A METHOD FOR LINEARLY SOLVING COMBINATORIAL
OPTIMIZATION AND RELIABILITY PROBLEMS ON
SERIES PARALLEL DISTRIBUTED NETWORK

by

E. Korach¹, S. Onn², D. Rotem³

Technical Report #402

February 1986

1 Computer Science Dept., Technion – IIT, Haifa 32000, Israel.
2 Faculty of Electrical Engineering, Technion – IIT, Haifa 32000, Israel.
3 Computer Science Dept., University of Waterloo and Lawrence Berkeley Labs.
A METHOD FOR LINEARLY SOLVING COMBINATORIAL OPTIMIZATION AND RELIABILITY PROBLEMS ON SERIES PARALLEL DISTRIBUTED NETWORK

(Extended Abstract)

by

E. Korach\textsuperscript{1}, S. Onn\textsuperscript{2}, D. Rotem\textsuperscript{3}

1. INTRODUCTION

In this paper we deal with the class of Series Parallel Graphs (SPG's) which is a subclass of the planar graphs, and includes as a subclass the outerplanar graphs and therefore also the circuits. The recursive structure of SPG's is attractive and have generated a lot of research, e.g. [DUF], [VALD], [SHI]. A variety of NP-complete problems were proved to be linear time solvable on series parallel graphs [TAK]. Other linear time algorithms appeared in [KOR] - for the chinese postman problem and for the multicommodity flow problem, and in [BOU] for the maximum weight independent set.

The recursive structure of Series Parallel Graphs also makes it convenient in a distributed network to add or delete processors from the network.

We present a general method for the design of distributed algorithms in SP-graphs that can be applied to solve the following classes of problems in the distributed model described below. These algorithms achieve linear message complexity. The problems are:

1) Leader electing;

\textsuperscript{1} Computer Science Dept., Technion - IIT, Haifa 32000, Israel.
\textsuperscript{2} Faculty of Electrical Engineering, Technion - IIT, Haifa 32000, Israel.
\textsuperscript{3} Computer Science Dept., University of Waterloo and Lawrence Berkeley Labs.
2) Minimum weight spanning tree;

3) The reliability of communication and the vertex connectivity between all pairs of vertices;

4) Optimal vertex disjoint-routing;

5) Hard combinatorial optimization problems such as the chinese postman problem, multicommodity flow, maximum matching and maximum weight independent set.

We introduce a new assumption concerning the distributed model which is of major importance in this paper and we believe it is achievable with modern technology. According to this assumption a processor can relay two of its edges one to another allowing a free passage of messages along this two edges path. In this way a processor does not have to deal with messages whose destination is other than itself, and saves both processor time and message delivery time. Therefore, a message passing in such a relayed path will be counted only once. Still, a primitive mechanism of $v$ checks the type of each message passing this route, and if one of a set of types ordered to the mechanism in advance by the processor is detected, the message is stopped and the processor is announced. In this case, even if the message after checking by the processor is sent along the relayed path without any processing in $v$ and the relaying continues, the message is counted, of course, as two messages. Hence, obvious comparison of our complexity results with others will be incorrect (e.g., using our model, the number of messages required for electing a leader in a ring becomes $O(|V|)$ using the following trivial algorithm: "on waking up send your identity to one arbitrary direction; on receiving higher identity pass it and become a relay; on receiving your identity you are the leader").

Next, we define a distributed Series Parallel Reduction Construction process (SPRC) as follows.

Starting from a two connected SP network, we distributively reduce the network to $k_2$, collecting and processing information concerning the problems to be solved, in linear number of messages sent. The reduction process follows the definitions in Section 2, but performed simultaneously in many focuses.
After reduction has terminated and construction has begun, a left-right order is established among the processors, based on some properties of SPG's (Section 3): Each processor knows which of his neighbors are to his left and which to his right. There are two possible methods of construction: the "totally left to right" construction which is demonstrated in section 4.2; And the "reverse order" construction. The latter one follows the definitions in section 2, but not necessarily performed in the exact reverses order of the reduction process. This method is described in section 3. This completes SPRC.

In order to describe the number of messages sent, we use the following notations. Let \( G = (V, E) \) be the two connected SPG describing the network; parallel (multiple) edges are allowed. We denote by \( E_s \) the set of edges obtained from \( E \) by replacing parallel edges by one representative edge. Then one could prove that \( |E_s| < 2|V| \). Also, we can assume (however, we will avoid doing so) that \( |E| \leq k |E_s| \), where \( k \) is a constant, which will imply \( O(|E_s|) = O(|V|) \).

