On Computing Graph Closures

Samir Khuller*

TR 88-921
June 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

* This work was supported by the National Science Foundation under grant DCR-85-52938 and PYI Matching Funds from AT&T Bell Labs.

On Computing Graph Closures

Samir Khuller *
Computer Science Department,
Cornell University,
Ithaca, NY 14853

June 10, 1988

Abstract

Given a graph G, the closure of G is the graph obtained from G by recursively joining pairs of non-adjacent vertices whose degree sum is at least n until no such pair remains. We give an efficient algorithm to compute the closure using F-heaps. We also define the general closure of a graph and show that computing the general closure is P-complete with respect to log space transformations.

1 Introduction

Let $G(V, E)$ be an undirected simple graph. The *closure* of G is the graph obtained from G by recursively joining pairs of non-adjacent vertices whose degree sum is at least n (the number of vertices in G) until no such pair remains. We denote the closure of G by $c(G)$. In [BoMu 76] it is shown that the closure of G is well defined and the sequence in which the edges are added is not important.

The closure of a graph has the following interesting property : A graph G has a Hamiltonian cycle if and only if the closure has a Hamiltonian cycle

*supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell labs.

(see [BoMu 76]). In fact, if one is given a hamiltonian cycle in $c(G)$ it is not hard to construct a hamiltonian cycle in G.

Here we show that the closure of a graph can be computed in $O(n^3)$ time. This improves the $O(n^4)$ bound given in [GoMi 84]. We make use of the F-heaps data structure, which has been used to improve the running time of many graph-theoretic algorithms. For details on F-heaps see [FrTa 84].

A natural generalization of the problem of computing the graph closure is considered. Given a graph G and a set of *admissible edges* E' (E' is a subset of the edges of $\overline{G}$), the *general closure* of G is the graph obtained from G by recursively joining pairs of non-adjacent vertices whose degree sum is at least n and the pair is in the set of admissible edges, until no such pair remains. We denote the general closure of G by $gc(G, E')$ where E' is the set of admissible edges. We show that computing the general closure of a graph is P-complete with respect to log space transformations, thereby giving evidence that the problem may be inherently sequential.

2 Algorithm

The graph is assumed to be represented as an adjacency matrix. The idea is to keep all the candidate edges in an F-heap choosing them one at a time to add to G. Each time we choose an edge to be added to G, the degrees of the two vertices incident to the edge increase and we update the corresponding edges of the F-heap which are incident to one of the two vertices.

Summarizing the steps of the Algorithm:
1)Make-heap(H).

2) For all pairs of vertices (u, v) do
 If $(u, v) \notin E$ then
 insert(uv,H) with key(uv)= n - deg(u) - deg(v);

3) While find-min(H) ≤ 0 do
 e := delete-min(H); {let $e = (x, y)$}
 For all $v \in V$ do
 If $(v, x) \notin E$ then decrease-key(1,vx,H);
 If $(v, y) \notin E$ then decrease-key(1,vy,H);
 od;

Add e to G;

od;

G now represents $c(G)$.

ANALYSIS:
Step 1 and 2 can clearly be implemented in $O(n^2)$ time. In step 3 there are atmost $O(n^2)$ iterations of the while loop since each edge gets added to G atmost once. Each iteration costs only $O(n)$ time since the amortized cost of $O(n)$ decrease-key operations is $O(n)$ (see [FrTa 84]).

The algorithm illustrates another use of the F-heaps data structure. The algorithm can also be implemented in $O(n^3)$ time without using F-heaps.

3 General Closure

Given $G(V, E)$ and a set of *admissible edges* E', the general closure of G is the graph obtained from G by recursively joining pairs of non-adjacent vertices whose degree sum is at least n and the pair is in the set of admissible edges, until no such pair remains. It is obvious that the algorithm to compute the closure can be easily modified to compute the general closure of a graph as efficiently. The only difference is that a set of admissible edges are specified and all new edges added to the graph are restricted to be in this set.

MONOTONE CIRCUIT VALUE PROBLEM:

A *monotone circuit* is defined to be a logical circuit $\alpha = (\alpha_1, \alpha_2, ..., \alpha_m)$, where each α_k is an input gate (having a value TRUE or FALSE), an AND gate ($\alpha_k = \alpha_i \wedge \alpha_j$, for some $i, j < k \leq m$) or an OR gate ($\alpha_k = \alpha_i \vee \alpha_j$, for some $i, j < k \leq m$). The *monotone circuit value problem* (also referred to as $MCVP$) is the problem of determining whether the last gate α_m receives the value TRUE or FALSE, given a truth assignment to the input gates. An example is given in figure 1.

The problem has been shown to be P-complete [Go 77], even when the input gates have fan-out 1 (they appear once as input to another gate) and each AND/OR gate has fan-out at most 2 [GSS 82].

TRANSFORMATION:

Computing the general closure of G can be turned into a decision problem by posing a question about the result of the algorithm, such as 'In the graph $gc(G. E')$ is there an edge (u, v) (assuming (u, v) is an admissible edge in E') ?'. We show that computing the general closure is P-complete by giving a log space transformation from the $MCVP$ to the decision problem described above.

For each instance of the $MCVP$ we construct a graph G^* in such a way that the circuit value of the considered instance is TRUE if and only if the general closure problem returns a 'yes' answer. Each gate of the circuit is represented by a subgraph.

For AND gate $k(k < m)$ with fan-out 2 $(\alpha_k = \alpha_i \wedge \alpha_j)$, we construct the subgraph as shown in figure 2. For OR gate $k(k < m)$ with fan-out 2 $(\alpha_k = \alpha_i \vee \alpha_j)$, we construct the subgraph as shown in figure 3. For a TRUE input gate k we construct the subgraph in figure 4. For a FALSE input gate k we construct the subgraph in figure 5. The number next to each vertex denotes the degree of the vertex. The number next to each edge denotes the deficiency of the edge, where the deficiency of admissible edge (u, v) is defined to be $deg(u) + deg(v)$ - 2N (N is defined later). Each dotted edge denotes an admissible edge. Real edges are not shown in the figure.

