prior to this we will explain the procedures that are used in the algorithm.

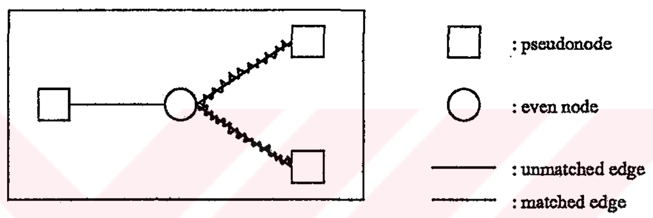
```
FIND-MIN($\Delta$)
begin
  $\Delta_1 = \min\{c'_{ij} : (i,j) \in E_s$, b(i) is + labelled (exterior) pseudonode,
                         b(j) is unlabelled$\}$
    Set $\rho_1 = (i,j)$ where (i,j) is the edge satisfying the minimum.
  $\Delta_2 = \frac{1}{2}\min\{c'_{ij} : (i,j) \in E_s$, b(i) and b(j) are + labelled (exterior) pseudonodes$\}$
    Set $\rho_2 = (i,j)$ where (i,j) is the edge satisfying the minimum.
  $\Delta_3 = \min\{y_k :$ k is $-$ labelled (exterior) pseudonode$\}$
    Set $\rho_3 = k$, $k$ is the exterior pseudonode satisfying the minimum.
```

Set $\Delta = \min\{\Delta_1, \Delta_2, \Delta_3\}$
end{**FIND-MIN**}

This procedure guarantees that after the dual variable changes, the dual feasibility is not violated. In addition it detects that there may be a shortest path that is not induced by the current surface graph, ($\Delta = \Delta_3$).

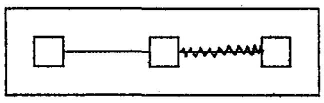
Before we continue with the description of the procedures, we will give the definitions of the blossoms that are frequently used.

Type-1 blossom is an odd set that contains one even node and one unmatched edge. The unmatched edge meets the even node and a pseudonode. The other pseudonodes in the blossom, if any, are adjoined to the even node through matched edges.



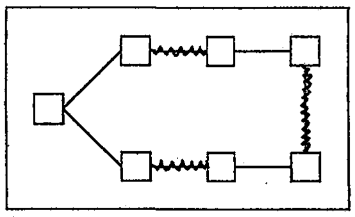
type-1 pseudonode

Type-2 blossom contains three pseudonodes. One of the pseudonodes is adjacent to the other two through a matched and an unmatched edge. The dual variable corresponding to that pseudonode is zero.



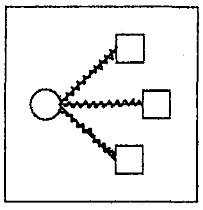
type-2 pseudonode

Type-3 blossom is an odd cycle. All the nodes in it are pseudonodes. One of them is incident to two unmatched edges in the blossom, while the others are incident to one matched, one unmatched edges.



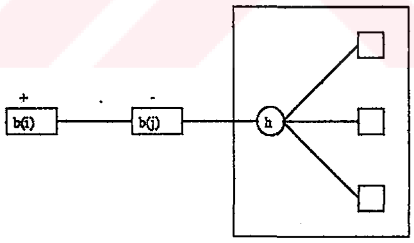
type-3 pseudonode

Type-4 blossom contains one even node and an odd number of pseudonodes that are adjacent to the even node through matched edges.

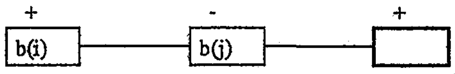

type-4 pseudonode

It is obvious that, there are odd number of pseudonodes of any type in any type of blossom.

```
GROW(i,j)            /* b(i) is + labelled, b(j) is unlabelled pseudonode
begin
  Label b(j) by -
  if the mate of b(j) is an even node h
    then begin
        Set $S = \{h\} \cup \{$mates of h other than b(j)$\}$
        SHRINK(S)
```



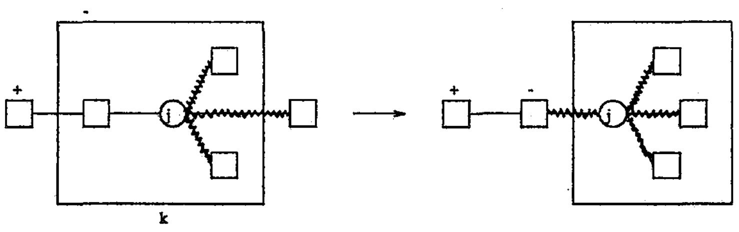
```
    end
    else Label the mate of b(j) by +
```



```
end{GROW}
```

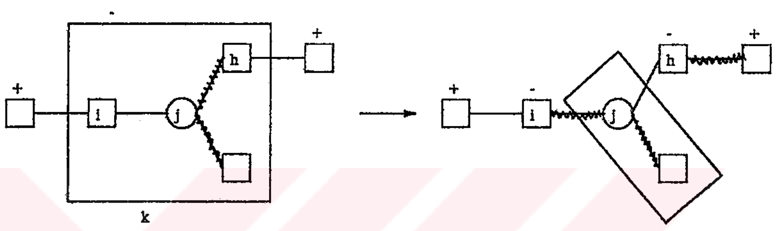
```
SHRINK(S)            /* S is a blossom.
begin
    Shrink S into pseudonode p          /* Replace set $S \subset V$ with p.
    Update $G_s$
    Set $y_p = 0$
    for all exterior node h in S
        begin
            Set b(h)=p
                $REAL(p) = REAL(p) \cup REAL(h)$
        end
    Label p by +
end{SHRINK}
```

```
EXPAND(k)            /* k is a – labelled pseudonode.
begin
    if k is type-1 pseudonode
        then begin
                Examine the matched edge at the coboundary of k
                if it emanates from an even node j in k
                    then begin
                        Unshrink k
                        Set $S = \{j\} \cup \{mates\ of\ j\} \cup \{mate\ of\ k\}$
                        Change the status of the unmatched edge in $\gamma(k)$
                        Label the exterior pseudonode in k but not in S by –
                        Update $G_s$
                        SHRINK(S)
```

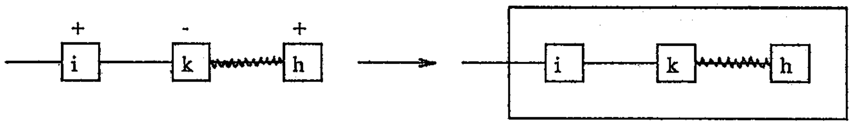


```
        end
```

```
                else begin
                    Unshrink k
                    Update $G_s$
                    Let i is the node from which, the unmatched edge in $\delta(k)$ is emanated.
                         h is the node from which, the matched edge in $\delta(k)$ is emanated.
                         j is the even node in k.
                    Swap the matched and unmatched edges over the path from h to i
                    Label i,h with $-$
                    Set $S = \{j\} \cup \{mates\ of\ j\ other\ than\ h\}$
                    SHRINK(S)
```

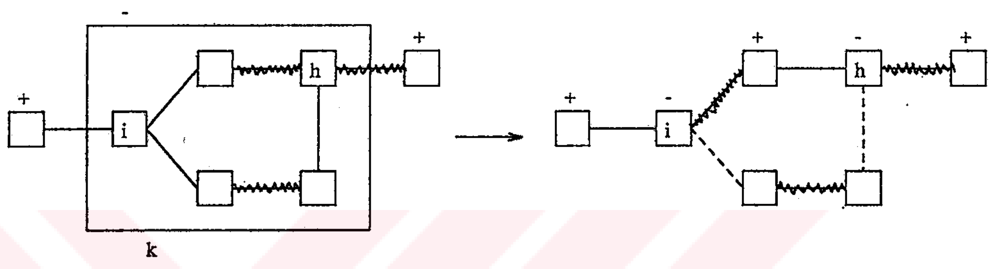


```
                end
        end
    if k is type-2 pseudonode
        then begin
                Let i and h are the two + labelled exterior pseudonodes adjacent to k.
                Set $S = \{i, k, h\}$
                SHRINK(S)
```

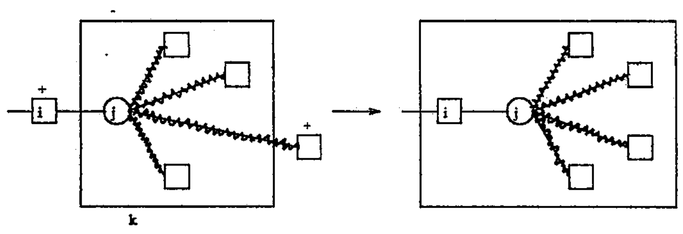


```
        end
```

```
if k is type-3 pseudonode
    then begin
        Unshrink k
        Update $G_s$
        Let i is the node from which, the unmatched edge in $\delta(k)$ is emanated.
             h is the node from which, the matched edge in $\delta(k)$ is emanated.
        Swap the matched and unmatched edges over the even path from h to i
        Label the nodes over the even path from h to i alternately by $-$ and $+$.
        Delete the remaining unlabelled part of k from the planted forest.
```

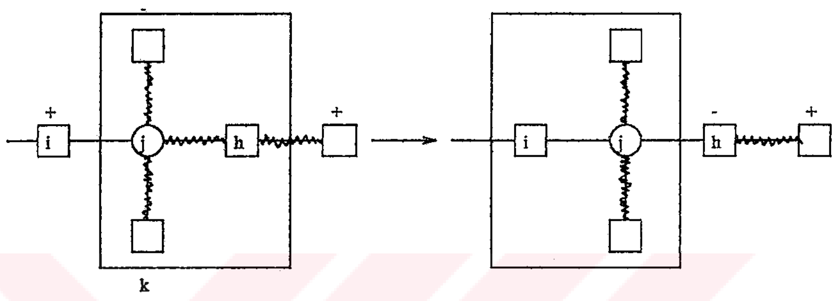


```
    end
if k is type-4 pseudonode
    then begin
        if the matched edge in $\delta(k)$ is emanated from the even node j
            then begin
                Unshrink k
                Update $G_s$
                Set $S = \{i, j\} \cup \{mates\ of\ j\}$
                SHRINK(S)
```



```
            end
```

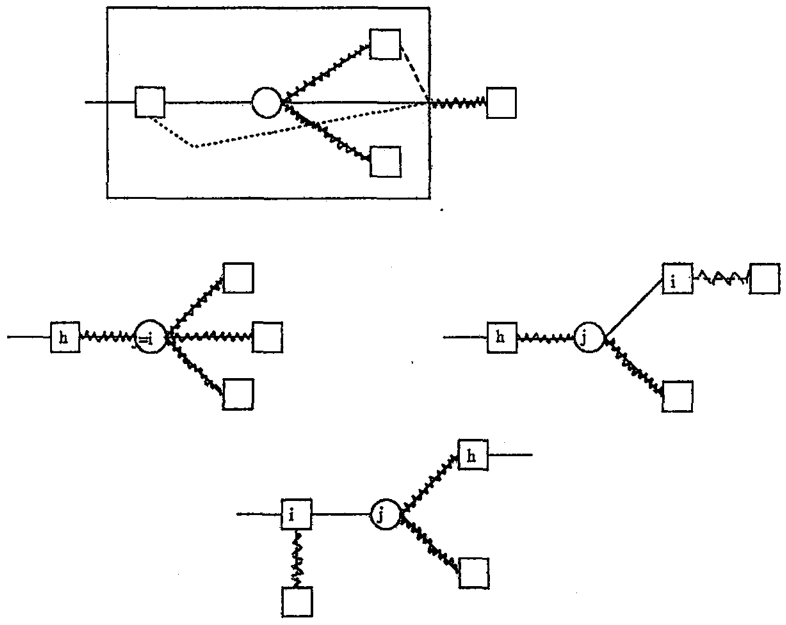
```
                  else begin
                      Let h is the pseudonode that the matched edge in $\delta(k)$ is emanated from.
                      Unshrink k
                      Update $G_s$
                      Change the status of the edge between j and h
                      Label h by $-$
                      Set $S = \{i, j\} \cup \{mates\ of\ j\}$
                      SHRINK(S)
```



```
                  end
            end
end{EXPAND}
```

