Finding a Maximum Density Subgraph

A. V. Goldberg [†]

Computer Science Division

University of California

Berkeley, California 94720

[†] Supported by a Fannie and John Hertz Foundation Fellowship.

Finding a Maximum Density Subgraph

A. V. Goldberg †

Computer Science Division

University of California

Berkeley, California 94720

1. Introduction

The density of a graph is the ratio of the number of edges to the number of vertices of the graph. In this paper we will develop a fast algorithm for the following problem: given an undirected graph $G=(V,E)$ with $|V|=n$, $|E|=m$, find a subgraph of G of maximum density. Since the density of a graph is half the average degree, the problem is equivalent to that of finding a maximum average degree subgraph of a graph. Note that we can find a maximum density subgraph of a directed graph by applying the algorithm to the associated undirected graph.

In the following discussion, we assume familiarity with basic network theory, including network flows and the max-flow min-cut theorem. See, for example, [2].

We reduce the problem of finding a maximum density subgraph to a series of minimum capacity cut computations, which in turn can be done using network flow techniques. A similar approach to the same problem was used by Picard and Queyranne [5]. Their solution, however, involves, in the worst case, n min-cut computations on networks with $n+2$ nodes, whereas the algorithm presented here requires $O(logn)$ min-cut computations on networks of the same size.

† Supported by a Fannie and John Hertz Foundation Fellowship.

As we will show, our algorithm generalizes to multigraphs and to graphs with nonnegative weights on edges. The algorithm also generalizes to graphs in which both nodes and edges are weighted.

The algorithm works as follows. Let D be the density of a desired subgraph. At each stage of the algorithm, we have a "guess" g for D. Then we construct a network and perform a min-cut computation that enables us to decide whether our a binary search for D and to find a maximum density subgraph.

2. Network Construction

Let d_i be the degree of vertex i of G. Given a "guess" g, we convert G into a network $N=(V_N, E_N)$ as follows.

We add source s and sink t to the set of vertices of G; replace each (undirected) edge of G by two directed edges of capacity 1 each; connect source to every node i of G by an edge of capacity m; and connect every node i of G to the sink by an edge of capacity $(m + 2g - d_i)$. Figure 1 illustrates the construction.

More formally,

$$
\begin{aligned}
&V_N = V \bigcup \{s, t\} \\
&E_N = \{(i,j) \mid \{i,j\} \in E\} \bigcup \{(s,i) \mid i \in V\} \bigcup \{(i,t) \mid i \in V\} \\
&c_{ij} = 1 && \{i,j\} \in E \\
&c_{si} = m && i \in V \\
&c_{it} = m + 2g - d_i && i \in V \\
&c_{ij} = 0 && \{i,j\} \notin E_N.
\end{aligned}
$$

Notice that all capacities are nonnegative because for any i, $d_i \leq m$, and our "guess" g will always be nonnegative.

A partition of V_N into two sets, S and T, such that $s \in S$ and $t \in T$, determines an s-t cut. Let $V_1 = S - \{s\}$, and $V_2 = T - \{t\}$. If $|V_1| = 0$, then the capacity of the cut $c(S, T) = m\,|V|$; otherwise, the capacity of the cut is given

by (see figure 2)

$$
\begin{aligned}
c(S,T) &= \sum_{i\in S,\, j\in T} c_{ij} \\
&= \sum_{j\in V_2} c_{sj} + \sum_{i\in V_1} c_{it} + \sum_{i\in V_1, j\in V_2} c_{ij} \\
&= m\mid V_2\mid + \left[m\mid V_1\mid + 2g\mid V_1\mid - \sum_{i\in V_1} d_i\right] + \sum_{i\in V_1, j\in V_2} c_{ij} \\
&= m\mid V\mid + \mid V_1\mid 2\left(g - \left(\frac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} 1}{2\mid V_1\mid}\right)\right).
\end{aligned}
$$

But $\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} 1$ is twice the number of edges in the subgraph of G induced by V_1, so

$$D_1 = \frac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} 1}{2\mid V_1\mid}$$

is the density of the subgraph of G generated by V_1. Therefore, $c(S_s, T_s) = m\mid V\mid + 2\mid V_1\mid(g - D_1)$. The following theorem that gives a way to tell whether g is too big or too small.