The input and output vertices of the gates are used, to connect the different subgraphs. If gate i is input to gate k, a vertex of degree N (which is an output vertex in the subgraph corresponding to gate i) is connected by an admissible edge to a vertex of degree N-1 (which is an input vertex in the subgraph corresponding to gate k). If the fan-out of a gate α_k is 1 then the output vertex in the subgraph corresponding to the gate is not attached to any vertex by admissible edges. We associate a TRUE (FALSE) value with the output of a gate if the output vertex has its degree increased due to the addition of an admissible edge incident to it (no admissible edges added, hence no change in the degree).

Let the total number of vertices in the graph generated by "putting" the subgraphs (corresponding to the gates) together be N. We refer to this set of nodes as V_1. Introduce N new nodes into the graph making the total number of nodes in the graph 2N. We refer to this set of nodes as V_2. For each node in $v \in V_1$ with degree d, introduce d edges from v to any d nodes in V_2. Introduce edges between all nodes in V_2.

It should be noted, that in the graph G^* (graph corresponding to the monotone circuit) initially, only the admissible edges corresponding to the TRUE input gates satisfy the condition for them to be added to the graph.

We now illustrate the simulation of each gate by the subgraph.

AND gate α_k:

1) Both inputs FALSE: Outputs are false since no admissible edges can be added. The degrees of the input vertices do not change, neither do the degrees of the output vertices change.

2) One input TRUE, other input FALSE: (see fig. 6(a) and 6(b)) The degree of the TRUE input vertex of gate α_k increases by 1, due to the addition of the edge e_1 (this edge gets added due to the increase in degree of the output vertex of gate α_j). Figure 6(b) illustrates the edges (shown in bold) which get added to the graph. Both the outputs are FALSE (since the degree of the output vertices does not change).

3)Both inputs TRUE: (see fig. 7(a) and 7(b)) Since the degree of vertex v_1 (fig. 7(b)) increases by 2 due to the addition of both e_1 and e_2 the edges e_3 and e_4 are added to the graph. This sets both the outputs to TRUE (degrees of output vertices increase).

OR gate α_k:

1) Both inputs FALSE: Clearly, both the outputs will be FALSE.

2) One input TRUE: (see fig. 8(a), 8(b), 8(c) and 8(d)) The degree of vertex v_1 (fig. 8(c)) increases by 1 and all the edges shown in fig. 8(d) get added to the graph.

It should be noted, that the addition of edges does not cause any change in the degree of the false input vertex. Both the outputs are set to TRUE.

3) Both inputs TRUE: (see fig. 9(a) and 9(b)) Clearly both outputs are set to TRUE.

It is obvious that the circuit has output TRUE if and only if the the degree of the output vertex of gate α_m increases (denoting a TRUE output).

Theorem 1 *Computing the general closure of a graph G is P-complete with respect to log space reductions.*

Proof: Since the monotone circuit value problem is P-complete, and the entire transformation shown above can be done in logarithmic work space, the general closure problem is P-complete. □

4 Acknowledgements

I would like to thank Estie Arkin for introducing me to this problem and for extremely useful discussions. Thanks also to Shyam Kapur for commenting on an earlier draft of the paper.

References

[BoMu 76] J.A.Bondy and U.S.R.Murty, 'Graph Theory with Applications', *North-Holland*, (1976).

[FrTa 84] M.L.Fredman and R.E.Tarjan, 'Fibonacci heaps and their uses in improved network optimization algorithms', *Proceedings of FOCS conference*, (1984), pp 338-346.

[Go 77] L.M.Goldschlager, 'The monotone and planar circuit value problems are log space complete for P', *SIGACT News*, vol 9, 2, summer (1977), pp 25-29.

[GSS 82] L.M.Goldschlager, R.A.Shaw and J.Staples, 'The maximum flow problem is log space complete for P', *Theoretical Computer Science*, 21, pp 105-111.

[GoMi 84] M.Gondran and M.Minoux, 'Graphs and Algorithms',.