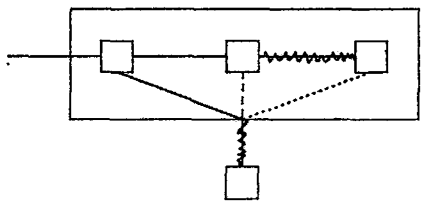
```
AUGMENT(i,j)            /* b(i) and b(j) are + labelled pseudonodes and
begin                   /* are in different alternating trees.
    Let $r_1$ is the root of the alternating tree containing b(i).
         $r_2$ is the root of the alternating tree containing b(j).
         $P_1$ is the path from $r_1$ to b(i).
         $P_2$ is the path from $r_2$ to b(i).
    Set $AP = P_1 \cup \{(i, j)\} \cup P_2$
    for all edge $e \in AP$
        Set $x_e = 1 - x_e$
    Delete the two trees from the planted forest and
    remove the labels of the pseudonodes on them
end{AUGMENT}
```

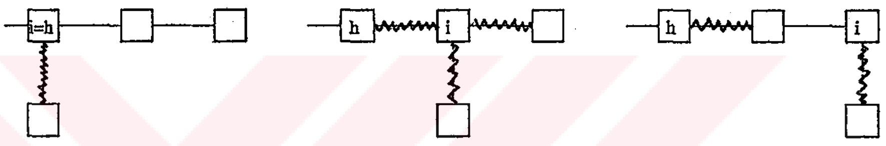
```
RECOVER(k)
begin
  if k is a real node
    then Do Nothing
  else begin
      Let i is the node in k from which the matched edge in $\delta$(k) is emanated
          h is the node in k from which the unmatched edge in $\delta$(k) is emanated
      if k is type-1 blossom
        then begin
            if i is even node
              then begin
                  Unshrink k
                  Change the status of unmatched edge in k
                  Update $G_s$
              end
              else begin
              Unshrink k
                  Let j is the even node in k
                  if the edge between i and j is matched
                    then Swap the matched and unmatched edges over the path
                            from i to h
                  Update $G_s$
              end
```



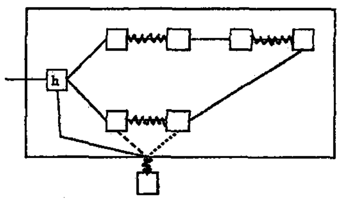
```
        end
```

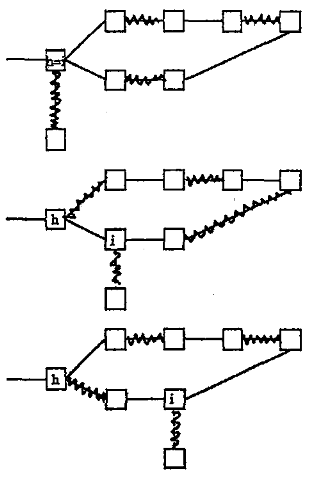
```
if k is type-2 blossom
  then begin
       Unshrink k
       if i and h are the same node
          then Do Nothing
          else Swap the matched and unmatched edges over the path from i to h
       Update $G_s$
```





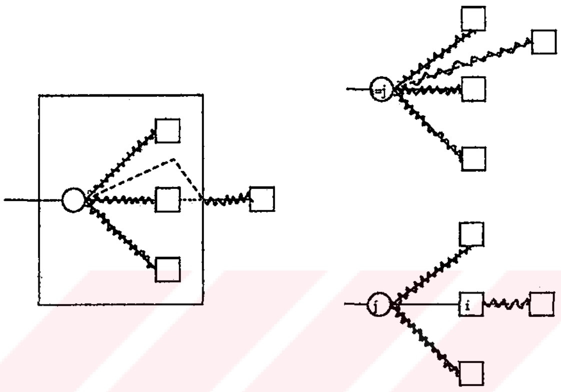
```
    end
if k is type-3 blossom
    then begin
         Unshrink k
         if i and h are the same node
            then Do Nothing
            else Swap the matched and unmatched edges over the path from i to
                    h such that, the first edge is matched
    Update $G_s$
```





```
    end
```

```
        if k is type-4 blossom
            then begin
                    Unshrink k
                    Let j is the even node in k
                    if i and j are the same node
                        then Do Nothing
                        else Change the status of the matched edge between i and j
                    Update $G_s$
```

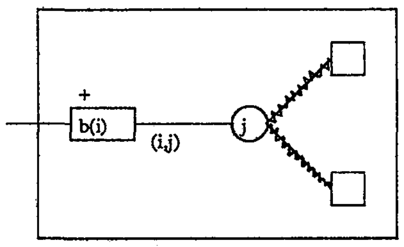


```
            end
        end
end{RECOVER}
```

MAIN ALGORITHM

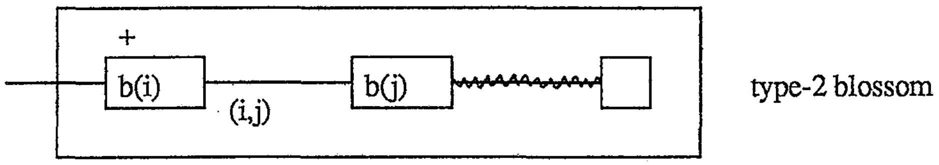
```
    Initialization
    Convert $G$ to $G_s$
    for all pseudonodes k in $G_s$
        begin
            Set $y_k = 0$
                $l_k = +$
                $REAL(k) = k$
        end
    for all edges e in $G_s$
        begin
            Set $x_e = 0$
                $c'_e = c_e$
        end
    for all nodes n in $G_s$
        Set b(n)=n
```

```
While $\exists$ a + labelled pseudonode in planted forest
begin          /* dual update
   FIND-MIN($\Delta$)
   for all + labelled exterior node k in $G_s$
      begin
         Set $y_k = y_k + \Delta$
         for each edge e in $\delta(k)$
            Set $c'_e = c'_e - \Delta$
      end
   for all − labelled exterior node k in $G_s$
      begin
         Set $y_k = y_k - \Delta$
         for each edge e in $\delta(k)$
            Set $c'_e = c'_e + \Delta$
      end
end
if $\Delta = \Delta_1$
   then begin
         Add the edge $\rho_1 = (i, j)$ to planted forest
         if b(j) is even node
            then begin
                  Set $S = \{b(i), j\} \cup \{mates\ of\ j\}$
                  SHRINK(S)
```

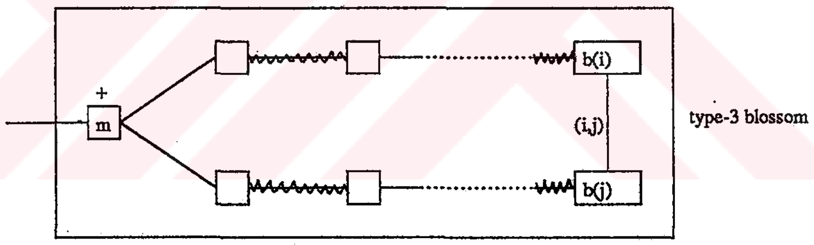


type-1 blossom

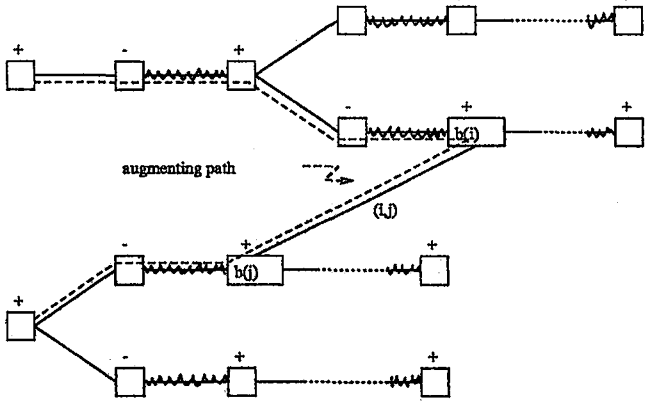
```
            end
            else begin
                  if dual variable corresponding to b(j) is zero
                     then begin
                           Set $S = \{b(i), b(j), mate\ of\ b(j)\}$
                           SHRINK(S)
```



```
                    end
                    else GROW($\rho_1$)
    end
if $\Delta = \Delta_2$
    then begin
        Add the edge $\rho_2 = (i,j)$ to planted forest
        Backtrack from b(i) and b(j) over planted forest
        if a common exterior pseudonode m is reached
            then begin
                Set $S$ = {the exterior pseudonodes over the path from m to b(i)}
                          $\cup${the exterior pseudonodes over the path from m to b(j)}
                SHRINK(S)
```



```
            end
            else AUGMENT($\rho_2$)
```



```
    end
if $\Delta = \Delta_3$
```

```
        then EXPAND($\rho_3$)
    end{While}

    for all exterior pseudonodes k in $G_s$
        RECOVER(k)

    end{MAIN ALGORITHM}
```

Let us study the complexity of a stage first. In any stage a newly formed pseudonode is given a + label and it can never be expanded during the same stage. Thus, Expand and Shrink are each called at most $O(| V |)$ times, so does Grow. This means Find-Min is called at most $O(| V |)$ times. The dual updates and the following reduced cost updates can be done in $O(| E |)$ time. The Shrink and Expand operations can be implemented in $O(| V |)$ time. The Grow operation needs constant time. Hence any stage of the algorithm can be accomplished in $O(| V || E |)$ time. Since there are at most $\frac{|V|}{2}$ stages, this results in $O(| V |^2| E |)$ total work requirement of the algorithm.

2.2 The Primal Algorithm

This algorithm is the adaptation of the primal matching algorithm by Cunningham and Marsh. It uses the same polyhedral description of the problem and uses similar data structures as the primal-dual algorithm does.

The primal algorithm begins with any primal feasible solution and try to improve the solution over negative cycles until the optimal solution is obtained. The algorithm works on the surface graph. The surface graph is the same with the previously defined except it contains artificial edges between the odd degree nodes.

Any primal feasible solution corresponds to a perfect matching over the exterior pseudonodes of surface graph. If C is an alternating cycle with respect to the matching M, then $M \oplus C$ is another perfect matching over the same nodes.

If we define c(M) as the cost of the set M $\subset E$

$$c(M) = \sum_{e \in M} c'_e$$

then

$$c(C \oplus M) = c(M) + c(C \setminus M) - c(C \cap M)$$

Definition 2.11 *A* **negative cycle C** *with respect to M is an alternating cycle, for which*

$$[c(C \setminus M) - c(C \cap M)]$$

is negative.

Theorem 2.2 *A matching M is optimal if and only if M does not admit negative cycle.*

Like primal-dual algorithm, primal algorithm consists of successive stages. Each stage begins by choosing an edge, in the current surface graph, with negative reduced cost and tries to find an alternating path with total zero reduced cost between the ends of the edge. Finding such a path means there is a negative cycle. Once such a cycle is found, we make a primal change. Any stage ends with a primal change and the reduced cost of at least one more edge becomes nonnegative.

At the beginning of any stage, we have a primal feasible solution x, a dual solution y and the corresponding G_s at hand. The equality subgraph of G_s is as defined before. The primal feasible solution x corresponds to a perfect matching M in equality subgraph $G_s(y)$. If the dual solution y is feasible, then it means we found optimum. Otherwise, there is an edge in surface graph with a negative reduced cost. At least one of the ends of that edge in G_s would be a pseudonode. We label that pseudonode by $-$ and its mate by $+$. Rooted at that $+$ labelled pseudonode, the algorithm grows an alternating tree until the other end of the edge is labelled by $+$. When the other end of the edge is labelled by $+$, primal change and the following mini-dual change are done in order to make the reduced cost of the edge non-negative. Once the reduced cost of the edge becomes non-negative, it remains non-negative. The stage ends after the primal change and the following mini-dual change. At the end of the stage, we have an updated primal feasible solution x and updated y. Moreover the number of infeasibility in y decreases at least by one.

To get an initial feasible solution, one can put artificial edges between the pairs of pseudonodes.The cost of these edges are equal to the cost of the path between the pairs of the pseudonodes, and dual variables of these pseudonodes are set to the half of the cost of the path.