THEOREM 1 Assume that S and T give a minimum capacity cut. If $\mid V_1\mid \neq 0$, then $g \leq D$; if $\mid V_1\mid = 0$ (i.e. $S = \{s\}$), then $g \geq D$.

PROOF Notice that if $S_s = \{s\}$, $T_s = V \bigcup \{t\}$, then the capacity of the cut (S_s, T_S) is $C_s = m\mid V\mid$. We have $C_s \geq c(S,T)$, i.e. $2\mid V_1\mid(g - D_1) \leq 0$. So if V_1 is not empty then $g \leq D_1$, i.e. there must exist a subgraph of G whose density is at least g. Thus, $g \leq D$, the maximum density of a subgraph of G.

Conversely, assume that there exists a subgraph of G, $\overline{G} = (\overline{V},\overline{E})$ with the density $\overline{D} > g$. Then the capacity of the cut defined by $\overline{S} = \overline{V} \bigcup \{s\}$, $\overline{T} = V_N - \overline{S}$ is

$$\overline{C} = m\mid V\mid + 2\mid \overline{V}\mid(g - \overline{D}) \leq C_s$$

It follows that in this case V_1 will not be empty, for otherwise the cut (S, T)

would not be a min-cut. In other words, if V_1 is empty, then the density of any subgraph of G is less than or equal to g, and $g \geq D$.

3. The Algorithm

The maximum density, D, can take on only a finite set of values. Recall that D is the ratio of a number of edges to the number of vertices of for some subgraph of G, so

$$D \in \left\{ \frac{m'}{n'} \mid 0 \leq m' \leq m, 1 \leq n' \leq n \right\}$$

It follows that D lies between 0 and m; furthermore, the smallest distance between the two different possibe values of D is $\frac{1}{n(n-1)}$. To see this, we notice that the difference Δ between two different possibe values is

$$\Delta = \frac{m_1}{n_1} - \frac{m_2}{n_2} = \frac{m_1 n_2 - m_2 n_1}{n_1 n_2}$$

If $n_1 = n_2$, then $|\Delta| \geq \frac{1}{n}$, otherwise $n_1 n_2 \leq n(n-1)$, so $|\Delta| \geq \frac{1}{n(n-1)}$. We have the following theorem:

THEOREM 2 If G' is a subgraph of G with density D', and no subgraph of G has a density greater than or equal to $D' + \frac{1}{n(n-1)}$, then G' is a maximum density subgraph.

The theorem tells us when we can stop the search. Now we can describe the algorithm.

The algorithm is given below.

During the execution of the algorithm, V_1 contains vertices of a subgraph of G with density greater than or equal to l. When the algorithm terminates, we know that there is no subgraph with density $l + \frac{1}{n(n-1)}$ or greater, so, by

```
l ← 0; u ← m; V1 ← empty;
while u − l ≥ 1/(n(n−1)) do
  begin
    g ← (u+l)/2;
    Construct N = (VN, EN);
    Find min-cut (S, T);
    if S = {s} then  u ← g
                else
                  begin
                    l ← g;
                    V1 ← S − {s};
                  end;
  end;
return (subgraph of G induced by V1)
```

Theorem 2, the subgraph returned is a maximum density subgraph.

4. The Running Time

THEOREM 3 Let $M(v, e)$ be the time required to find a minimum capacity cut in a network with v vertices and e edges. Then the algorithm runs in time $O(M(n, n+m)\log n)$.

PROOF There is only one loop which is executed $\lceil \log((m+1)n(n-1)) \rceil = O(\log n)$ times. Inside the loop, finding min-cut is the dominating step. The network has $n+2 = O(n)$ vertices and $2m+2n = O(n+m)$ edges. So the running time is $O(M(n, n+m)\log n)$.

