We now use Assumption 6.4 to note that we can rewrite $\sum_{(i,j)\in(\bar{S},S)} x_{ij}$ as $\sum_{(i,j)\in(S,\bar{S})} x_{ji}$. Consequently,

$$\Delta v \leq \sum_{(i,j)\in(S,\bar{S})} (u_{ij} - x_{ij} + x_{ji}) = \sum_{(S,\bar{S})} r_{ij}.$$

The following property is now immediate.

Property 6.2. *For any flow x of value v in a network, the additional flow that can be sent from the source node s to the sink node t is less than or equal to the residual capacity of any s–t cut.*

.4 GENERIC AUGMENTING PATH ALGORITHM

In this section, we describe one of the simplest and most intuitive algorithms for solving the maximum flow problem. This algorithm is known as the *augmenting path algorithm*.

We refer to a directed path from the source to the sink in the residual network as an *augmenting path*. We define the residual capacity of an augmenting path as the minimum residual capacity of any arc in the path. For example, the residual network in Figure 6.10(b), contains exactly one augmenting path 1–3–2–4, and the residual capacity of this path is $\delta = \min\{r_{13}, r_{32}, r_{24}\} = \min\{1, 2, 1\} = 1$. Observe that, by definition, the capacity δ of an augmenting path is always positive. Consequently, whenever the network contains an augmenting path, we can send additional flow from the source to the sink. The generic augmenting path algorithm is essentially based on this simple observation. The algorithm proceeds by identifying augmenting paths and augmenting flows on these paths until the network contains no such path. Figure 6.12 describes the generic augmenting path algorithm.

```
algorithm augmenting path;
begin
    x : = 0;
    while G(x) contains a directed path from node s to node t do
    begin
        identify an augmenting path P from node s to node t;
        δ : = min{r_ij : (i, j) ∈ P};
        augment δ units of flow along P and update G(x);
    end;
end;
```

Figure 6.12 Generic augmenting path algorithm.

We use the maximum flow problem given in Figure 6.13(a) to illustrate the algorithm. Suppose that the algorithm selects the path 1–3–4 for augmentation. The residual capacity of this path is $\delta = \min\{r_{13}, r_{34}\} = \min\{4, 5\} = 4$. This augmentation reduces the residual capacity of arc (1, 3) to zero (thus we delete it from the residual network) and increases the residual capacity of arc (3, 1) to 4 (so we add this arc to the residual network). The augmentation also decreases the residual capacity of arc (3, 4) from 5 to 1 and increases the residual capacity of arc (4, 3) from 0 to 4. Figure 6.13(b) shows the residual network at this stage. In the second iteration, suppose that the algorithm selects the path 1–2–3–4. The residual capacity of this

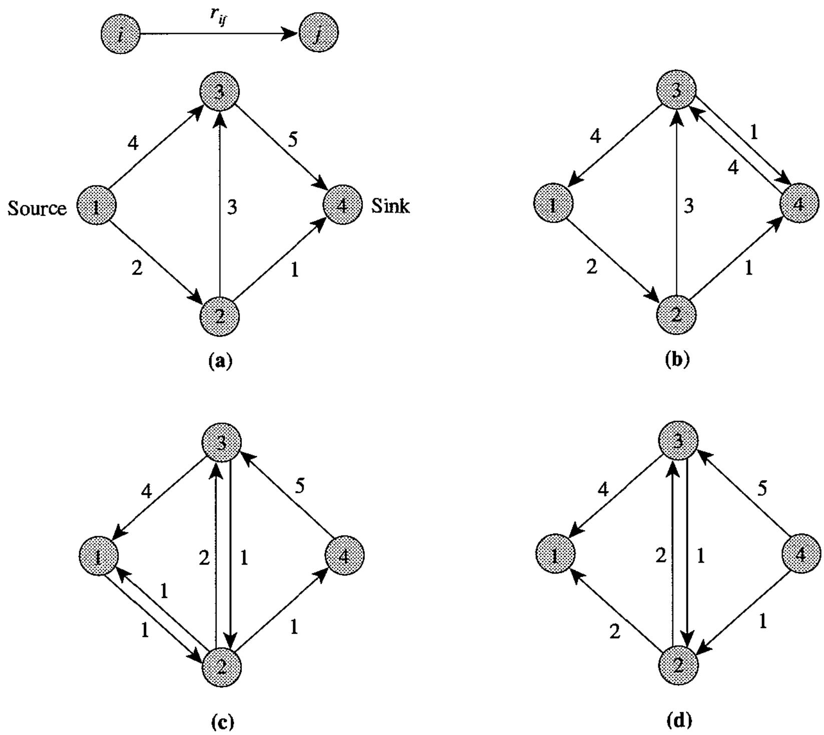

Figure 6.13 Illustrating the generic augmenting path algorithm: (a) residual network for the zero flow; (b) network after augmenting four units along the path 1–3–4; (c) network after augmenting one unit along the path 1–2–3–4; (d) network after augmenting one unit along the path 1–2–4.