The primal algorithm uses the same procedures as the primal-dual algorithm does. The only difference is in the FIND-MIN procedure. For the primal algorithm, there may be some edges that have negative reduced cost in the surface graph, however the algorithm uses the edges with non-negative reduced cost to add the alternating tree. Thus there is a slight difference in FIND-MIN for this algorithm.

Now, we will first describe the procedure FIND-MIN and then the main algorithm.

```
FIND-MIN($\Delta$)
begin
   $\Delta_1 = \min\{c'_{ij} : c_{ij} \geq 0 \; (i,j) \in E_s,$ b(i) is + labelled (exterior) pseudonode,
                                   b(j) is unlabelled$\}$
      Set $\rho_1 = (i,j)$ where (i,j) is the edge satisfying the minimum.
   $\Delta_2 = \frac{1}{2}\min\{c'_{ij} : c_{ij} \geq 0 \; (i,j) \in E_s,$ b(i) and b(j) are + labelled (exterior) pseudonodes$\}$
      Set $\rho_2 = (i,j)$ where (i,j) is the edge satisfying the minimum.
   $\Delta_3 = \min\{y_k :$ k is - labelled (exterior) pseudonode$\}$
      Set $\rho_3 = k$ k is the exterior pseudonode satisfying the minimum.
   Set $\Delta = \min\{\Delta_1, \Delta_2, \Delta_3\}$
end{FIND-MIN}
```

```
MAIN ALGORITHM
begin
   Initialization
   begin
      Start with any primal feasible solution x and corresponding dual solution y
      Convert $G$ to $G_s$
   While $\exists$ an edge e in $G_s$ such that $c'_e < 0$
   begin
      Choose an edge e=(i,j) such that

                        $c'_{ij} < 0$ and $c'_{ij} = \min\{c'_e : e \in \delta(b(i))\}$

      Set k=b(i)          /* b(i) is a pseudonode in $G_s$
      Let h is the mate of k
      Label k by $-$ and h by $+$
      Set continue=true
      while (continue) and (the other end of e is not labelled by $+$)
      begin
         if b(i) is a real node and $y_{b(i)}$ is zero
            then begin
                  Remove all the labels
                  Go to the other end of e
                  Set k=b(j)          /* b(j) is a pseudonode in $G_s$
                  Let h is the mate of k
                  Label k by $-$ and h by $+$
            end
```

```
    begin            /* dual update
      FIND-MIN($\Delta$)
      Set $\Delta = \min\{\Delta, -c'_e\}$
      for each + labelled exterior pseudonode n in $G_s$
        begin
          Set $y_n = y_n + \Delta$
          for each edge $f \in \delta(n)$
            Set $c'_f = c'_f - \Delta$
        end
      for each $-$ labelled exterior pseudonode n in $G_s$
        begin
          Set $y_n = y_n - \Delta$
          for each edge $f \in \delta(n)$
            Set $c'_f = c'_f + \Delta$
        end
    end
    if $\Delta = \Delta_1$
      then begin
          Add the edge $\rho_1$ to planted forest
          if $\rho_1 \in \delta(p)$ such that p is an exterior pseudonode, $y_p > 0$
            then GROW($\rho_1$)
            else SHRINK
      end
    if $\Delta = \Delta_2$
      then begin
          Add the edge $\rho_2$ to planted forest
          SHRINK
      end
    if $\Delta = \Delta_3$
      EXPAND($\rho_3$)
    if $\Delta = -c'_e$
      then begin
          Set continue=false
          Delete the alternating tree and
          remove the labels of the nodes on it
      end
  end{ while}
```

```
        if the other end of e is labelled by +
          then begin
              Set $x_e = 1$
              for all edge f over the path between the ends of e
                Set $x_f = 1 - x_f$
          end
        while $c'_e < 0$
        begin          /* mini-dual update
          Let k is the pseudonode such that $e \in \delta(k)$ and $y_k > 0$
          Set $\bar{\Delta} = \min\{y_k, -c'_e\}$
          for all edge $f \in \delta(k)$
            Set $c'_f = c'_f + \bar{\Delta}$
          Set $y_k = y_k - \bar{\Delta}$
          if $y_k = 0$ then RECOVER(k)
        end{while}
        Delete the alternating tree and
        remove the labels of the nodes on it
      end {While}
      for all exterior pseudonodes k in $G_s$
        RECOVER(k)
    end{MAIN ALGORITHM}
```

This algorithm has at most $| V |$ stages. In each stage Expand and Shrink are each called at most $O(| V |)$ times, so does Grow. The primal and the following mini dual changes can be executed at most in $O(| E |)$ time, and these calculations are employed only once in a stage. The dual changes and the following reduced cost updates require at most $O(| E |)$ time, and are employed at most $O(| V |)$ times in a stage. Thus the total work requirement of the algorithm is $O(| V |^2 | E |)$.

Chapter 3

An $O(|V||E|\log(|V|))$ IMPLEMENTATION

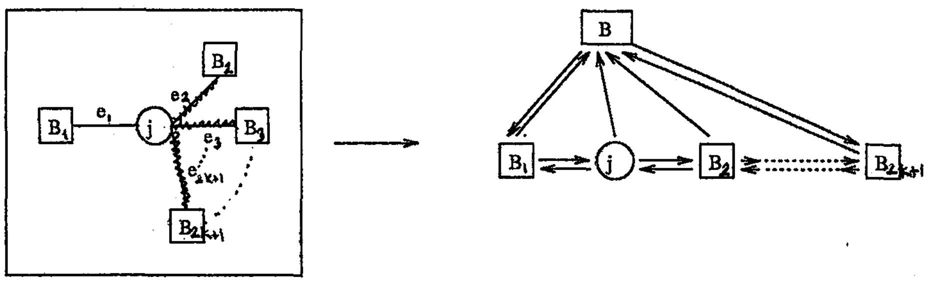
In the generic primal-dual algorithm described in the previous chapter, the most costly part is the frequent updates of the dual variables and reduced costs. Before each update it is required to examine all the edges in order to calculate the maximum possible change without violation of the set of dual constraints, (2.7),(2.8), and complementary slackness conditions, (2.9),(2.10). Thus one may examine an edge $O(|V|)$ times throughout a stage.

In order to reduce the amount of computation, we postpone the dual updates to the end of the stage, and we reduce the work for FIND-MIN by finding some invariants.

Before going into algorithmic detail we will describe the convenient data structure for this implementation.

For each blossom we define a tree that represents the structure of the blossom. Also for each blossom B, we keep the type, the base and the mate of the blossom B. Now it is better to introduce these structures for different types of blossoms.

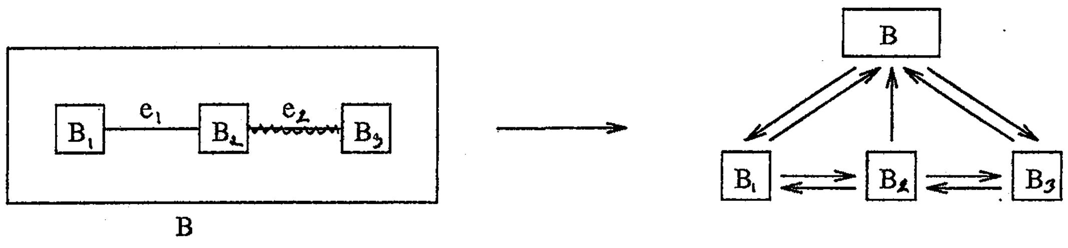
1. B is the first type of blossom. The corresponding structure tree is shown below.



The base of this blossom is B_1 and it can be represented by the list

$$\{(B_1, e_1), j, (B_i, e_i)|_{i=2}^{2k+1}\}$$

2. B is type-2 blossom.

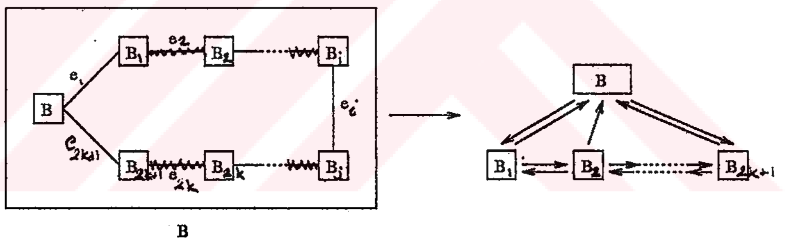


The dual variable of B_2 is zero.
The base of this type of blossom is B_1 and the corresponding list representing this structure is

$$\{(B_1, e_1), B_2, (B_3, e_2)\}$$

.

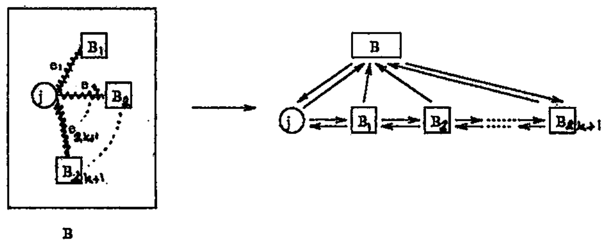
3. B is type-3 blossom, i.e., it contains an odd cycle.



The base of this blossom is B_1. The corresponding list is

$$\{(B_i, e_i)|_{i=1}^{2k+1}\}$$

4. B is the fourth type blossom, the structure tree representing this type of blossom is shown below.



The corresponding list is

$$\{j, (B_i, e_i)|_{i=1}^{2k+1}\}$$

where j is the base of the blossom.

The leaves of the structure trees corresponding to any blossom B are real nodes that are included in B. These leaves are maintained in doubly linked lists that we call REAL(B).

In surface graph any exterior pseudonode k has at most one mate. So we keep m(k)=(p,e) for each exterior pseudonode k, where the first part denotes the mate of k and the second part denotes the matched edge that k is incident to. However, original nodes might have even number of mates. Thus, for each original node j we keep a mate list $M(j) = \{(B_i, e_i)|_{i=1}2k\}$.

Any blossom in planted forest is labelled by $+(i,j)$ or $-(i,j)$. The second part of the label records the edge by which the blossom received the label. The second part of the label makes the backtracking along the alternating tree easier. We indicate the label of a pseudonode k with l_k.

In each stage, we define a variable D. At the beginning of each stage it is set to zero and it keeps the sum of dual variable changes done throuhgout the stage. Thus, at any time in a stage, it gives a lower bound on the length of the shortest augmenting path. Related wirh D, for each node i, we define d_i which is an offset for dual variable y_i. It is initially zero. Whenever the node is added to the planted forest, it is set to the current value of D.

For any pseudonode k, we define a partially updated dual variable $\bar{y}_k$. $\bar{y}_k$ is set to the current value of y_k whenever the exterior pseudonode k is added to the planted forest. At any time,

$$\begin{array}{lll} y_k = \bar{y}_k + (D - d_k) & , & \text{for exterior pseudonode k with } l_k = + \\ y_k = \bar{y}_k - (D - d_k) & , & \text{for exterior pseudonode k with } l_k = - \\ y_k = \bar{y}_k & , & \text{for all interior pseudonodes k and} \\ & & \text{exterior pseudonodes k with } l_k = 0 \end{array}$$

Since the reduced cost updates are done at the and of the stages, throughout a stage we maitain a new variable $\tilde{c}_{ij}$ for any edge $(i,j) \in E$. It is set to c'_{ij} at the beginning of the stage and is updated at the end of the stage so that $c'_{ij} = \tilde{c}_{ij}$.

Furthermore, for any node i, we maintain $\bar{e}_i$ which is explicitly defined as

$$\bar{e}_i = \sum_{j \in I(i) \setminus \{i\}} y_j$$

where I(i) is the set of pseudonodes containing i at the beginning of the stage.

Proposition 3.1 *For any $(i,j) \in E$, at any time in a stage the equality*

$$c'_{ij} = \tilde{c}_{ij} + \gamma_i + \gamma_j$$

holds, where

$$\gamma_k = \begin{cases} \bar{e}_{b(k)} & , \ \textit{if } l_{b(k)} = 0 \\ \bar{e}_{b(k)} + (D - d_{b(k)}) & , \ \textit{if } l_{b(k)} = - \\ \bar{e}_{b(k)} - (D - d_{b(k)}) & , \ \textit{if } l_{b(k)} = + \end{cases}$$

and (i,j) is not involved in a shrink operation.