If we use Karzanov algorithm [1] which finds a minimum capacity cut in a network with k vertices in $O(k^3)$ steps, we can find a maximum density subgraph in time $O(n^3 \log n)$. Of course, use of a faster min-cut algorithm improves the

running time of our graph density algorithm. The algorithm of Sleator and Tarjan [6], for example, has $M(v,e) = O(ve\log v)$. The use of this min-cut algorithm results in an $O(n(n+m)\log(n)\log(n+m))$ graph density algorithm, which gives a better bound for sparse graphs. For a discussion of other max-flow min-cut algorithms, see [4].

5. Generalization to weighted graphs

We can generalize the notion of density to weighted graphs in a natural way by defining the density of a weighted graph to be the ratio of the sum of weights of all edges of the graph to the number of vertices in it. With minor modifications, the algorithm can solve the problem for graphs with nonnegative rational weights (also, for multigraphs, which in the context of the problem are equivalent to integer weighted graphs).

By multiplying all weights of the edges by the product of denominators of weights, we obtain an equivalent problem with integer weights. The network is constructed as before except the capacities of the edges that correspond to the edges of G are equal to the weights of the corresponding edges, and d_i is the sum of weights of all edges of G incident to the node i. Since the weights are integers, we can stop the binary search when the size of the interval of uncertainty is less than $\frac{1}{n(n-1)}$ (the proof is same as before). The density of any subgraph of G does not exceed W, the sum of weights of all edges of the graph, so we start the search with u equal to W. W is no greater than the product of all denominators and non-zero numerators of the weights in the original problem, and the product is less than or equal to 2^l, where l is the length of the input to the problem. Note that $l \geq n$ for any reasonable encoding of the problem, so the number of iterations of the algorithm is $O(log(n(n-1)2^l)) = O(l)$. The total running time of the algorithm is polynomial in l; if we use Karsanov min-cut algorithm [1], the run-

ning time is $O(ln^3) \leq O(l^4)$.

Notice that we require the graph to have nonnegative weights on edges so that the resulting network has nonnegative capacities and we can apply a max-flow min-cut algorithm.

We also can find a maximum density subgraph if weights are nonnegative real numbers. To do this we use Megiddo's technique [3]. In order for this technique to be applicable the essential computation (the max-flow, min cut computation) must use only additions and comparisons. Karsanov algorithm, as well as many other algorithms for max-flow computation, have this property. We also require the algorithm to produce a maximal S-set of the min-cut so that the remark to Theorem 1 applies. Karzanov algorithm is an example of such an algorithm.

The network is constructed as before except g is kept as a symbolic parameter. At every step of the algorithm, we have some interval of uncertainty $[x, y]$ for g (initially, $[-\infty, \infty]$). Because of the requirement on the network flow algorithm, the expressions that occur during the computation of the network flow algorithm are linear in g (possibly constants). Every time the network flow algorithm makes a comparison, the outcome may depend on g. However, since the functions compared are linear in g, there is at most one critical value g'. If $g' \notin [x, y]$, we know which branch to choose; otherwise, we determine if the "guess" g' is too big or too small in the same way as in the original algorithm, and choose $[x, g']$ or $[g', y]$ as our next interval of uncertainty.

When the symbolic computation is finished, we know that there is only one plausible value of g in the interval of uncertainty $[x, y]$, so we plug x instead of g and do a network flow computation to obtain a maximum density subgraph. Because of the second requirement on the network flow algorithm, the subgraph is produced even if the maximal density is equal to x. For more details, see

Megiddo's paper [3].

It follows from the Megiddo's main theorem that if the network flow algorithm uses $O(p(n,m))$ comparisons and $O(q(n,m))$ additions, and therefore runs in time $O(p(n,m)+q(n,m))$, then the algorithm described above runs in time $O(p(n,m)(p(n,m)+q(n,m)))$.

If Karsanov min-cut algorithm is used, the running time is $O(n^6)$.