Based on the SPRC process as described, we solve one by one or simultaneously, in a unified manner, a variety of problems which are classified below. It is easy to see that \( \Omega(|E|) \) is a trivial lower bound for all the problems discussed here. Therefore, we may assume that in an initialization process, every processor sends messages across all its incident edges and thereafter knows the identities of the processors across them. This requires \( 2|E| \) messages. Therefore, from now on we assume that such an initialization process has been performed, and all the algorithms require \( O(|V|) \) additional messages.

1. Leader Electing.

After a leader is elected, it is easy to decide upon two vertices \( x, y \) such that \( (V, E, x, y) \) is Two-Terminal Series Parallel (TTSP). From now on, any \( x-y \) pair is assumed to be such a pair.

2. Minimum weight spanning tree, given a weight function \( w: E \rightarrow Q \), weights not necessarily distinct.
3. Reliability of communication and connectivity between \( x \) and \( y \).
   a. Determining the vertex connectivity \( k \) (the number of vertex disjoint paths). In fact, by solving the problem for one \( x-y \) pair, we get as a by-product the connectivity for all pairs \( v_1,v_2 \in V \) (not necessarily TIST pairs).
   b. Determining the number of different simple \( x-y \) paths. This problem is, for general graphs, complete in \#P time computation complexity family, whose members are at least as hard as any NP-complete problem [VAL].
   c. Determining the number of simple edge disjoint \( x-y \) paths.
   d. Given a function \( P: E \rightarrow [0,1] \), where \( p(e) \) is the probability of \( e \) not to fail in a known period of time, determining the probability of \( x-y \) not becoming disconnected during this time period. This problem is also in \#P-complete [VAL].

4. Optimal vertex disjoint routing. Given \( m \leq k \) as in 3.a, physically routing \( m \) vertex disjoint \( x-y \) paths which are optimal by one of the following criteria, and computing the value of the criteria for each path:
   a. Minimum probability of path disconnection where \( p \) is as in 3.d.
   b. Minimum path weight where \( w \) is as in 2.

5. Hard combinatorial optimization problems.
   a. Given \( T \subseteq V \), \( |T| \equiv 0 \pmod{2} \), and \( w: E \rightarrow \mathbb{Q}^+ \), determining a minimum weight \( T \)-join. This is a generalization of the chinese postman problem, obtained when \( T = \{v \mid (v \in V) \land (d(v) \equiv 1 \pmod{2})\} \).
   b. Finding a maximum cardinality matching.
   c. Given \( w \) as in a. and \( F \subseteq E \), finding a solution to the multicommodity flow if one exists and indicating if not, where \( w(f) \), \( f \in F \), is the demand between \( f \)'s two end vertices and \( w(e) \), \( e \in E \setminus F \), is the capacity of \( e \).

We remark that all pairs \( x-y \) such that \( (V,E,x,y) \) is TTSP, for which the problems above could be solved, could be detected by a process which uses \( O(|V|^2) \) messages.

In Section 2, we define SPG's and TTSP's and discuss some of their properties. In Section 3, we describe the basic construction-reduction process (SPRC) in the presence of two distinct \( x-y \) vertices. In Section 4 we give some examples demonstrating
our method for distributively solving problems on SPG's. Finally, concluding remarks are given in section 5.

2. PRELIMINARIES

2.1 Definitions and Properties of Series Parallel Graphs (SPG's)

In this paper we are interested only in two-connected SPG's; we do not allow self-loops \((V, V)\) edges, \(v \in V\). We start by giving some definitions.

1. We say that:
   a. \(G'\) is obtained from \(G=(V,E)\) by a "series construction" if 
      \[ G'=(V \cup \{v\}, E \cup \{(u,v),(v,w)\} \setminus \{(u,w)\}) \] 
      and \(u,w \in V, v \notin V, (u,w) \in E\).
   b. \(G'\) is obtained from \(G=(V,E)\) by a "series reduction if 
      \[ G'=(V \setminus v, E \setminus \{(u,v),(v,w)\}) \] 
      and \(v,w,u \in V, \deg(v)=2, (u,v),(v,w) \in E\).
   c. \(G'\) is obtained from \(G=(V,E)\) by a "parallel construction" if 
      \[ G'=(V,E \cup \{(u,v)\}) \] 
      and \(u,v \in V, (u,v) \in E\) (an edge doubling).
   d. \(G'\) is obtained from \(G=(V,E)\) by a "parallel reduction" if 
      \[ G'=(V,E \setminus (u,v)) \] 
      and \(u,v \in V\) and there are at least two \((u,v)\) edges in \(E\).