For any (i,j) with $l_{b(i)} = +$, we define d_{ij} as

$$d_{ij} = \begin{cases} d_{b(i)} + \tilde{c}_{ij} + \bar{e}_{b(i)} & , \ \text{if } l_{b(j)} \neq + \\ \frac{1}{2}(\tilde{c}_{ij} + d_{b(i)} + d_{b(j)} + \bar{e}_{b(i)} + \bar{e}_j) & , \ \text{if } l_{b(j)} = + \end{cases}$$

During the generic algorithm, in any stage, we calculate $\delta_1, \delta_2, \delta_3$ where

$$\begin{aligned} \Delta_1 &= \min\{c'_{ij} : \ (i,j) \in E, \ l_{b(i)} = + \text{ and } l_{b(j)} = 0\} \\ \Delta_1 &= \tfrac{1}{2}\min\{c'_{ij} : \ (i,j) \in E, \ l_{b(i)} = +, \ l_{b(j)} = + \text{ and } b(i) \neq b(j)\} \\ \Delta_3 &= \min\{y_k : \ k \text{ is an exterior pseudonode with } l_k = -\} \end{aligned}$$

But in this implementation, we do not maintain explicitly the dual variables y_k and the reduced costs c_{ij} throughout a stage. Hence, we try to find a way for calculating these minimums by using $\bar{y}_k$ and $\tilde{c}_{ij}$.

So let's examine these three cases:

i) If $l_{b(i)} = +$ and $l_{b(j)} = 0$

$$\begin{aligned} c'_{ij} &= c_{ij} - \sum_{k \in I(i) \setminus \{b(i)\}} y_k - \sum_{k \in I(j)} y_k - y_{b(i)} \\ &= c_{ij} - \sum_{k \in I(i) \setminus \{b(i)\}} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k - [\bar{y}_{b(i)} + (D - d_{b(i)})] \\ &= d_{b(i)} + c_{ij} - \sum_{k \in I(i)} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k - D \end{aligned}$$

$$\Rightarrow \quad c'_{ij} + D = d_{b(i)} + c_{ij} - \sum_{k \in I(i)} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k$$

ii) If $l_{b(i)} = l_{b(j)} = +$

$$\begin{aligned}
c'_{ij} &= c_{ij} - \sum_{k\in I(i)\setminus\{b(i)\}} y_k - \sum_{k\in I(j)\setminus\{b(j)\}} y_k - y_{b(i)} - y_{b(j)} \\
&= c_{ij} - \sum_{k\in I(i)\setminus\{b(i)\}} \bar{y}_k - \sum_{k\in I(j)\setminus\{b(j)\}} \bar{y}_k - [\bar{y}_{b(i)} + (D - d_{b(i)})] - [\bar{y}_{b(j)} + (D - d_{b(j)})] \\
&= c_{ij} - \sum_{k\in I(i)} \bar{y}_k - \sum_{k\in I(j)} \bar{y}_k - 2D + d_{b(i)} + d_{b(j)}
\end{aligned}$$

$$\Rightarrow \ \tfrac{1}{2}c'_{ij} + D = \tfrac{1}{2}(\bar{c}_{ij} + d_{b(i)} + d_{b(j)})$$

iii) If $l_k = 0$

$$y_k = \bar{y}_k - (D - d_k)$$

$$\Rightarrow \quad y_k + D = \bar{y}_k + d_k$$

Note that

$$\begin{aligned}
d_{ij} + \bar{e}_{b(j)} &= D + c'_{ij} &&, \text{ for } (i,j)\in E \text{ with } l_{b(i)} = +,\ l_{b(j)} = 0 \\
d_{ij} &= D + \tfrac{1}{2}c'_{ij} &&, \text{ for } (i,j)\in E \text{ with } l_{b(i)} = l_{b(j)} = +,\ \ b(i)\neq b(j) \\
d_k + \bar{y}_k &= D + y_k &&, \text{ for exterior pseudonode } k \text{ with } l_k = -
\end{aligned}$$

Thus if we compute $\Delta_i + D$ rather than Δ_i we do not explicitly need the dual variables y_k and reduced cost c'_{ij} but rather $\bar{y}_k$, $\tilde{c}_{ij}$ and d_k.

Proposition 3.2 *Whenever the pseudonode k is labelled + in a stage; either it remains + labelled or it is contained in another pseudonode that has + label throughout the stage.*

Lemma 3.1 *For any $(i,j)\in E$ with $l_{b(i)} = +$. d_{ij} remains constant.*

Proof:

We look at two cases: $l_{b(j)} \neq +$ and $l_{b(j)} = +$ throughout the stage.

i) First assume that $l_{b(i)} = +$, $l_{b(j)} \neq +$. For that case:

$$d_{ij} = d_{b(i)} + c_{ij} - \sum_{k\in I(i)} \bar{y}_k - \bar{e}_j - \bar{y}_j$$

Let h be the first exterior pseudonode containing i with + label in the current stage. Then

$$d_{ij} = d_h + c_{ij} - \sum_{k\in I(i)} \bar{y}_k - \bar{e}_j - \bar{y}_j$$

As long as h remains exterior, d_{ij} would not change. Now assume h involves in a shrink operation and h' is the new exterior pseudonode containing i. Since h' is newly added to the planted forest d'_h is equal to current D. If we write d_{ij} after the shrinking operation:

$$\begin{aligned} d_{ij} &= D + c_{ij} - \sum_{k \in I(i) \setminus \{h,h'\}} \bar{y}_k - y_h - y_{h'} - \bar{e}_j - \bar{y}_j \\ &= D + c_{ij} - \sum_{k \in I(i) \setminus \{h,h'\}} \bar{y}_k - [\bar{y}_h + (D - d_h)] - 0 - \bar{e}_j - \bar{y}_j \\ &= d_h + c_{ij} - \sum_{k \in I(i)} \bar{y}_k - \bar{e}_j - \bar{y}_j \end{aligned}$$

it is equal to the previous one.

ii) Now assume $l_{b(i)} = l_{b(j)} = +$ and $b(i) \neq b(j)$ at the beginning of the stage. Let $b(i) = h_1$ and $b(j) = h_2$. Then

$$d_{ij} = d_{h_1} + d_{h_2} + c_{ij} - \sum_{k \in I(i)} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k$$

As long as h_1 and h_2 are not involved any shrinking operation d_{ij} remains constant. Now assume a shrinking operation is employed and at the end b(i) changes to h'_1, then $d_{h'_1} = D$, and d_{ij} becomes:

$$\begin{aligned} d_{ij} &= D + d_{h_2} + c_{ij} - \sum_{k \in I(i) \setminus \{h_1,h'_1\}} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k - y_{h_1} - y_{h'_1} \\ &= D + d_{h_2} + c_{ij} - \sum_{k \in I(i) \setminus \{h_1,h'_1\}} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k - [\bar{y}_{h_1} + (D - d_{h_1})] - 0 \\ &= d_{h_1} + d_{h_2} + c_{ij} - \sum_{k \in I(i)} \bar{y}_k - \sum_{k \in I(j)} \bar{y}_k \end{aligned}$$

that is equal to the previous one.
By using the same argument, for the cases h_2 or both h_1 and h_2 are invoved in a shrinking operation, the proof can be done. □

So in order to decide the edge that is joined to the planted forest, we can use d_{ij} values. For each edge (i, j), d_{ij} is calculated at most twice in a stage, one when b(i) is labelled by + and the other when b(j) is labelled by +. The lemma 3.1 motivates us to store d_{ij} values, for any edge, throughout any stage. We use fibonacci heaps to determine the appropriate minimums that guide the successive steps of the algorithm. The reason to use fibonacci heap is that, we can do find-min, insert operations in constant time and delete-min in $O(\log n)$ time.

In a stage, we maintain three types fibonacci heaps, FH-V, FH-E and FH-P(k). For unlabelled exterior nodes and $-$ labelled pseudonodes we maintain a fibonacci heap FH-V. The nodes in FH-V are ordered with the v_k values where

$$v_k = \begin{cases} \min\{d_{ij} + \bar{e}_{b(j)} : \ (i,j) \in E, \ \ j \in REAL(k), \ \ l_{b(i)} = +\}, & \text{k is an exterior pseudonode, } l_k = 0. \\ d_k + \bar{y}_k, & \text{k is an exterior pseudonode, } l_k = -. \end{cases}$$

For edges (i,j) with $l_{b(i)} = l_{b(j)} = +$ we use FH-E. The edges are ordered with the key value d_{ij}.

For each $-$ labelled and unlabelled exterior pseudonode we maintain another fibonacci heap FH-P(k) with the property that whenever the exterior pseudonode k is expanded, the fibonacci heap can be split into the groups that each groups corresponds to exterior nodes in k. The split can be done in $O(\log n)$ time [18, 1, 17]. These fibonacci heaps keep real nodes with the key value v'_j in an order that allow easy split in the case of expand, where

$$v'_j = \min\{d_{ij} : \ \ l_{b(i)} = +, \ \ (i,j) \in E\}$$

The usefulness of v'_j arises from the following property:

$$v_k = \min\{v'_j : \ \ j \in REAL(k)\} + \bar{e}_k$$

Now we will describe the operations and the main algorithm. First we will introduce the new procedure SCAN. This procedure is called whenever an exterior pseudonode is labelled by $+$, and it computes the key values necessary for the ordering of the nodes and the edges in the surface graph in the appropriate fibonacci heap.