path is $\delta = \min\{2, 3, 1\} = 1$. Augmenting 1 unit of flow along this path yields the residual network shown in Figure 6.13(c). In the third iteration, the algorithm augments 1 unit of flow along the path 1–2–4. Figure 6.13(d) shows the corresponding residual network. Now the residual network contains no augmenting path, so the algorithm terminates.

Relationship between the Original and Residual Networks

In implementing any version of the generic augmenting path algorithm, we have the option of working directly on the original network with the flows x_{ij}, or maintaining the residual network $G(x)$ and keeping track of the residual capacities r_{ij} and, when the algorithm terminates, recovering the actual flow variables x_{ij}. To see how we can use either alternative, it is helpful to understand the relationship between arc flows in the original network and residual capacities in the residual network.

First, let us consider the concept of an augmenting path in the original network. An augmenting path in the original network G is a path P (not necessarily directed) from the source to the sink with $x_{ij} < u_{ij}$ on every forward arc (i, j) and $x_{ij} > 0$ on every backward arc (i, j). It is easy to show that the original network G contains an augmenting path with respect to a flow x if and only if the residual network $G(x)$ contains a directed path from the source to the sink.

Now suppose that we update the residual capacities at some point in the algorithm. What is the effect on the arc flows x_{ij}? The definition of the residual capacity (i.e., $r_{ij} = u_{ij} - x_{ij} + x_{ji}$) implies that an additional flow of δ units on arc (i, j) in the residual network corresponds to (1) an increase in x_{ij} by δ units in the original network, or (2) a decrease in x_{ji} by δ units in the original network, or (3) a convex combination of (1) and (2). We use the example given in Figure 6.14(a) and the corresponding residual network in Figure 6.14(b) to illustrate these possibilities. Augmenting 1 unit of flow on the path 1–2–4–3–5–6 in the network produces the residual network in Figure 6.14(c) with the corresponding arc flows shown in Figure 6.14(d). Comparing the solution in Figure 6.14(d) with that in Figure 6.14(a), we find that the flow augmentation increases the flow on arcs (1, 2), (2, 4), (3, 5), (5, 6) and decreases the flow on arc (3, 4).

Finally, suppose that we are given values for the residual capacities. How should we determine the flows x_{ij}? Observe that since $r_{ij} = u_{ij} - x_{ij} + x_{ji}$, many combinations of x_{ij} and x_{ji} correspond to the same value of r_{ij}. We can determine