2. A two connected SPG is defined recursively, as follows:
   a. \(c_2\) is a two connected SPG;
   b. Any graph obtained from a two connected SPG by either a series or a parallel construction is a two connected SPG.

Next, we describe some properties without proofs. For more detailed discussion see [BUF], [VALD], [SHI], [KOR].

1. A two connected SPG is two vertex connected.

2. If \(G\) is a two connected SPG, then so is \(G'\) which is obtained from \(G\) by either series or parallel reduction. From this it follows:
Some Properties:

1. Let $G = (V,E,x,y)$ be a two connected SPG. If $(x,y) \in E$ then $G$ is TTSP.

2. Let $G = (V,E)$ be a two connected SPG. For every $z,y \in V$ such that $G = (V,E,z,y)$ is TTSP, the number of vertex disjoint $z-y$ paths in $G$ equals the number of components in $G[V \setminus \{z,y\}]$ (induced subgraph) plus the number of $(z,y)$ edges in $E$.

3. Let $G = (V,E)$ be a two connected SPG. For every $z,y \in V$ such that $(V,E,z,y)$ is TTSP, there exists a sequence of series and parallel reductions that reduces $G$ into $c_2$ where $V(c_2) = \{x,y\}$.

3. A DISTRIBUTED SERIES-PARALLEL REDUCTION CONSTRUCTION PROCESS (SPRC)

In this section we briefly describe the SPRC and outline the leader electing process. We introduce the "2-path" term and give intuitive reasoning for the proof of linear number of messages' complexity.

As mentioned in the Introduction, SPRC follows the definitions of series and parallel reductions given in Section 2. Of major importance, however, is the "2-path" term defined as follows:

**Definition:** A simple path $p = (v_1,v_2,...,v_k)$ is called a "2-path" if the following hold:

a. $d(v_i) = 2$, $1 < i < k$, $k \geq 3$.

b. The condition: $v_j$ has received a message from $v_i$ at a time in which $(d(v_i) = 2) \land (d(v_j) > 2)$ holds exactly in the following two cases:

1) $(i = 2) \land (j = 1)$;

2) $(i = k - 1) \land (j = k)$.

Note that the construction of a specific 2-path is completed when the two ends are fixed by receiving the messages as in b.

We assume that in the beginning of SPRC all vertices are in "AWAKE" status, which means that each vertex already knows the identity of the neighbor across each edge incident to it. This is achieved, as mentioned in the introduction with $2|E|$ messages sent.
3. Any two connected SPG can be reduced by a sequence of series and parallel reductions into \( c_2 \). Furthermore:

4. Let \( G = (V,E) \), then, for every \( x,y \in V \) such that \( (x,y) \in E \) there exists a sequence of series and parallel reductions that reduces \( G \) into \( c_2 \) where \( V(c_2) = \{x,y\} \).

5. A two connected graph \( G \) is a two connected SPG if and only if it contains no subgraph which is a homeomorph of \( k_4 \) [DUF].

Finally let us denote by \( G_0, G_1, ..., G_k \) a sequence of graphs obtained by a sequence of series and parallel reductions, starting with \( G_0 = G \), such that \( G_{k-1} = c_2 \) and \( G_k = k_2 \).

Let \( I = \{0,1, ..., k\} \). We denote, \( i \in I \), \( G_i = (V_i,E_i) \). We define a function

\[
H: \bigcup_{i \in I} E_i \to \{G' \subseteq G\}
\]

recursively as follows:

Let \( e \in E_0 \) be a \( (v_1,v_2) \) edge. Then \( H(e) = \{\{v_1,v_2\},\{e\}\} \).

If \( e \in E_{i+1} \) has replaced \( e_1,e_2 \in E_i \) either by a parallel or by a series reduction, then

\[
H(e) = (V(H(e_1)) \cup V(H(e_2)), E(H(e_1)) \cup E(H(e_2))).
\]

**Lemma 2.1**: Let \( j \in I, e,e' \in E_j, e \neq e', e \) be a \( (v_1,v_2) \) edge, \( e' \) be a \( (v_1',v_2') \) edge.

Then:

a. \( V(H(e)) \cap V(H(e')) \subseteq \{v_1,v_2\} \cap \{v_1',v_2'\} \)

and

\[
E(H(e)) \cap E(H(e')) = \phi.
\]

b. \( \forall u \in V(H(e)) \setminus \{v_1,v_2\} \), \( \forall w \in V(H(e')) \setminus \{v_1',v_2'\} \) no \( (u,v) \) edge exists in \( E \).

c. \( H(e) \) is connected.

d. \( V = \bigcup_{e \in E_j} V(H(e)); E = \bigcup_{e \in E_j} E(H(e)) \).