SCAN(i,D)

```
begin
  for each $k \in REAL(i)$ and each $(k,j) \in E$
    begin
      if $l_{b(j)} \neq +$ or ($l_{b(j)} = +$ and $d_{kj} = \infty$)
        then Set $d_{kj} = D + \tilde{c}_{kj} + \bar{e}_i + D - d_i$
        else Set $d_{kj} = \frac{1}{2}(d_{kj} + D + \bar{e}_i + D - d_i)$
      if $l_{b(j)} = 0$
        then begin
              Set $v_{b(j)} = \min(v_{b(j)}, d_{kj} + \bar{e}_{b(j)})$
              if the value of $v_{b(j)}$ changes
                then Set $\rho(b(j)) = (k,j)$ and adjust the position of b(j) on FH-V
        end
```

```
            if $l_{b(j)} = +$ and $b(k) \neq b(j)$
                then Place (k,j) on FH-E
            if $l_{b(j)} \neq +$
                then begin
                        Set $v'_j = \min(v'_j, d_{kj})$
                        if the value of $v'_j$ changes
                            then Set $\rho'(j) = (k,j)$ and adjust the position of j on FH-P(b(j))
                end
        end
end{SCAN}
```

```
FIND-MIN(D)
begin
    Set $D_1 = \min\{v_j : \text{ j on FH-V}\}$
    Set $D_2 = \infty$
    while FH-E is not empty
        then begin
                Take the top edge (k,j)
                if $b(k) \neq b(j)$
                    then Set $D_2 = d_{kj}$
                    else Take the edge off from FH-E
        end
    end{while}
    Set $D = \min(D_1, D_2)$
end{FIND-MIN}
```

```
GROW(i,j,D)        /* b(i) is + labelled, b(j) is unlabelled pseudonode.
begin
    Set $l_{b(j)} = -, (i,j)$
         $d_{b(j)} = 0$
         $v_{b(j)} = d_{b(j)} + \bar{y}_{b(j)}$
    Put b(j) on FH-V
    if the mate of b(j) is an original node h
        then begin
                Delete (b(j),e) from M(h)
                Set $S = \{h\} \cup M(h)$
                SHRINK(S,p,h,4,b(j),e,D)
                Set m(b(j))=(p,e)
        end
        else begin
```

```
            Let m(b(j))=(k,e), k is pseudonode
            Set $l_k = +, e$, $d_k = D$
            SCAN(k,D)
        end
end{GROW}

SHRINK(S,p,base,type,mate,label,D)
begin
    Shrink S into a new pseudonode p
    Construct the structure tree corresponding to p
    Update $G_s$
    Set $l_p = label$ $d_p = D$ $\bar{y}_p = 0$
         type(p)=type base(p)=base m(p)=mate
    for each exterior node h in S
        begin
            Set $d_h = D$
            if $l_h = +$
                then begin
                        Set $\bar{y}_h = \bar{y}_h + (D - d_h)$
                        for all edges $e \in \delta(h)$ and $e \in \gamma(S)$
                            Set $\tilde{c}_e = \tilde{c}_e - (D - d_h)$
                end
            if $l_h = -$
                then begin
                        Set $\bar{y}_h = \bar{y}_h - (D - d_h)$
                        for all edges $e \in \delta(h)$ and $e \in \gamma(S)$
                            Set $\tilde{c}_e = \tilde{c}_e + (D - d_h)$
                        SCAN(h,D)
                end
            if $l_h = 0$
                then SCAN(h,D)
        end
end{SHRINK}

EXPAND(k,D)        /* k is a – labelled pseudonode.
begin
    if k is type-1 pseudonode        /* $\{(B_1, e_1), h, (B_i, e_i)|_{i=2}^{2}k + 1\}$
        then begin
                Let m(k)=(B,e)
                Unshrink k
        for all h in k
            Set $d_h = D$ $\bar{e}_h = \bar{e}_k + \bar{y}_k$
        Update $G_s$
        if e is emanated from even node j
            then begin                /* // is operation that concatenates lists.
                    Set M(j)=M(j)//(B,e)
                         S={j,M(j)}
                    SHRINK(S,p,j,4,$(-,e_1)$,m(B),D)
                    Set $l_{B_1} = l_k$ $x_{e_1} = 1$ $m(B_1) = (p, e_1)$
                    Delete FH-P(k)
                    Set $v_{B_1} = d_{B_1} + \bar{y}_{B_1}$ and put $B_1$ on FH-V
            end
        if e is emanated from $B_i, i \neq 1$
            then begin
                    Set M(j)=M(j)\$(B_i, e_i)$
                         S=$\{j, M(j)\}$
                    SHRINK(S,p,j,4,(-,$e_1$),$(B_i, e_i)$,D)
                    Set $l_{B_1} = l_k$ $x_{e_1} = 1$ $m(B_1) = (p, e_1)$
                         $l_{B_i} = -, e_i$ $m(B_i) = m(k)$
                    Delete FH-P(k)
                    Set $v_{B_1} = d_{B_1} + \bar{y}_{B_1}$ and put $B_1$ on FH-V
                    Set $v_{B_i} = d_{B_i} + \bar{y}_{B_i}$ and put $B_i$ on FH-V
            end
        if e is emanated from $B_1$
            then begin
                    Split FH-P(k)
                    Set $l_{B_1} = l_k$ $m_{B_1} = m_k$
                         $S = \{j\} \cup \{M(j)\}$
                    for all h in S
                        if FH-P(h) is not empty
                            then begin
                                    Set $v_h = v'_k + \bar{e}_h$
                                         $p(h) = p'(k)$ where k is the top element in FH-P(h)
                                         $l_h = 0$
    end
 if k is type-2 pseudonode          /* $\{(B_1, e_1), B_2, (B_3, e_2)\}$
    then begin
        Let m(k)=(B,e)
            l(k)=-,e' and e' between B' and k
        Set $S = \{(B', e'), k, (B, e)\}$
```

```
                SHRINK($S, p, B', 1, l_{B'}, m_{B'}, D$)
         end
      if k is type-3 pseudonode          /* $\{(B_i, e_i)|_{i=1}^{2} k + 1\}$
         then begin
              Let m(k)=(B,e)
                 e is emanated from $B_i$
              Unshrink k
              for all h in k
                 Split FH-P(k) into FH-P(h)
                 Set $\bar{e}_h = \bar{e}_k + \bar{y}_k$
                 Define the odd length path from $B_i$ to $B_1$
                 Swap the matched and unmatched edges over this path
                 for all pseudonodes h over this path
                    begin
                       Delete FH-P(h)
                       Set $d_h = D$
                       Label h (alternately label the nodes over the odd path by - and +
                             beginning with - from $B_i$)
                       if h is + labelled
                          then SCAN(h,D)
                             if h is $-$ labelled
                             then Set $v_h = d_h + \bar{y}_h$ and Put h on FH-V
                    end
                 for each node h over the remaining even length path
                    begin
                       if FH-P(h) is not empty
                          then begin
                                Set $v_h = v'_k + \bar{e}_h$
                                   $p(h) = p'(k)$ where k is the top element in FH-P(h)
                                   $l_h = 0$
                          end
                    end
         end
      if k is type-4 pseudonode          /* $\{j, (B_i, e_i)|_{i=1}^{2} k + 1\}$
         then begin
              Let m(k)=(B,e)
                 $l_k = (B', e')$
              Unshrink k
              for each h in k
                 Set $d_h = D$ $\bar{e}_h = \bar{e}_k + \bar{y}_k$
```

```
        Update $G_s$
        if e is emanated from even node j
          then begin
                Set M(j)=M(j)//(B,e)
                    S={j,M(j)}
                SHRINK(S,p,B',1,$l_{B'}$,,m(B),D)
                Set $l_{B_1} = l_k$ $x_{e_1} = 1$ $m(B_1) = (p, e_1)$
                Delete FH-P(k)
          end
        if e is emanated from $B_i$
          then begin
                Set M(j)=M(j)\$(B_i, e_i)$
                    S=$\{(B', e')j, M(j)\}$
                SHRINK(S,p,B',1,$l_{B'}$,$m(B')$,D)
                Set $l_{B_i} = -, e_i$ $x_{e_i} = 0$ $m(B_i) = (B, e)$
                Delete FH-P(k)
                Set $v_{B_i} = d_{B_i} + \bar{y}_{B_i}$ and put $B_i$ on FH-V
          end
    end
end{EXPAND}
```

```
AUGMENT(i,j,D)          /* b(i) and b(j) are + labelled pseudonodes
begin                   /* and are in different alternating trees.
  Let $r_1$ is the root of the alternating tree containing b(i).
     $r_2$ is the root of the alternating tree containing b(j).
     $P_1$ is the path from $r_1$ to b(i).
     $P_2$ is the path from $r_2$ to b(j).
  Set $AP = P_1 \cup (i,j) \cup P_2$
  for each edge $e \in AP$
    Set $x_e = 1 - x_e$ and changes the mates respectively
  for each exterior pseudonode k with $l_k = +$
     Set $y_k = \bar{y}_k + (D - d_k)$
  for each exterior pseudonode k with $l_k = -$
     Set $y_k = \bar{y}_k - (D - d_k)$
  for each edge $(i,j) \in E$
    if $l_{b(i)} = +$ and $l_{b(j)} = 0$
       then Set $\tilde{c}_{ij} = d_{ij} - D + \bar{e}_{b(j)}$
    if $l_{b(i)} = +$ and $l_{b(j)} = -$
       then Set $\tilde{c}_{ij} = d_{ij} - d_{b(j)} + \bar{e}_{b(j)}$
```

```
        if $l_{b(i)} = l_{b(j)} = +$ and $b(i) \neq b(j)$
            then Set $\tilde{c}_{ij} = 2d_{ij} - 2D$
        if $l_{b(i)} \neq +$ and $l_{b(j)} \neq +$
            then begin
                Set $\tilde{c}_{ij} = \tilde{c}_{ij} + \bar{e}_{b(i)} + \bar{e}_{b(j)}$
                if $l_{b(i)} = -$
                    then Set $\tilde{c}_{ij} = \tilde{c}_{ij} + (D - d_{b(i)})$
                if $l_{b(j)} = -$
                    then Set $\tilde{c}_{ij} = \tilde{c}_{ij} + (D - d_{b(j)})$
            end
    Delete thetwo alternating trees resulting augmentation from planted forest and
    remove the labels of the nodes over them
    if $\exists$ a + labelled node in $G_s$
        then begin
                Set D=0
                for all k in $G_s$
                    Set $d_k = 0$, $e_k = 0$, $v_k = \infty$
                for all edge $(i,j) \in E$
                    Set $d_{ij} = \infty$
                for all +labelled node k in $G_s$
                    SCAN(k,D)
        end
end{AUGMENT}
```

MAIN ALGORITHM

```
begin
    Initialization
    begin
        for all n in $G_s$
            Set $d_n = 0$, $\bar{e}_n = 0$, $v_n = \infty$, $l_n = 0$
        for all edge $(i,j) \in E$
            Set $d_{ij} = \infty$, $\tilde{c}_{ij} = c_{ij}$
        Set D=0
        for all pseudonode k in $G_s$
            Set $y_k = 0$, $m_k = 0$
                $REAL(k) = k$, $l_k = +, 0$
            SCAN(k,D)
```

```
 While ∃ a + labelled pseudonode in planted forest
   begin
     FIND-MIN(D)
     if $D = D_1$
       then begin
           Delete $v_j$ from FH-V
           if $l_j = 0$
             then begin
                 Add $\rho(j) = (i, k)$ to planted forest
                 if j is even node
                   then begin
                       Set $S = \{b(i), j, \} \cup \{M(j)\}$
                       SHRINK($S, p, 1, b(i), m_{b(i)}, D$)
                   end
                   else begin
                       if $y_j > 0$
                         then GROW(i,k,D)
                         else begin
                             Set S={b(i),j,m(j)}
                             SHRINK(S,p,2,b(i),m(b(i)),D)
                         end
                   end
             end
             else EXPAND(j,D)
       end
     if $D = D_2$
       then begin
           Add the edge (i,j) to the planted forest
           Backtrack from b(i) and b(j) over the planted forest
           if a common node m is reached
             then begin
                 Set S={The odd cycle begins and ends at m }
                 SHRINK(S,p,3,m,m(m),D)
             end
             else AUGMENT(i,j,D)
       end
 end{While}
 for all exterior pseudonodes k in $G_s$
   RECOVER(k)
end{MAIN-ALGORITHM}
```

The improvement of this implementation is derived from the use of fibonacci heaps on which each operations can be implemented in $O(\log|V|)$ time. Obviously there are O(V) stages each of which ends up with an augmentation.

In each stage, a vertex is scanned only when it is first labelled by $+$. Thus each edge can be examined at most twice. This results in $O(E\log|V|)$ total work requirement of Scan per stage. Shrink, Expand and Grow operations are each called at most O(V) times. Shrink and Expand takes $O(\log|V|)$ time and Grow takes constant time. So for these operations at most $O(V\log|V|)$ time is expended.

Augmentation is called only once in a stage and it takes O(E) time.

Finally, a node can be placed in FH-V at most twice, once when it is labelled by 0, and once when it is labelled by $-$, and an edge can be placed on FH-E at most once in a stage. Thus Find-Min can remove a vertex from FH-V at most twice and can remove an edge from FH-E at most once, that requires at most $O(E\log|V|)$ time per stage.

Consequently, the overall time bound of this algorithm is $O(VE\log|V|)$.

Chapter 4

THE SUCCESSIVE ALGORITHM

In the following, we describe the successive algorithm for the Chinese postman problem. Although the worst case time bound is the same as before, $O(\mid V \mid\mid E \mid \log \mid V \mid)$, provided by using the data structure suggested previously, the size of the alternating trees we worked on and the number of dual changes are decreased. This will speed up the algorithm. Beside this, we can divide the graph into two and solve the problem over two parts simultaneously and then combine the solutions. We haven't identified the rules of dividing the graph into two that gives rise to a parallel algorithm yet. But, we accept this algorithm as a first step towards such researches. In addition, this algorithm can easily be adapted to some variations required by the real life applications.

The motivation for this algorithm comes from the idea given for the primal matching algorithm by Derigs [9]. The notations and the definitions we will use during the description of the algorithm have already been given in previous chapters. For the implementation, the data structure given in chapter 3 is used.

4.1 The Successive Algorithm

This is neither primal nor dual algorithm. It consisits of stages each of which composed of two steps, namely **dual step** and **primal step**. We work on two complementary surface graphs, the dual surface graph $G_{DS} = (V_{DS}, E_{DS})$, and the primal surface graph $G_{PS} = (V_{PS}, E_{PS})$. Initially the dual surface graph is equal to the original graph except the odd degree nodes correspond to pseudonodes and the primal surface graph is empty. As the algorithm proceeds, we remove some part of the dual surface graph and add to the primal surface graph. The algorithm terminates when the dual surface graph is empty. We call the edges with one end in G_{DS} and the other end in G_{PS} as **hidden edges**, E_H.