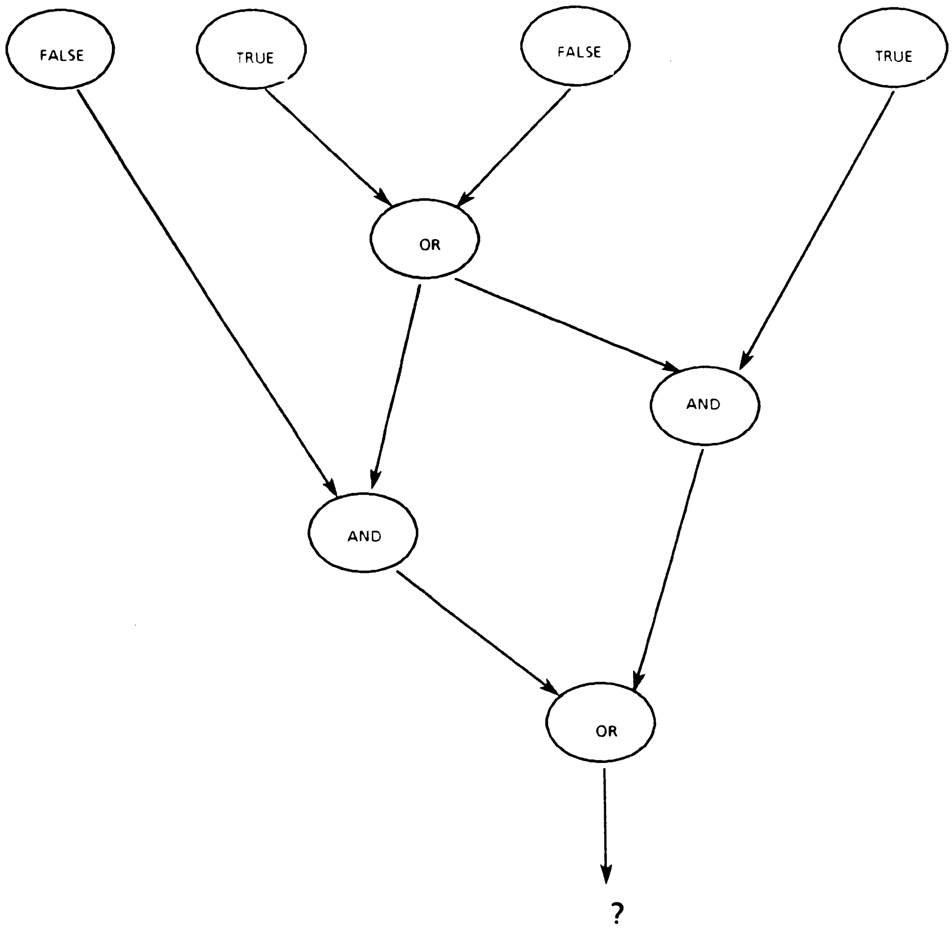
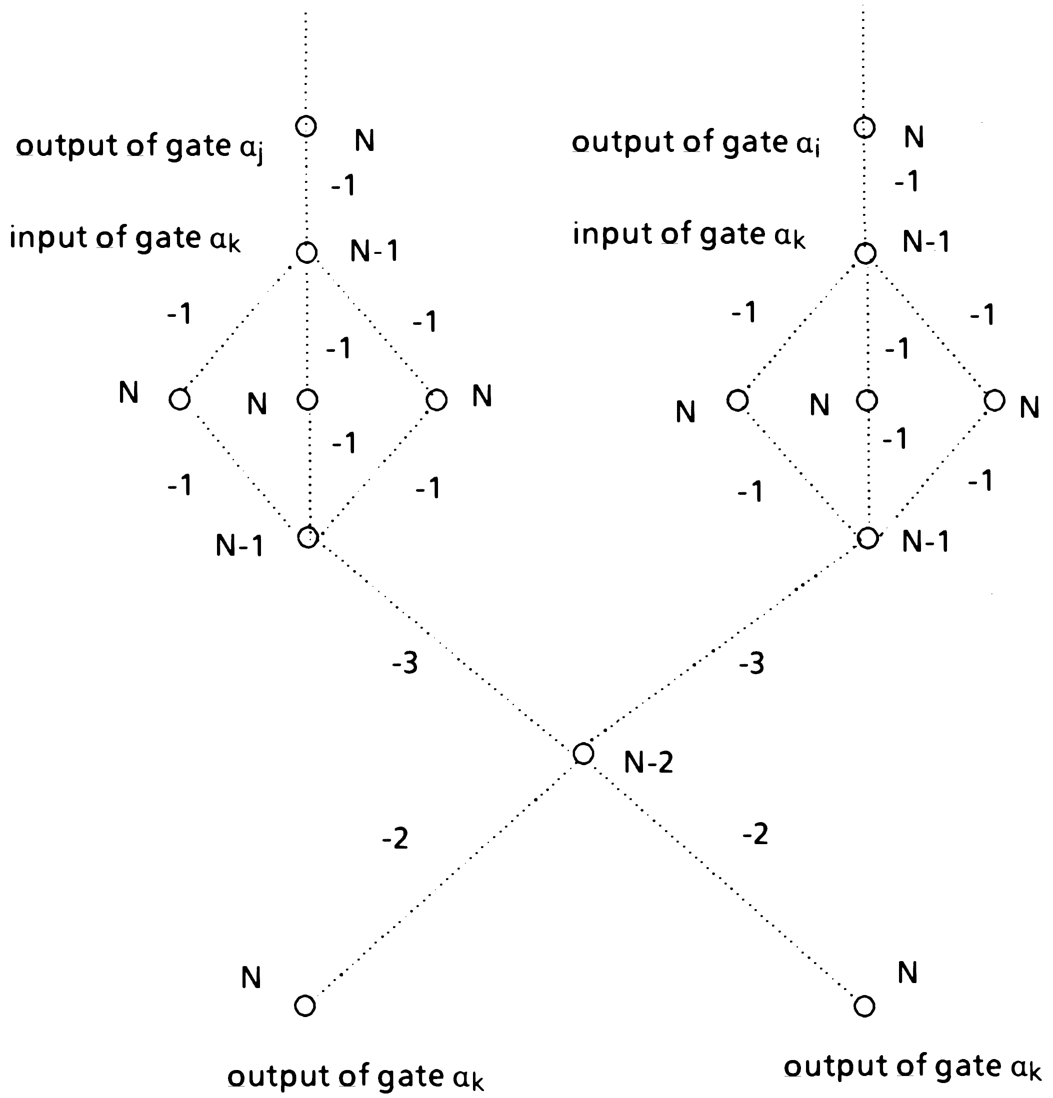