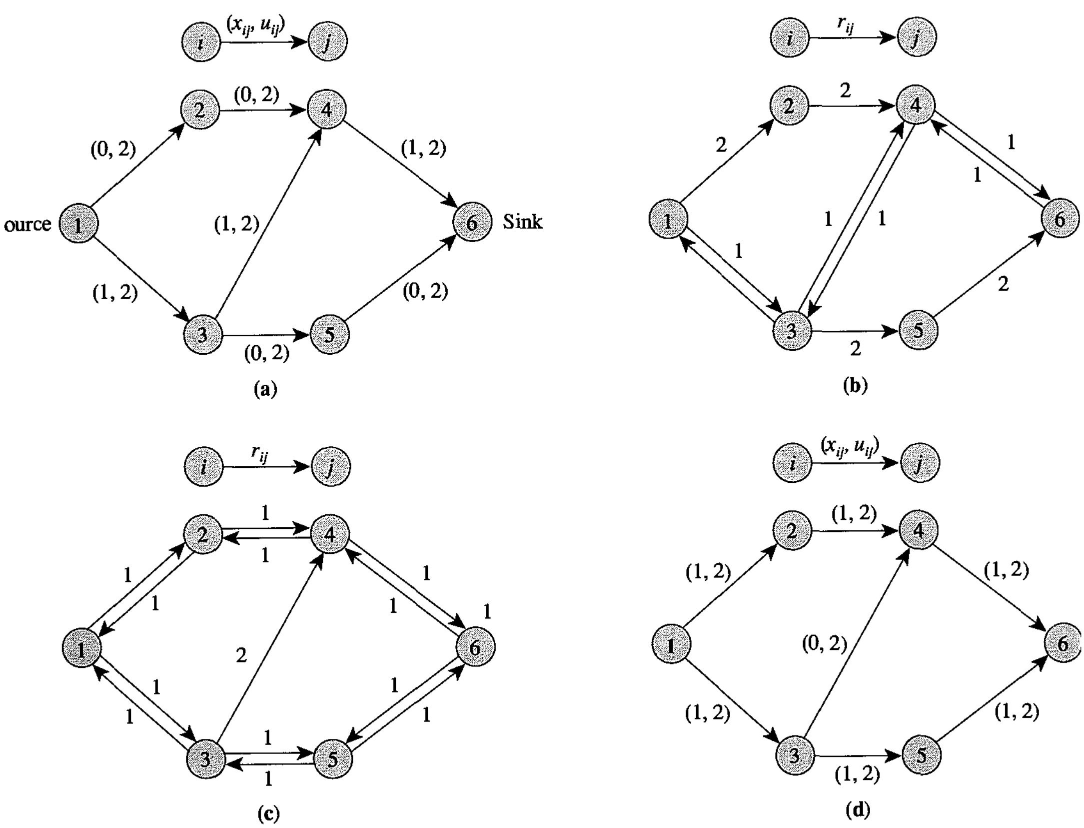

Figure 6.14 The effect of augmentation on flow decomposition: (a) original network with a flow x; (b) residual network for flow x; (c) residual network after augmenting one unit along the path 1–2–4–3–5–6; (d) flow in the original network after the augmentation.

one such choice as follows. To highlight this choice, let us rewrite $r_{ij} = u_{ij} - x_{ij} + x_{ji}$ as $x_{ij} - x_{ji} = u_{ij} - r_{ij}$. Now, if $u_{ij} \geq r_{ij}$, we set $x_{ij} = u_{ij} - r_{ij}$ and $x_{ji} = 0$; otherwise, we set $x_{ij} = 0$ and $x_{ji} = r_{ij} - u_{ij}$.

Effect of Augmentation on Flow Decomposition

To obtain better insight concerning the augmenting path algorithm, let us illustrate the effect of an augmentation on the flow decomposition on the preceding example. Figure 6.15(a) gives the decomposition of the initial flow and Figure 6.15(b) gives the decomposition of the flow after we have augmented 1 unit of flow on the path 1–2–4–3–5–6. Although we augmented 1 unit of flow along the path 1–2–4–3–5–6, the flow decomposition contains no such path. Why?

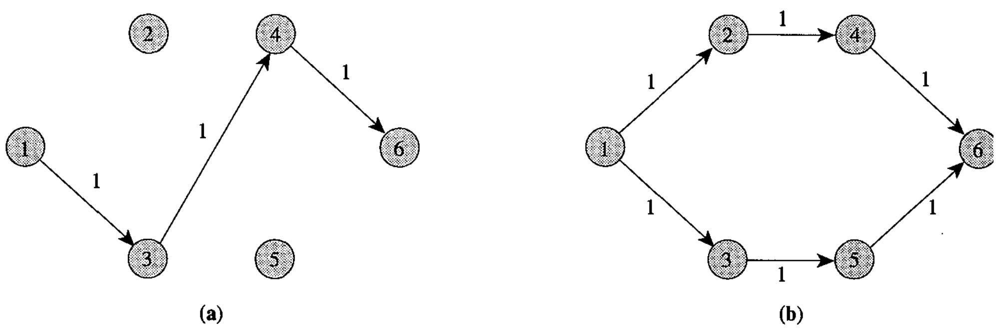

Figure 6.15 Flow decomposition of the solution in (a) Figure 6.14(a) and (b) Figure 6.14(d).