6. Generalization to node weighted graphs

We assume now that both edges and nodes of G are weighted. Let $w_e, e \in E$ be weights of the edges and $v_j, 1 \le j \le n$ be weights of the nodes. As before, we assume that $w_e \ge 0$. The density of G is defined as $\dfrac{\sum_{e \in E} w_e + \sum_{1 \le j \le n} v_j}{n}$. We show how to modify the above algorithms to find a maximal density subgraph in this case.

The network has the same structure, but edge capacities are a little different. We define the capacities to be

$$\begin{aligned} c_{ij} &= w_{ij} && \{i,j\} \in V \\ c_{si} &= m' && i \in V \\ c_{it} &= m' + 2g - d_i - 2v_i && i \in V \\ c_{ij} &= 0 && \{i,j\} \notin E_N \end{aligned}$$

where m' is such that all capacities are nonnegative, and d_i is the sum of weights of all edges of G adjacent to node i. These capacities are much like before, except there is a term $-2v_i$ in the expression for capacities of the edges connecting node i to sink to account for the weight of the node, and m' is used instead of m to insure nonnegativity of all edges.

If S and T define an s–t cut, its capacity is

$$c(S,T) = \sum_{i \in S, j \in T} c_{ij}$$

$$= \sum_{j \in V_2} c_{sj} + \sum_{i \in V_1} c_{it} + \sum_{i \in V_1, j \in V_2} c_{ij}$$

$$= m' \mid V_2 \mid + \left(m' \mid V_1 \mid + 2g \mid V_1 \mid - \sum_{i \in V_1} d_i - 2 \sum_{i \in V_1} v_i \right) + \sum_{i \in V_1, j \in V_2} w_{ij}$$

$$= m' \mid V \mid + \mid V_1 \mid 2 \left(g - \left(\frac{\sum_{i \in V_1} d_i - \sum_{i \in V_1, j \in V_2} w_{ij}}{2 \mid V_1 \mid} + \frac{\sum_{i \in V_1} v_i}{\mid V_1 \mid} \right) \right)$$

But $\sum_{i \in V_1} d_i - \sum_{i \in V_1, j \in V_2} w_{ij}$ is twice the weight of edges in the subgraph of G generated by V_1, so

$$D_1 = \frac{\sum_{i \in V_1} d_i - \sum_{i \in V_1, j \in V_2} w_{ij}}{2 \mid V_1 \mid} + \frac{\sum_{i \in V_1} v_i}{\mid V_1 \mid}$$

is the density of the subgraph of G generated by V_1.

It is straightforward to see that all the above results go through, and we can solve the problem with weighted nodes in the same way (for rational or real weights).

It is also reasonable to define the density of the node weighted graph to be the ratio of the sum of weights of edges over the sum of weights of nodes. The reader can verify that the algorithm can be modified to handle this definition as well by changing capacities of the edges going into the sink as follows:

$$c_{it} = m' + 2gv_i - d_i \qquad i \in V.$$

7. Conclusion

We have shown how to combine the network flow techniques with binary search to obtain an efficient algorithm for finding a maximum density subgraph of a graph. We also demonstrated that the algorithm can be easily adapted to handle various of generalizations of the problem.

Acknowledgment

The author is very grateful to Gene Lawler and David Shmoys for helpful discussion and suggestions.

References

[1] Karzanov, A. V., Soviet Math. Dokl., Vol 15, 1974, 434-437.

[2] Lawler, E.L., "Combinatorial Optimization: *Networks and matroids*", Holt, Rinehart, and Winston, New York, 1976.

[3] Megiddo, N., "Combinatorial Optimization with Rational Objective Function", Mathematics of Operations Research, Vol.4, No.4, November 1979.

[4] Papadimitriou, C. H. and Steiglitz, K., *Algorithms and Complexity*", Prentice-Hall, Inc, 1982, 216-217.

[5] Picard, J.-C. and Queyranne, M., with Applications to Graph Theory", *Networks*, Vol 12 (1982), 141-159.