Figure 1. A logical circuit

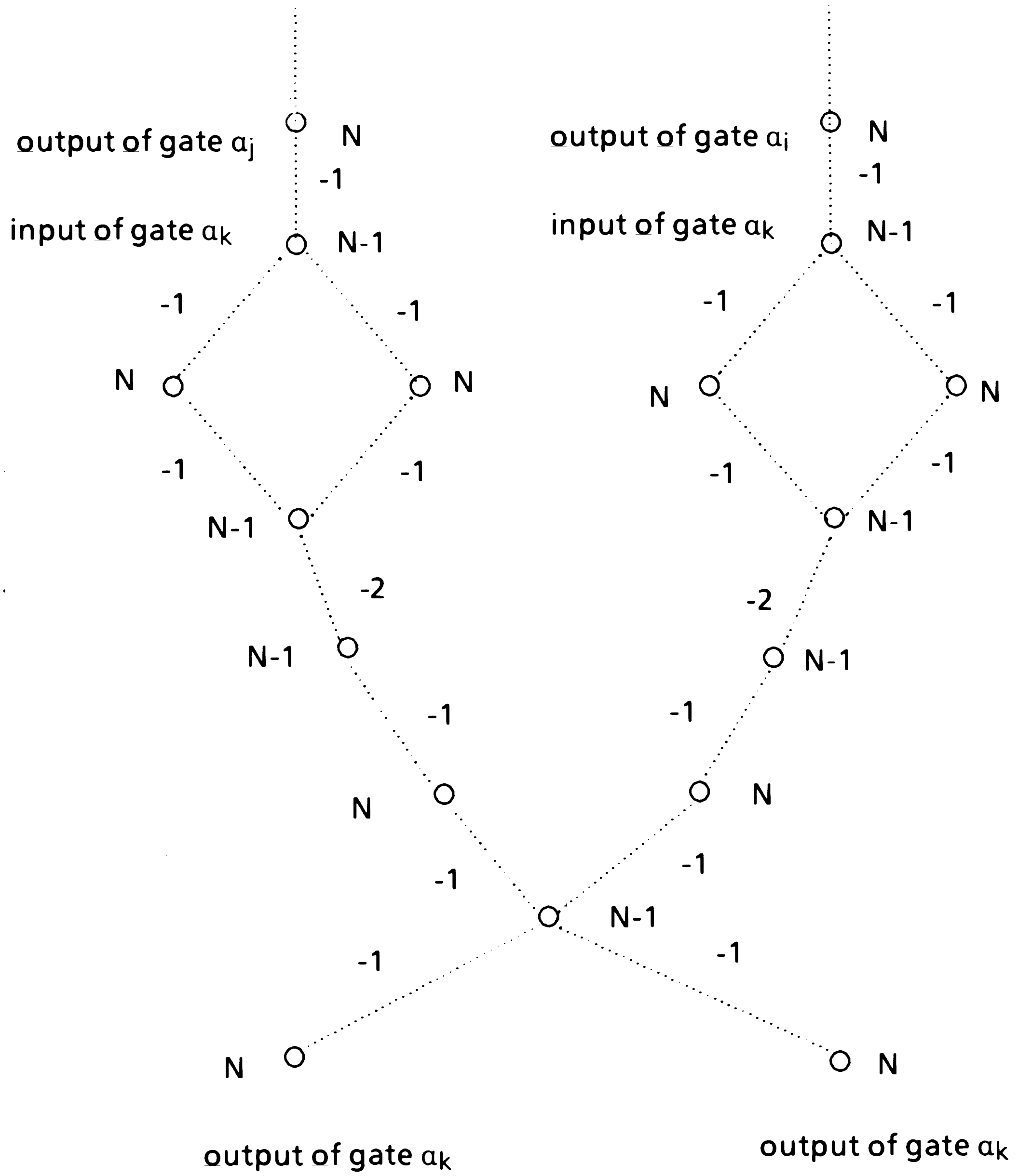


-x next to an admissible edge denotes the deficiency of the edge

X next to a vertex denotes the degree of the vertex

.... denotes an admissible edge

Figure 2. AND gate α_k



-x next to an admissible edge denotes the deficiency of the edge

X next to a vertex denotes the degree of the vertex

.... denotes an admissible edge

Figure 3. OR gate α_k

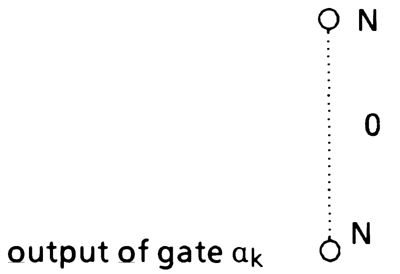
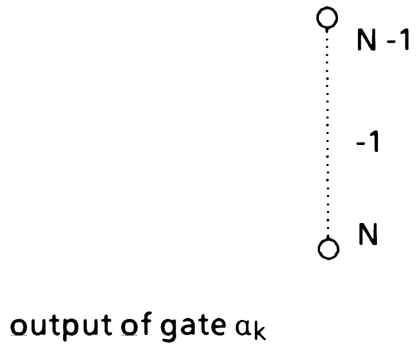