The path 1–3–4–6 defining the flow in Figure 6.14(a) contains three segments: the path up to node 3, arc (3, 4) as a forward arc, and the path up to node 6. We can view this path as an augmentation on the zero flow. Similarly, the path 1–2–4–3–5–6 contains three segments: the path up to node 4, arc (3, 4) as a backward arc, and the path up to node 6. We can view the augmentation on the path 1–2–4–3–5–6 as linking the initial segment of the path 1–3–4–6 with the last segment of the augmentation, linking the last segment of the path 1–3–4–6 with the initial segment of the augmentation, and canceling the flow on arc (3, 4), which then drops from both the path 1–3–4–6 and the augmentation (see Figure 6.16). In general, we can

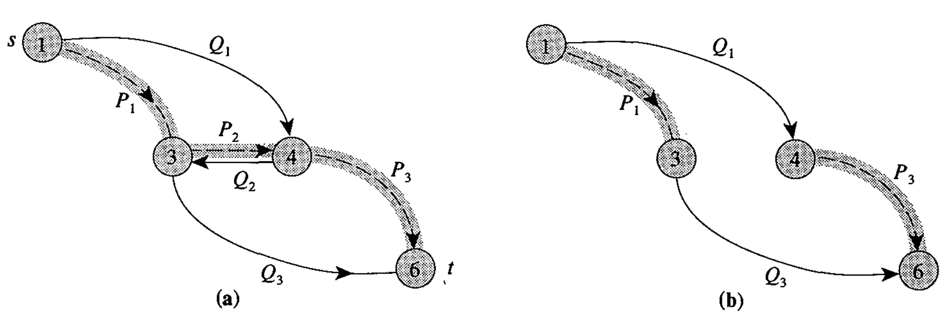

Figure 6.16 The effect of augmentation on flow decomposition: (a) the two augmentations P_1–P_2–P_3 and Q_1–Q_2–Q_3; (b) net effect of these augmentations.

view each augmentation as "pasting together" segments of the current flow decomposition to obtain a new flow decomposition.

.5 LABELING ALGORITHM AND THE MAX-FLOW MIN-CUT THEOREM

In this section we discuss the augmenting path algorithm in more detail. In our discussion of this algorithm in the preceding section, we did not discuss some important details, such as (1) how to identify an augmenting path or show that the network contains no such path, and (2) whether the algorithm terminates in finite number of iterations, and when it terminates, whether it has obtained a maximum flow. In this section we consider these issues for a specific implementation of the generic augmenting path algorithm known as the *labeling algorithm*. The labeling algorithm is not a polynomial-time algorithm. In Chapter 7, building on the ideas established in this chapter, we describe two polynomial-time implementations of this algorithm.

The labeling algorithm uses a search technique (as described in Section 3.4) to identify a directed path in $G(x)$ from the source to the sink. The algorithm *fans out* from the source node to find all nodes that are reachable from the source along a directed path in the residual network. At any step the algorithm has partitioned the nodes in the network into two groups: *labeled* and *unlabeled*. Labeled nodes are those nodes that the algorithm has reached in the fanning out process and so the algorithm has determined a directed path from the source to these nodes in the residual network; the unlabeled nodes are those nodes that the algorithm has not reached as yet by the fanning-out process. The algorithm iteratively selects a labeled node and scans its arc adjacency list (in the residual network) to reach and label additional nodes. Eventually, the sink becomes labeled and the algorithm sends the maximum possible flow on the path from node s to node t. It then erases the labels and repeats this process. The algorithm terminates when it has scanned all the labeled nodes and the sink remains unlabeled, implying that the source node is not connected to the sink node in the residual network. Figure 6.17 gives an algorithmic description of the labeling algorithm.

Correctness of the Labeling Algorithm and Related Results

To study the correctness of the labeling algorithm, note that in each iteration (i.e., an execution of the whole loop), the algorithm either performs an augmentation or terminates because it cannot label the sink. In the latter case we must show that the current flow x is a maximum flow. Suppose at this stage that S is the set of labeled nodes and $\bar{S} = N - S$ is the set of unlabeled nodes. Clearly, $s \in S$ and $t \in \bar{S}$. Since the algorithm cannot label any node in $\bar{S}$ from any node in S, $r_{ij} = 0$ for each $(i, j) \in (S, \bar{S})$. Furthermore, since $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$, $x_{ij} \le u_{ij}$ and $x_{ji} \ge 0$, the condition $r_{ij} = 0$ implies that $x_{ij} = u_{ij}$ for every arc $(i, j) \in (S, \bar{S})$ and $x_{ij} = 0$ for every arc $(i, j) \in (\bar{S}, S)$. [Recall our assumption that for each arc $(i, j) \in A$,