The dual surface graph is a dual feasible subgraph and the dual step is done over that subgraph. We choose a pseudonode on that subgraph and grow an alternating tree rooted at that node. Whenever the alternating tree reaches another pseudonode in G_{DS} we stop growing and augment. In that time, we have a matched pair of pseudonodes at hand. We remove that pair with adjacent even degree nodes from the dual surface graph and add that part into the primal surface graph. After that operation some of the hidden edges become the element of E_{PS} and some of the edges in E_{DS} become hidden. Then the primal step begins. As I said just before, some of the hidden edges are made the edges of the primal surface graph and only these edges may violate the dual fesibility of G_{PS}. So the primal step is done for keeping the dual feasibility over the subgraph G_{PS}, and the stage terminates. At the termination of any stage the primal surface graph becomes both primal and dual feasible.

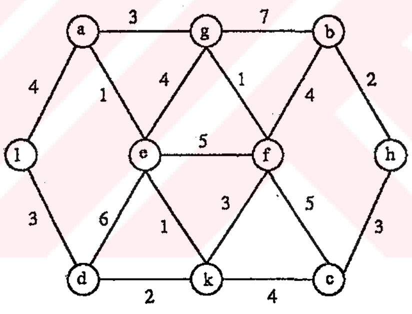
```
MAIN ALGORITHM
begin
Initialization
  Set $G_{DS} = (V_{DS}, E_{DS})$ where $V_{DS} = V$ except all degree nodes are pseudonodes
                                  $E_{DS} = E$
     $G_{PS} = (V_{PS}, E_{PS})$ where $V_{PS} = \emptyset$
                                  $E_{PS} = \emptyset$
     $E_H = \emptyset$
  for each pseudonode k in $G_{DS}$
    begin
      Set $y_k = 0$
          $l_k = 0$
          $REAL(k) = k$
    end
  for each edge e in $G_{DS}$
    begin
      Set $x_e = 0$
          $c'_e = c_e$
    end
  for each node n in $G_{DS}$
    Set b(n)=n
  While $G_{DS}$ is not empty
  begin
    Choose a pseudonode k in $G_{DS}$
    Set $l_k = +$
    while ∃ a + labelled pseudonode in$G_{DS}$
    begin{dual step}
        Set $\Delta = \min\{\Delta_1, \Delta_2\}$
    /* where

  /* $\Delta_1 = \min\{c'_e : e = (i,j),\ l_{b(i)} = +,\ l_{b(j)} = 0$ and $b(j)$ is even degree node $\}$.
  /* $\Delta_2 = \frac{1}{2}\min\{c'_e : e = (i,j), l_{b(i)} = l_{b(j)},$ and $b(i) \neq b(j)\}$

            $y_{b(i)} = y_{b(i)} + \Delta$
            for all edge $e \in \delta(b(i))$ and $e \in E$
                Set $c'_e = c'_e - \Delta$
        If $\Delta = \Delta_1$
            then begin
                SHRINK b(i) and b(j) into a new pseudonode p
                Set $l_p = +$
                    $y_p = 0$
            end
        If $\Delta = \Delta_2$
            then begin
                Set $x_e = 1$ where e=(i,j) is the edge satisfying the above minimum.
                Remove all the labels
                Delete the matched pair (b(i),b(j)) with adjacent original nodes from $G_{DS}$
                Add that part to $G_{PS}$
                Update the hidden edges $E_H$, $G_{DS}$, $G_{PS}$
            end
    end { dual step }
    If $\exists e \in \delta(b(i))$ such that ($c'_e < 0$ and $c'_e = \min\{c'_e : e \in \delta(b(i))\}$)
        then make a primal stage for e
end { While }
  for all exterior pseudonodes k in $G_{PS}$
    RECOVER(k)
end { MAIN-ALGORITHM }
```

The successive algorithm has at most $\frac{V}{2}$ stages. In each stage we make one dual step that can be accomplished at most in $O(|E| \log(|V|))$ time. At the beginning of each primal step we introduced a matched pair of pseudonodes one of which has a 0 dual variable. Thus, in each stage we make at most one primal step. A primal step requires $O(|E| \log(|V|))$ total work requirement provided by using the data structure we describe in chapter 3. Consequently, the worst case time bound of the algorithm is $O(|V||E| \log(|V|))$

Chapter 5

EXAMPLES FOR THE ALGORITHMS PROVIDED

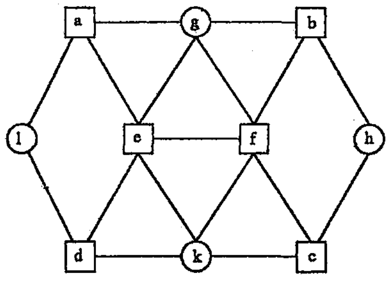
In this chapter we apply the three algorithms to the same graph given below. The numbers written over the edges are the costs of the edges.



We first apply the primal dual algorithm.

Primal-dual algorithm:

Initialization

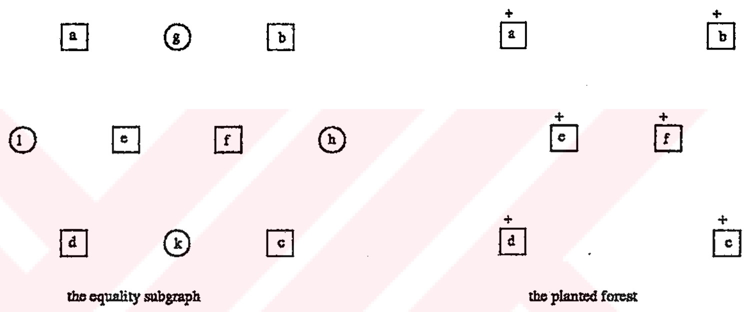


the surface graph

The initial values of the dual variables and the reduced costs are:

n	y_n	reduced costs									
		a	b	c	d	e	f	g	h	k	l
a	0					1		3			4
b	0						4	7	2		
c	0						5		3	4	
d	0					6				2	3
e	0						5	4		1	
f	0							1		3	

Based on these values, the equality subgaph and the planted forest can be defined as:



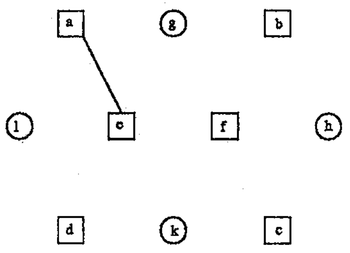
Stage 1

$$\begin{aligned} d_1 &= 2, && \text{because of the edge } (b,h) \\ d_2 &= \tfrac{1}{2}, && \text{because of the edge } (a,e) \\ d_3 &= \infty, && \text{there is no pseudonode with } - \text{ label} \\ &\Rightarrow \ d = \tfrac{1}{2} \end{aligned}$$

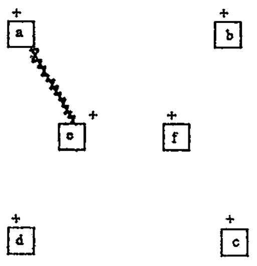
We make the dual update and the reduced cost updates.

n	y_n	reduced costs									
		a	b	c	d	e	f	g	h	k	l
a	$\frac{1}{2}$					0		$\frac{5}{2}$			$\frac{7}{2}$
b	$\frac{1}{2}$						3	$\frac{13}{2}$	$\frac{3}{2}$		
c	$\frac{1}{2}$						4		$\frac{5}{2}$	$\frac{7}{2}$	
d	$\frac{1}{2}$					5				$\frac{3}{2}$	$\frac{5}{2}$
e	$\frac{1}{2}$						4	$\frac{7}{2}$		$\frac{1}{2}$	
f	$\frac{1}{2}$							$\frac{1}{2}$		$\frac{5}{2}$	

After these updates the edge (a,e) is added to the equality subgraph. Since it joins two + labelled nodes in the planted forest, it results in an augmentation.



the equality subgraph



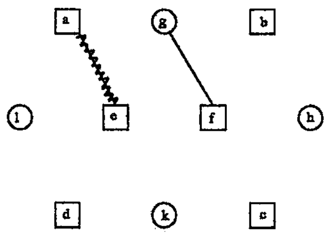
the planted forest

We delete the matched pair (a,e) from the planted forest and the stage ends.

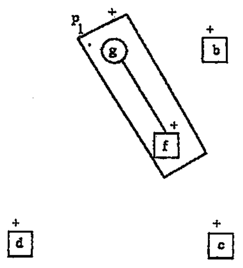
Stage 2

$$\begin{aligned} d_1 &= \tfrac{1}{2}, && \text{because of the edge } (f,g) \\ d_2 &= \tfrac{3}{2}, && \text{because of the edge } (b,f) \\ d_3 &= \infty, && \text{there is no pseudonode with } - \text{ label} \\ &\Rightarrow\ d = \tfrac{1}{2} \end{aligned}$$

n	y_n	reduced costs									
		a	b	c	d	e	f	g	h	k	l
b	1						2	6	1		
c	1						3		2	3	
d	1					6				1	2
f	1							0		2	



the equality subgraph

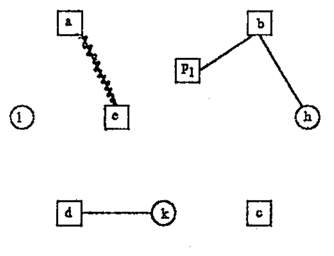


the planted forest

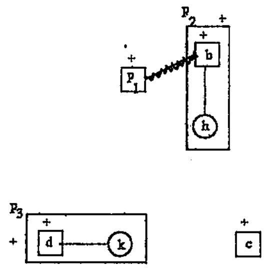
Set $y_{p_1} = 0$

$$\begin{aligned} d_1 &= 1, && \text{because of the edges } (b,h),(d,k) \\ d_2 &= 1, && \text{because of the edge } (b,f) \\ d_3 &= \infty, && \text{there is no pseudonode with } - \text{ label} \\ &\Rightarrow\ d = 1 \end{aligned}$$

b(n)	$y_{b(n)}$	n	reduced costs									
			b	c	d	f	g	a	e	h	k	l
b	2	b				0	4			0		
c	2	c				1				1		
d	2	d							$\frac{7}{2}$		0	1
p_1	1	f					0		$\frac{5}{2}$		1	
		g						$\frac{3}{2}$	$\frac{5}{2}$			



the equality subgraph



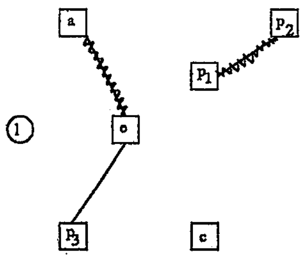
the planted forest

Set $y_{p_2} = y_{p_3} = 0$. Delete the matched pair (p_1, p_2) from the planted forest. The stage ends with this augmentation.