[6] Sleator, D.D., PH.D. dissertation, Stanford University, 1930.

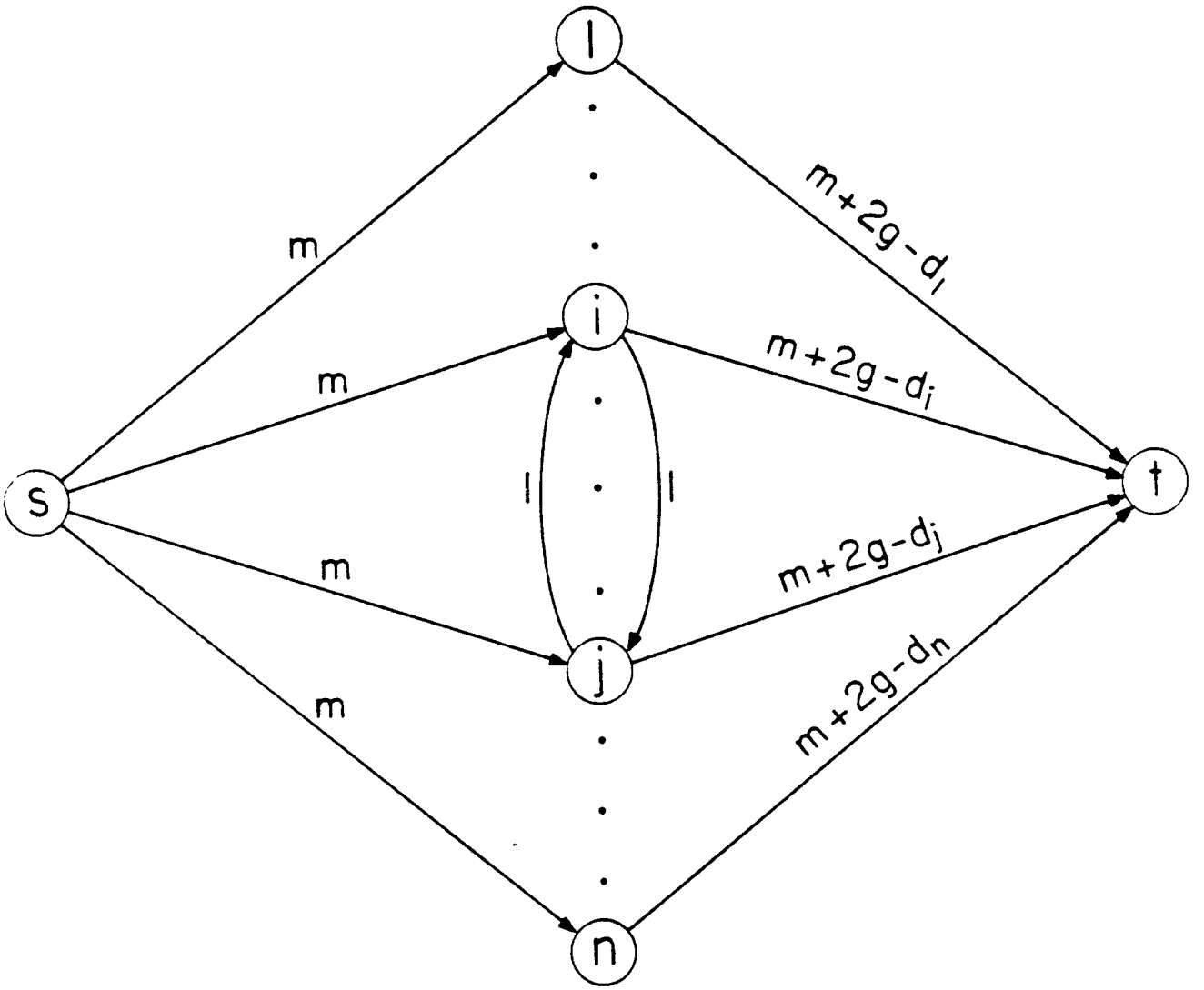

Figure 1

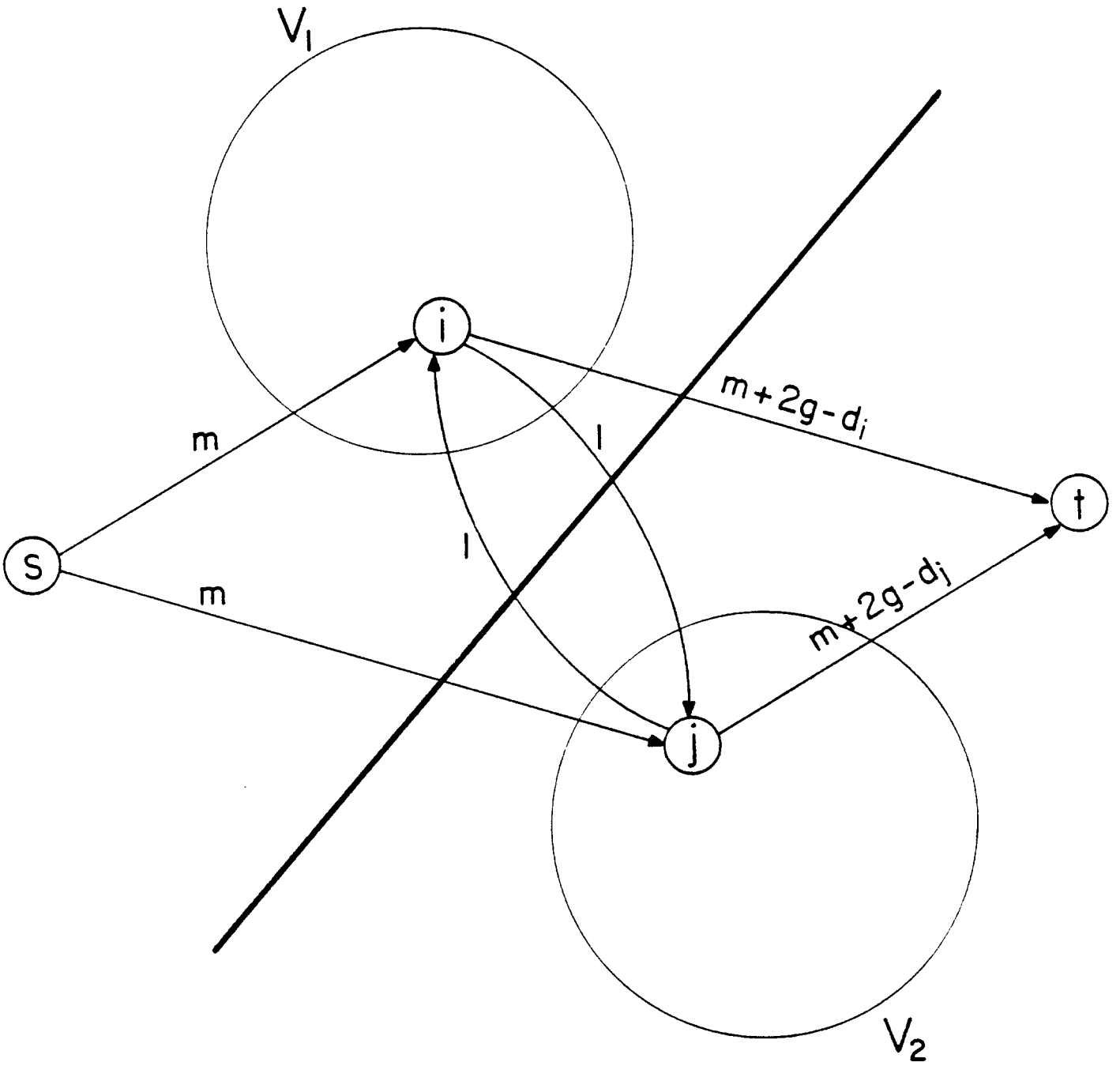

Figure 2