Figure 4. TRUE gate α_k



-x next to an admissible edge denotes the deficiency of the edge

X next to a vertex denotes the degree of the vertex

.... denotes an admissible edge

Figure 5. FALSE gate α_k

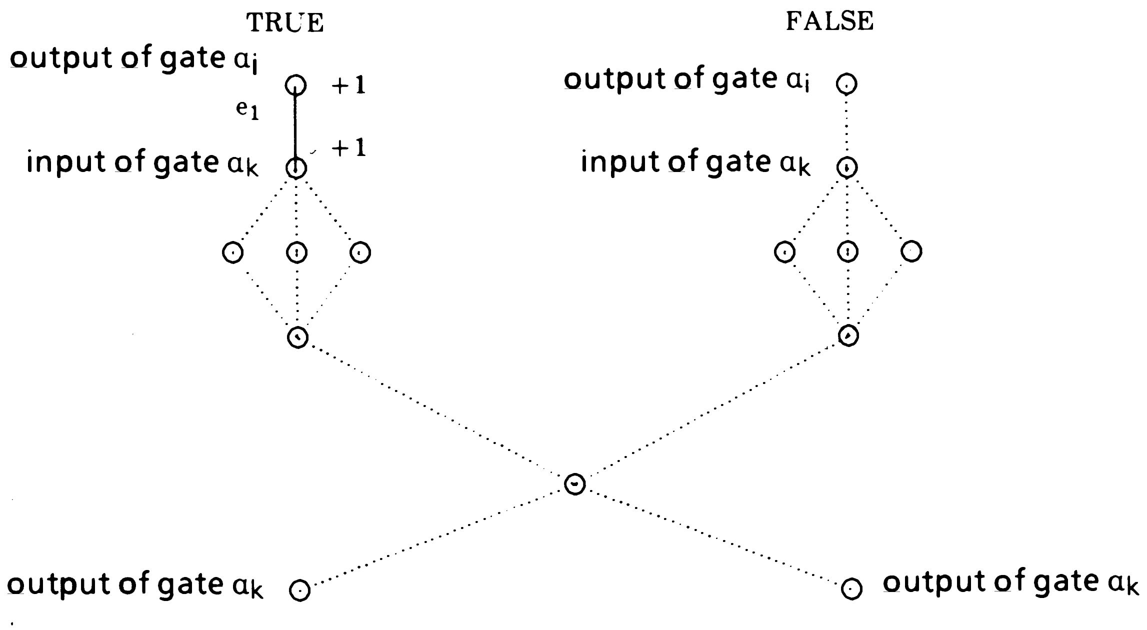


Figure 6(a). AND gate α_k

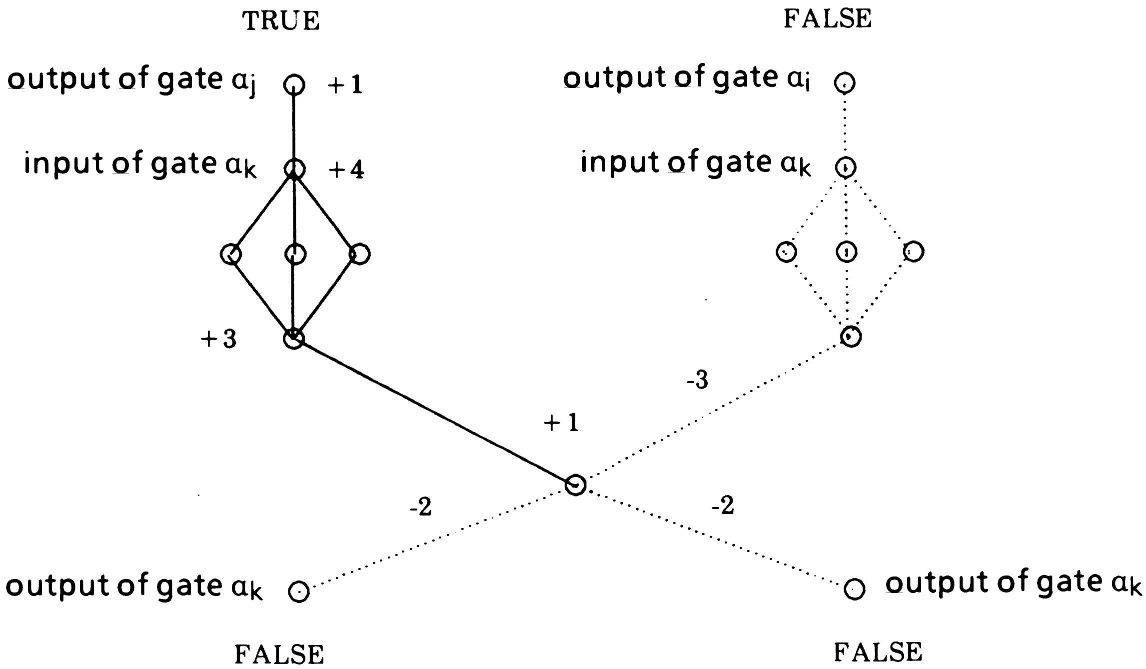


Figure 6(b). AND gate α_k

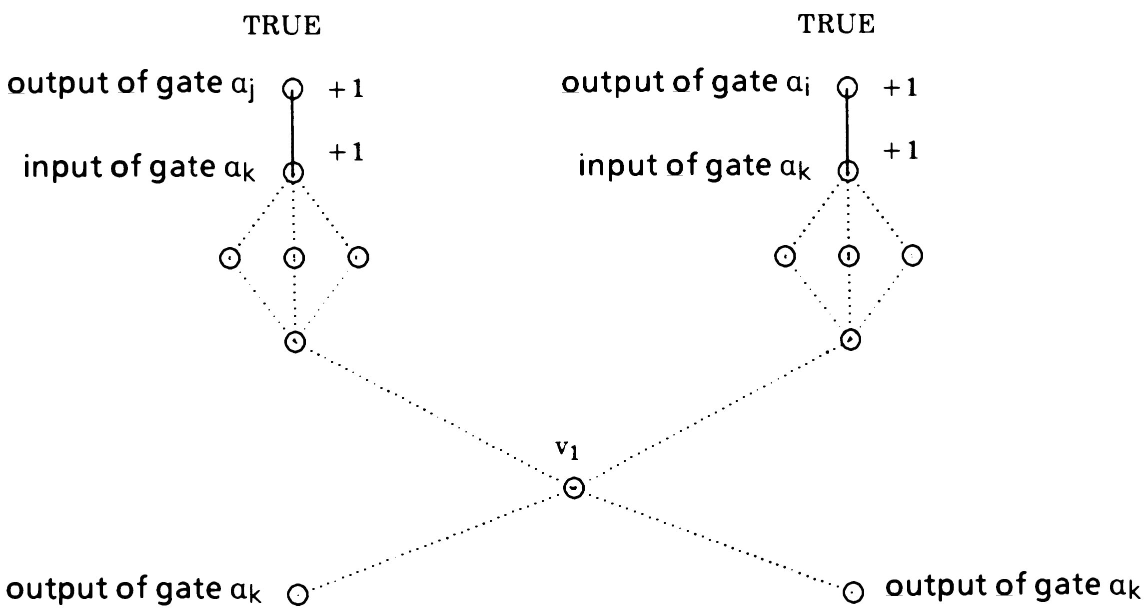


Figure 7(a). AND gate a_k

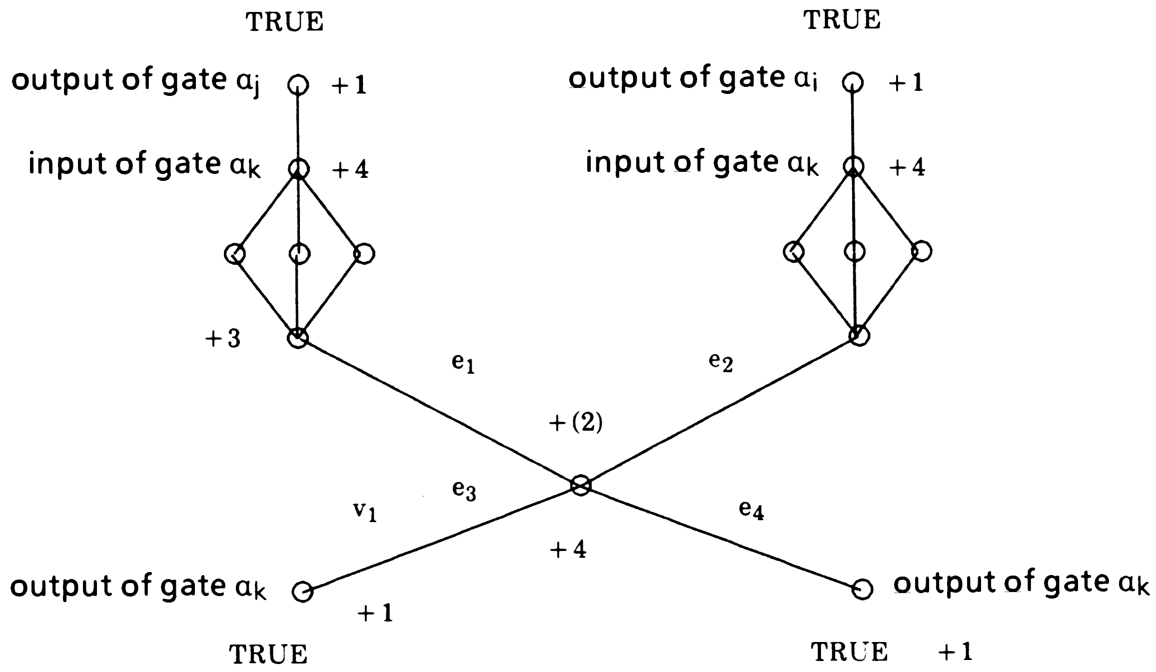