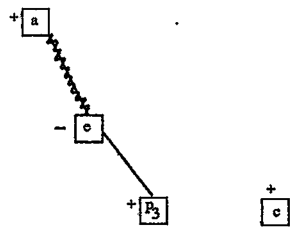
Stage 3

$$\begin{aligned} d_1 &= \tfrac{1}{2}, && \text{because of the edge } (e,k) \\ d_2 &= 1, && \text{because of the edge } (c,k) \\ d_3 &= \infty, && \text{there is no pseudonode with } - \text{ label} \\ &\Rightarrow \; d = 1 \end{aligned}$$

b(n)	$y_{b(n)}$	n	reduced costs									
			c	d	k	a	b	e	f	g	h	l
c	$\frac{5}{2}$	c			1				$\frac{1}{2}$		$\frac{1}{2}$	
p_3	$\frac{1}{2}$	d			0			3				$\frac{1}{2}$
		k						0	$\frac{1}{2}$			



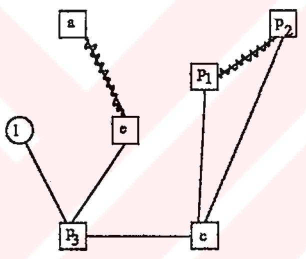
the equality subgraph



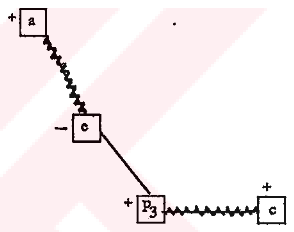
the planted forest

$$\begin{aligned} d_1 &= \tfrac{1}{2}, && \text{because of the edges } (f,k),(c,f),(d,l) \\ d_2 &= \tfrac{1}{2}, && \text{because of the edge } (c,k) \\ d_3 &= \tfrac{1}{2}, && \text{because of the pseudonode } e \\ &\Rightarrow\ d = \tfrac{1}{2} \end{aligned}$$

b(n)	$y_{b(n)}$	n	reduced costs									
			a	c	d	k	e	b	f	g	h	l
a	1	a					0			1		3
c	3	c				0			0		0	
p_3	1	d				0	3					0
		k					0		0			
e	0	e							3	3		

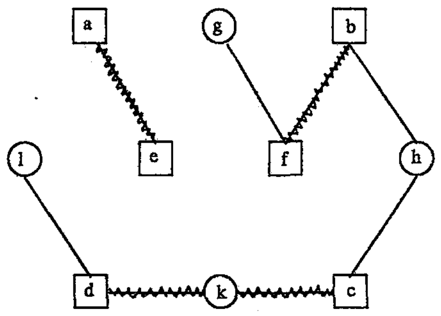


the equality subgraph



the planted forest

The edge (c,k) that is added to the equality subgraph results in an augmentation since it joins two + labelled nodes of the planted forest. Thus the stage ends. There is no free pseudonode left in the planted forest. We recover the blossoms p_1, p_2 and p_3. Then the algorithm stops. The solution is:



$x_{ae} = x_{bf} = x_{ck} = x_{dk} = 1$ and $x_e = 0$, for all other e.

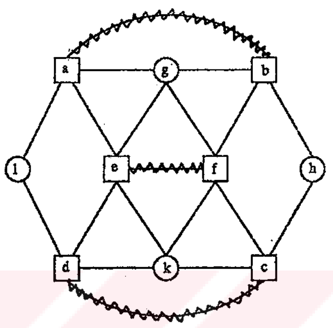
$y_a = 1$, $y_b = 2$, $y_c = 3$, $y_d = 2$, $y_e = 0$, $y_f = 1$, $y_{p_1} = 1$, $y_{p_2} = 0$, $y_{p_3} = 1$.

Now we apply the primal algorithm.

Primal Algorithm:

Initialization

We begin with a primal feasible solution. For the initial feasible solution one can put artificial edges. In this particular example we put two artificial edges, (a,b) and (c,d). The cost of the artificial edges are equal to the cost of the paths between the end nodes of the artificial edges.

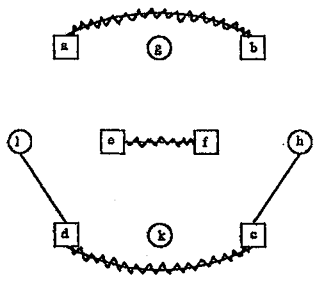


The initial primal feasible solution is $x_{ab} = x_{cd} = x_{ef} = 1$ and $x_e = 0$ for the other edges. Corresponding to that primal solution we choose the dual solution as $y_a = y_b = 5$, $y_c = y_d = 3$, and $y_e = y_f = \frac{5}{2}$.

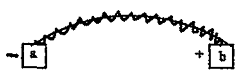
		reduced costs									
n	y_n	a	b	c	d	e	f	g	h	k	l
a	5		0			$-\frac{13}{2}$		-2			-1
b	5						$-\frac{7}{2}$	2	-3		
c	3				0		$-\frac{1}{2}$		0		
d	3					$\frac{1}{2}$				-1	0
e	$\frac{5}{2}$						0	$\frac{3}{2}$		$-\frac{3}{2}$	
f	$\frac{5}{2}$							$-\frac{3}{2}$			

Stage 1

Choose an edge with negative reduced cost, let e=(a,e).



the equality subgraph



the planted forest

$-c'_e = \frac{13}{2}$

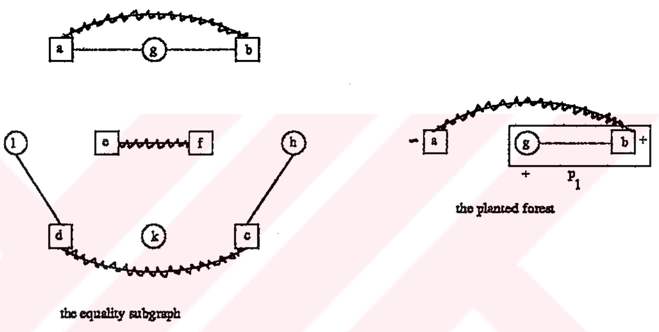
$d_1 = 2,$ because of the edge (b, g)

$d_2 = \infty,$ there is no edge between + labelled nodes with nonnegative reduced cost

$d_3 = 5,$ because of the pseudonode a

$\Rightarrow d = 2$

		reduced costs									
n	y_n	a	b	c	d	e	f	g	h	k	l
a	3		0			$-\frac{9}{2}$		0			1
b	7						$-\frac{11}{2}$	0	-5		



Set $y_{p_1} = 0$.

$-c'_e = \frac{9}{2}$

$d_1 = \frac{3}{2},$ because of the edge (e, g)

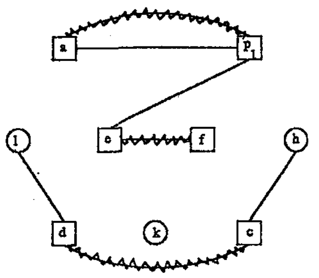
$d_2 = \infty,$ there is no edge between + labelled nodes with nonnegative reduced cost

$d_3 = 3,$ because of the pseudonode a

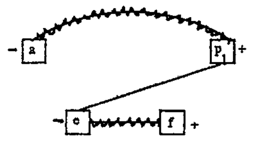
$\Rightarrow d = \frac{3}{2}$

Make dual update and reduced cost update.

			reduced costs									
b(n)	$y_{b(n)}$	n	a	b	g	c	d	e	f	h	k	l
a	$\frac{3}{2}$	a			0			-3				$\frac{5}{2}$
p_1	$\frac{3}{2}$	b			0				-7	$-\frac{13}{2}$		
		g						0	-3			



the equality subgraph



the planted forest

$-c'_e = \; 3$

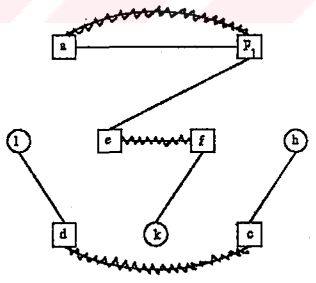
$d_1 = \; \frac{1}{2}$, because of the edge (f, k)

$d_2 = \; \infty$, there is no edge between $+$ labelled nodes with nonnegative reduced cost

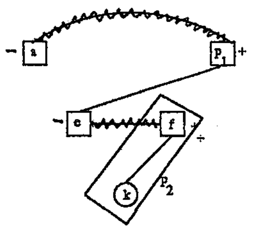
$d_3 = \; \frac{3}{2}$, because of the pseudonode a

$\Rightarrow \; d = \frac{1}{2}$

			reduced costs									
b(n)	$y_{b(n)}$	n	a	b	g	e	f	c	d	h	k	l
a	1	a			0	-2						3
p_1	2	b			0		-8			-7		
		g				0	-4					
e	2	e					0		1		-1	
f	3	f						-1			0	



the equality subgraph



the planted forest

Set $y_{p_2} = 0$

$-c'_e = \; 2$

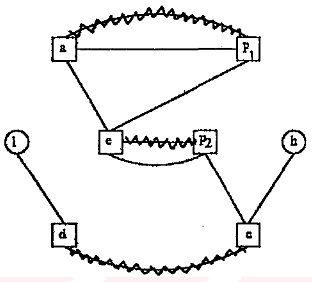
$d_1 = \; 1$, because of the edge (c, k)

$d_2 = \; \infty$, there is no edge between $+$ labelled nodes with nonnegative reduced cost

$d_3 = \; 1$, because of the pseudonode a

$\Rightarrow \; d = 1$

			reduced costs									
b(n)	$y_{b(n)}$	n	a	b	g	e	f	k	c	d	h	l
a	0	a			0	0						4
p_1	3	b			0		-10			-8		
		g				0	-6					
e	1	e					0	0		2		
p_2	1	f						0	-2			
		k							0	-2		

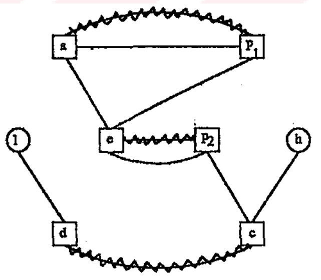


the equality subgraph

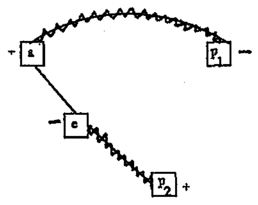
The reduced cost of the edge e=(a,e) became 0. Thus the stage ends.

Stage 2

Choose another edge with negative reduced cost, let e=(b,f). Label the pseudonode p_1 that contains b by - and its mate a by +. Since the edge (a,e) is in the equality subgraph, grow the alternating tree by using the edge (a,e). Label e by - and p_2 by +.



the equality subgraph



the planted forest

Since b is contained in a + labelled pseudode, we make primal change and following mini dual change.

$d = \min\{y_{p_1}, -c'_e\} = \min\{3, 7\} = 3$

			reduced costs									
b(n)	$y_{b(n)}$	n	b	g	a	c	d	e	f	h	k	l
p_1	0	b		0					-7	-5		
		g			3			0	-3			

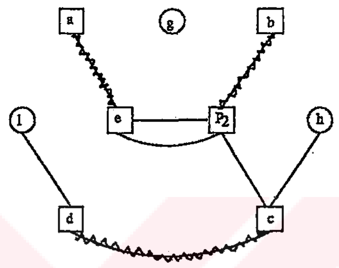
$d = \min\{y_b, -c'_e\} = \min\{7, 7\} = 7$

n	y_n	reduced costs									
		b	g	a	c	d	e	f	h	k	l
b	0		7					0	2		

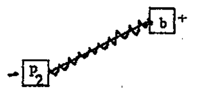
This is the end of stage 2.

Stage 3

Choose another edge say e=(f,g) with negative reduced cost. Label the pseudonode containing f, p_2, by - and its mate b by +.



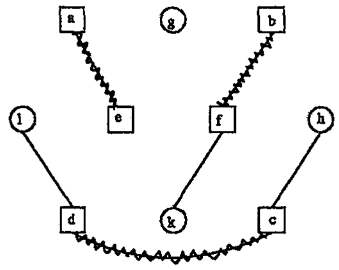
the equality subgraph



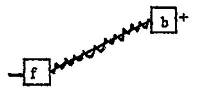
the planted forest

$$
\begin{aligned}
-c'_e &= 3 & & \\
d_1 &= 2, & & \text{because of the edge } (b,h) \\
d_2 &= \infty, & & \text{there is no edge between + labelled nodes with nonnegative reduced cost} \\
d_3 &= 1, & & \text{because of the pseudonode } p_2 \\
\Rightarrow & \; d = 1 & &
\end{aligned}
$$

b(n)	$y_{b(n)}$	n	reduced costs									
			f	k	b	a	c	d	e	g	h	l
p_2	0	f		0	0		-1		-1	-2		
		k					1	-1	1			
b	1	b								6	1	



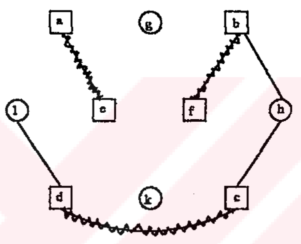
the equality subgraph



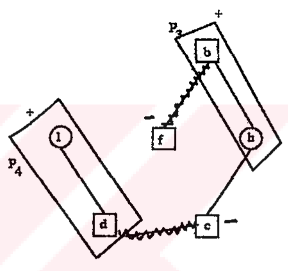
the planted forest

$$
\begin{aligned}
-c'_e &= 2 \\
d_1 &= 1, && \text{because of the edge } (b,h) \\
d_2 &= \infty, && \text{there is no edge between + labelled nodes with nonnegative reduced cost} \\
d_3 &= 3, && \text{because of the pseudonode } f \\
&\Rightarrow d = 1
\end{aligned}
$$

n	y_n	reduced costs									
		f	b	a	c	d	e	g	h	k	l
f	2		0		0		2	-1		1	
b	2							5	0		



the equality subgraph



the planted forest

Set $y_{p_3} = y_{p_4} = 0$

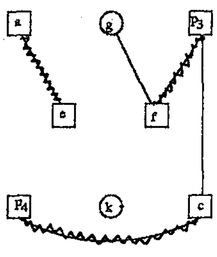
$$
\begin{aligned}
-c'_e &= 1 \\
d_1 &= 2, && \text{because of the edge } (d,e) \\
d_2 &= \infty, && \text{there is no edge between + labelled nodes with nonnegative reduced cost} \\
d_3 &= 2, && \text{because of the pseudonode } f \\
&\Rightarrow d = 1
\end{aligned}
$$

b(n)	$y_{b(n)}$	n	reduced costs									
			f	b	h	c	d	l	a	e	g	k
f	1	f		0		2				2	0	2
p_3	1	b			0						4	
		h				0						
c	2	c										2
p_4	1	d						0		1		-2
		l							3			

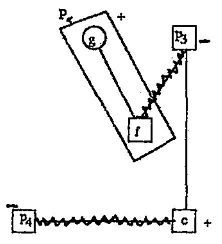
The reduced cost of the edge e=(f,g) became 0. Thus stage 3 ends.

Stage 4

We choose the edge e=(d,k). We label p_4 by - and its mate c by +.



the equality subgraph

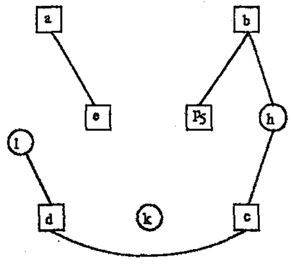


the planted forest

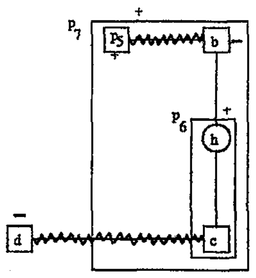
Set $y_{p_5} = 0$

$$\begin{aligned} -c'_e &= 2 \\ d_1 &= 1, && \text{because of the edge } (f,e) \\ d_2 &= 1, && \text{because of the edge } (f,c) \\ d_3 &= 2, && \text{because of the pseudonodes } p_3, p_4 \\ &\Rightarrow d = 1 \end{aligned}$$

			reduced costs									
b(n)	$y_{b(n)}$	n	f	g	b	h	c	d	l	a	e	k
p_5	1	f		0	0		0				2	1
		g			4					2	2	
p_3	0	b				0						
		h					0					
c	3	c						0				1
p_4	0	d							0		2	-1
		l								4		



the equality subgraph



the planted forest

Set $y_{p_6} = y_{p_7} = 0$. After the dual update we get the optimal solution.

			reduced costs									
b(n)	$y_{b(n)}$	n	d	c	h	b	f	g	a	e	k	l
d	2	d								3	0	1
p_7	1	c			0		0				0	
		h				0						
		b					0	4				
		f						0		1	0	
		g							1	1		

And finally we solve the problem by applying our successive algorithm.

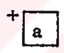
Successive Algorithm:

Initialization

Initially the primal surface graph and the set of hidden edges are empty. The dual surface graph is equal to the original graph except the odd degree nodes correspond to the pseudonodes. $x_e = 0$ for all e and $y_n = 0$ for all n.

Stage 1

We choose a as a free pseudonode in the dual surface graph and label it by +.

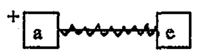


the planted forest

$d_1 =$	1,	because of the edge (a, e)
$d_2 =$	∞,	there is no edge in the planted forest that joins two + labelled nodes.
$d_3 =$	∞,	there is no pseudonode in the planted forest with - label.

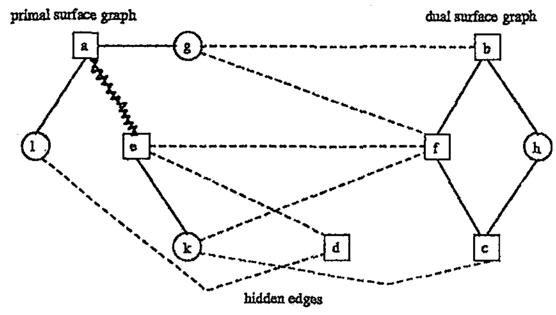
$$\Rightarrow \quad d = 1$$

$$y_a = 1 \quad \begin{array}{l} c'_{ag} = 2 \\ c'_{ae} = 0 \\ c'_{al} = 3 \end{array}$$



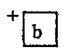
the planted forest

The edge (a,e) enters into the planted forest. We made augmentation using the edge (a,e) and put the matched pair (a,e) together with the neighbour even nodes to the primal surface graph. Since there is no edge in the primal surface graph that violates the dual feasibility, the stage ends.



Stage 2

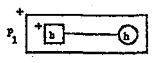
We choose another free pseudonode from the dual surface graph. Let it is b, label b by +.



the planted forest

$d_1 =$	$2,$	because of the edge (b, h)
$d_2 =$	$\infty,$	there is no edge in the planted forest that joins two + labelled nodes.
$d_3 =$	$\infty,$	there is no pseudonode in the planted forest with - label.
$\Rightarrow$	$d = 2$	

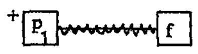
$$c'_{bh} = 0$$
$$y_b = 2 \quad c'_{bg} = 5$$
$$c'_{bf} = 2$$



the planted forest

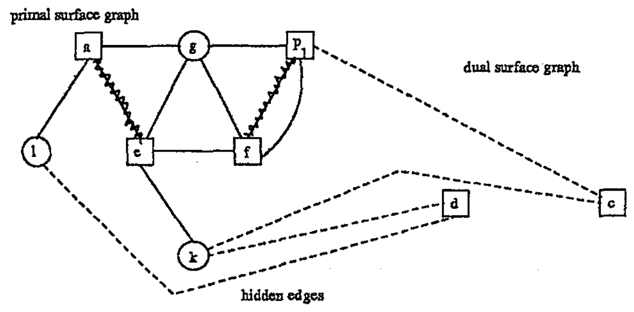
$d_1 =$	$2,$	because of the edge (b, f)
$d_2 =$	$\infty,$	there is no edge in the planted forest that joins two + labelled nodes.
$d_3 =$	$\infty,$	there is no pseudonode in the planted forest with - label.
$\Rightarrow$	$d = 2$	

$$c'_{bf} = 0$$
$$y_{p_1} = 2 \quad c'_{bg} = 3$$
$$c'_{hc} = 1$$



the planted forest

We augment p_1 and f and remove the pair together with the adjacent even nodes from the dual surface graph and add to the primal surface graph.



Stage 3

Since there is no path between c and d in the dual surface graph, we put an artificial matched edge between c and d with a high cost and put the matched pair into the primal surface graph. After making one primal stage over the primal surface graph we can obtain the solution.

Chapter 6

CONCLUDING REMARKS

In most of the cases, the real-life applications of the CPP are the large scale problems. Thus the parallel algorithms with the use of multiprocessors are very important for the goal of obtaining good solutions faster.

The main concern for this work was to develop an iterative algorithm that can exhibit parallelism. Having the same worst case time bound as the other algorithms in the literature, the successive algorithm can be thought of as a first step towards parallelism. It is parallel in a sense that, the primal and the dual steps can be concurrently done over the two disjoint subgraphs, the primal and the dual surface graphs. Beside this, the size of the alternating trees that the algorithm works on are small. These provide some speedups.

For the future research, there is a lot that has to be done. First, the successive algorithm, together with the primal-dual and the primal algorithm, have to be implemented using the same data structures in order to test the efficiency of it. As a complementary research, the algorithm can be improved so that it will be parallel in a more efficient fashion. Additionally, the variants of the algorithm that can handle the problems with multi postmen can be designed.

Bibliography

[1] M.O. Ball and U. Derigs, "An analysis an alternative strategies for implementing matching algorithms", *Networks* 13, 517-549 (1983).

[2] F. Barahona and A.R. Mahjoub, "On the cut polytope", *Mathematical Programming* 36, 157-173 (1986).

[3] F. Barahona, R. Maynard, R. Rammal and J.P. Uhry, "Morphology of ground states of two-dimensional frustration model", *Journal of Physics A, Mathematical and General* 15, 673-679 (1986).

[4] F. Barahona, "The max cut problem on graphs not contractible to K_5", *Operations Research Letters* 2, 107-111 (1982).

[5] F. Barahona, "Planar multicommodity flows, max-cut and the Chinese postman problem", Report 87454-OR, Institut für Operations Research, Universität, Bonn (1987).

[6] F. Barahona, "On via minimization", Research Report CORR 88-10, University of Waterloo, (1988).

[7] F. Barahona, "On some applications of the Chinese postman problem", Research Report CORR 88-85, University of Waterloo, (1988).

[8] W.H. Cunningham and A.B. Marsh, "A primal algorithm for optimum matching", *Mathematical Programming Study* 8, 50-72 (1978).

[9] U. Derigs, *Programming in Networks and Graphs*, (Springer-Verlag, Berlin, 1988).

[10] M. Dror, H. Stern and P. Trudeau, "Postman tour on a graph with precedence relations on arcs", *Network* 17, 283-294 (1987).

[11] J. Edmonds, "Paths, trees and flowers", *Canadian Journal of Mathematics* 17, 449-467 (1965).

[12] J. Edmonds, "The Chinese Postman Problem", *Operations Research* 13, suppl. 373, (1965).

[13] J. Edmonds, "Maximum matching and a polyhedron with (0,1)-vertices", *Journal of Research of the National Bureau of Standards,* 69B, 125-130 (1965).

[14] J. Edmonds and E.L. Johnson, "Matching: A well-solved class of integer linear programs", *Combinatorial structures and their applications*, 89-92, (Gordon and Breach, New York, 1970).

[15] J. Edmonds and E.L. Johnson, "Matching, Euler tours and the Chinese Postman", *Mathematical Programming* 5, 88-124 (1973).

[16] A. Frank, E. Tardos and A. Sebö, "Covering directed and odd cuts", *Mathematical Programming Study* 22, 99-112 (1984).

[17] M. Fredman and R. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms", *Journal of ACM* 34, 596-615 (1987).

[18] Z. Galil, S. Micali and H. Gabow, "An $O(|E||V|\log(|V|))$ algorithm for finding a maximal weighted matching in general graphs", *SIAM Journal of Computing* 15, 120-130 (1986).

[19] E. Korach, "Packing of T-cuts and other aspects of dual integrality", Ph.D. thesis, University of Waterloo, (1982).

[20] E. Korach and M. Penn, "Tight integral duality gap in the Chinese postman problem", Technical Report #360, Israel Institute of Technology, Department of Computer Science, Haifa (1985).

[21] Mei-Ko Kwan, "Graphic programming using odd or even points", *Chinese Mathematics* 1, 273-277 (1962).

[22] E. Lawler, *Combinatorial Optimization: networks and matroids*, (Holt, Rinehard and Winston, New York, 1976)

[23] L. Lovász and M.D. Plummer, *Matching Theory*, (Annals of Discrete Math (29), Elsevier Science Publishers B.V., Amsterdam 1986).

[24] K. Matsumoto, T. Ninhizenki and M. Saito, "The planar multicommodity flows, maximum matching and negative cycles" *SIAM Journal of Computing* 15, 495-510 (1986).

[25] C.H. Papadimitriou, *Combinatorial Optimization: Algorithms and Complexity,*, (Prentice-Hall Inc., New Jersey, 1982).

[26] A. Sebö, "The Schrijver system of odd join polyhedra", *Combinatorica* 8, 103-106 (1982).

[27] A. Sebö, "Finding the t-join structure of graphs" *Mathematical Programming* 36, 123-134 (1986).

[28] A. Sebö, "Undirected distances and the postman structure of graphs", *Journal of Combinatorial Theory, Series B* 49, 10-39 (1990).

[29] P.D. Seymour, "On odd cuts and plane multicommodity", *Proc. London Mathematics Society* 42, 178-192 (1972).