Figure 7(b). AND gate a

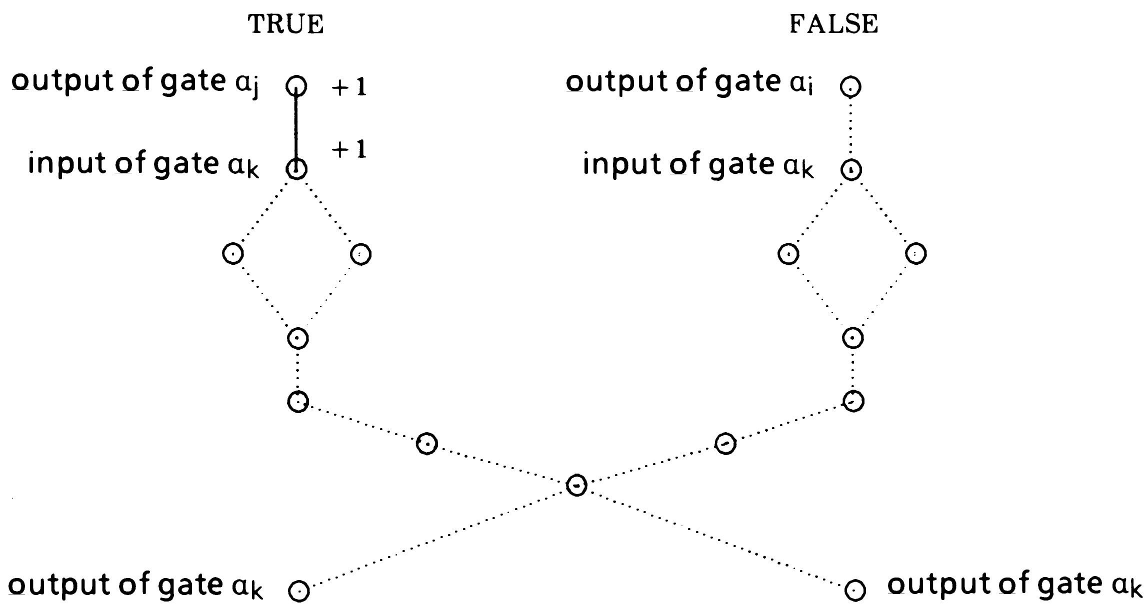


Figure 8(a). OR gate α_k

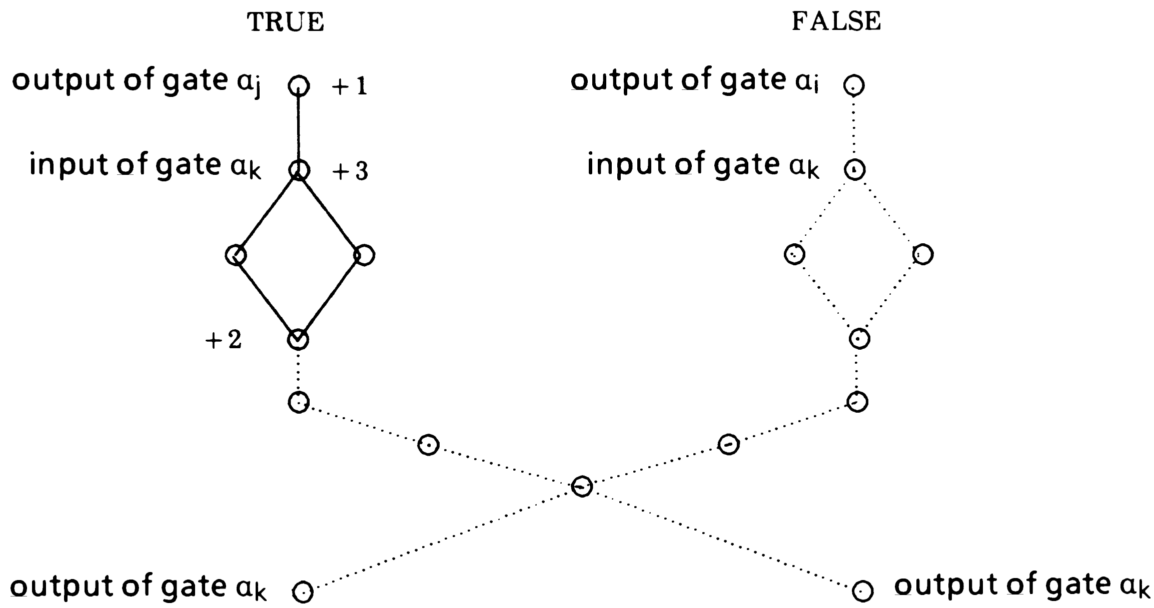


Figure 8(b). OR gate α_k

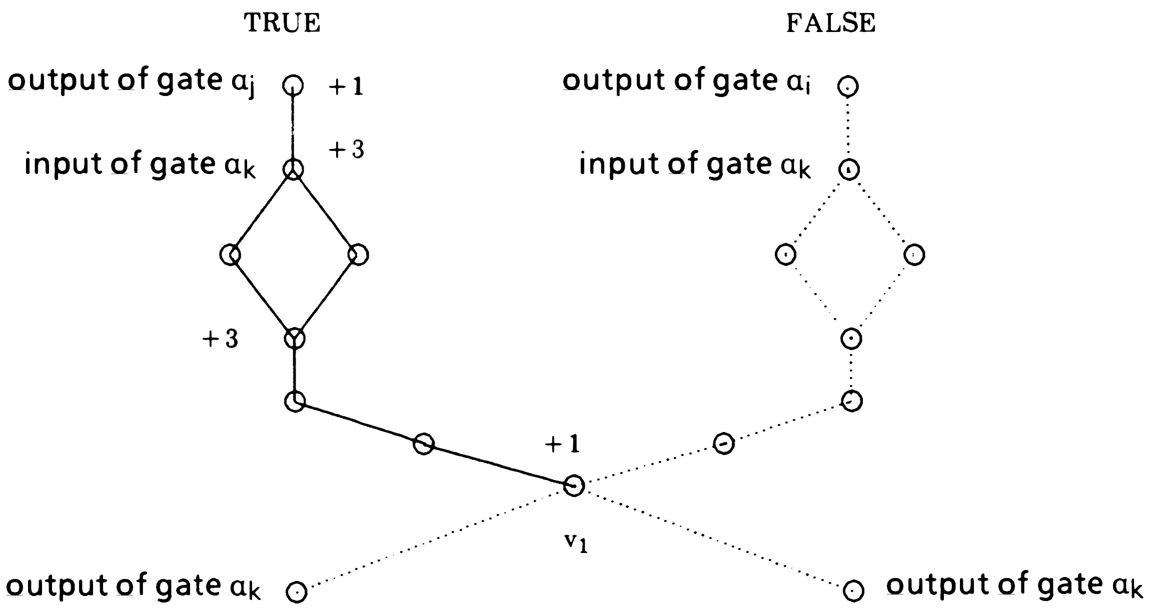


Figure 8(c). OR gate a_k

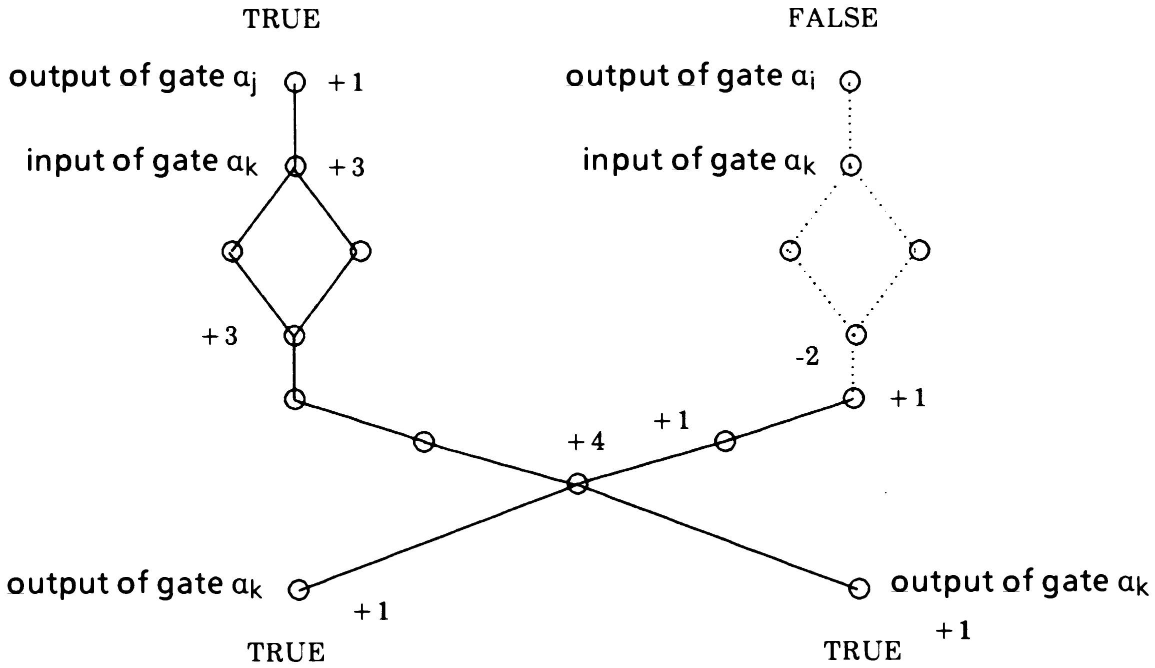


Figure 8(d). OR gate a_k

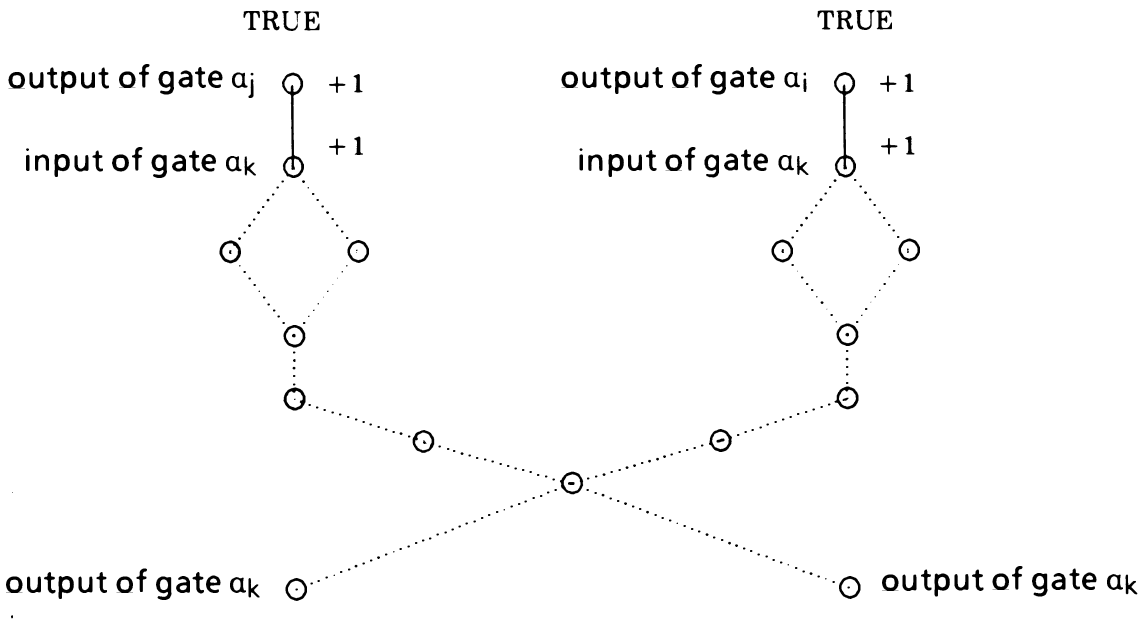


Figure 9(a). OR gate a_k

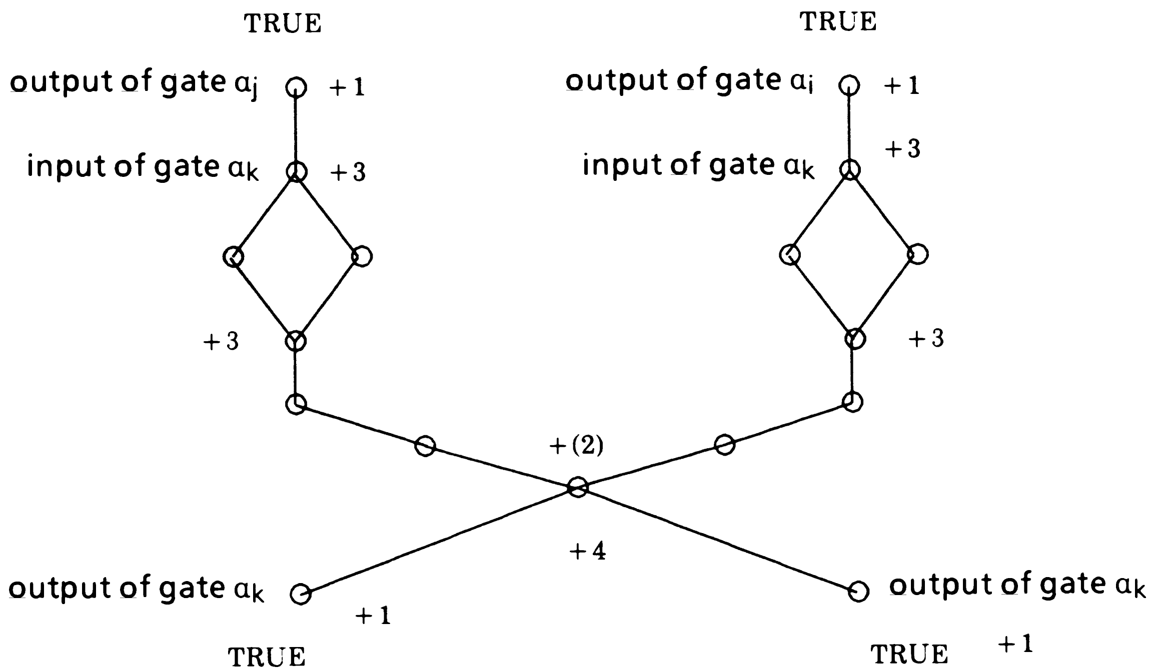


Figure 9(b). OR gate a_k