**Proof**: Directly follows from the definitions of series and parallel reductions.

### 2.2 Two Terminal SPG (TISP)

**Definition**: \( G = (V,E,x,y) \), where \( x,y \in V \), is a "two connected two terminal SPG (TISP)" if \( G' = (V,E \cup \{(x,y)\}) \) is a two connected SPG.
Parallel reductions will be performed with no messages sent. Each of the end vertices of parallel edges will choose one representative edge which is best by some criterion, (say highest probability of no fail) and the variables referred to this edge (probability, weight, capacity, etc.) as required by the specific problem that we solve will be "parallel equivalent" as defined to that problem. An agreement between neighbors \( v \) and \( u \) upon the selected edge between them will be achieved later, when \( v \) has received a message from \( u \) on \( u \)'s selected edge and vice-versa as it is described below.

Before turning to describe series-reductions, we introduce the notion of "SR-information". Suppose \( v \) has \( d(v)=2 \) and its two incident edges are \( e_1=(v,v_1) \), \( e_2=(v,v_2) \). At the end of the reduction, a new "series equivalent" edge (by means of the solved problem) \( e=(v_1,v_2) \) should replace \( e_1, e_2 \) and \( v \) should be vanished. In order that \( v_1, v_2 \) be able to update their local neighborhood, \( v \) should send on \( e_1 \) information concerning \( e_2, v_2 \) and internal \( v \)-information if required, and similarly on \( e_2 \). This is the SRI notion.

We present here only the series reductions that are applied in the case of TTSP with distinct vertices \( x \) and \( y \) for clarity reasons although a unified approach for both TTSP and SP graphs can be defined.

Series reductions in our distributed algorithm will be performed on a complete 2-path \( v_1, \ldots, v_k \), rather that on each of its internal vertices. The main reason for this approach is that all successive pairs \( v_i, v_{i+1} \), \( 1 \leq i < k \) along the 2-path should, first of all, agree on the selected edge \( e_i \) between them, and then each could relay its two incident edges. Otherwise, \( O(k) \) SRI's could traverse the path, which could have \( O(k) \) yet unrelayed vertices, ending up with \( O(k^2) \) messages sent.

Hence, the 2-path reduction is performed as follows. Each of the internal vertices \( v_i \), discovering \( d(v_i)=2 \), sends his SRI on his two selected edges. He stores any of the SRI's he receives (at most one on each edge). If the SRI from \( v_{i-1} \) has been received on \( (v_{i-1},v_i) \) edge other than the one selected by \( v_i \), and \( v_{i-1}>v_i \), then \( v_i \) agrees to \( v_{i-1} \)'s choice. The same happens between each adjacent pair on the path. The end vertices
$v_1, v_k$ accept the selection of edges made by $v_2, v_{k-1}$ respectively. Let $e=(v_1, v_2)$ be the edge selected by $v_2$. Because arbitrary long delays on the messages in the links could occur, the end vertex $v_1$, not necessarily agrees with $v_2$, at this moment, on the number of parallel edges between them. Since $v_2$ knows already the number of vertex disjoint paths between them on $H(e)$, it includes this number in its SRI, and $v_1$ will wait until it counts the same number of vertex disjoint paths in $H(e)$. Only then it will respond by sending on $e$ a message $ST(v_1)$. Similarly, $v_k$ sends $ST(v_k)$. Each internal vertex passes on these messages. Only after an internal vertex has received stop-message from each direction and passed it on, he passes the two SRI's he had received, and then relays its two incident edges (which are, in this stage, guaranteed to be agreed upon with his neighbors) one to another. This process assures that the secondly sent SRI's, that follow the stop messages, will meet in their way to the end of the path only relayed vertices. This is so, because of the FIFO assumption on the links, and therefore their whole way to the end of the path will be counted as one message only.

The end vertex $v_1$ receives in the following order: $SRI(v_2), ST(v_k), SRI(v_3), ..., SRI(v_{k-1})$. He detects the end of the path reduction by comparing "his new adjacent vertex" contained in each SRI with $v_k$. A similar argument holds for $v_k$.

Each of the internal vertices is "vanished" from the network, and will continue the algorithm only after receiving a construction message on some edge.

The number of messages sent in such a 2-path reduction is exactly $6 \cdot (k-2)$, and exactly $k-2$ vertices vanishes. Therefore, the total number of messages sent in the reduction process, terminated in $k_2$, is $6 \cdot (|V| - 2)$.

In the case that we start with $G=(V,E,x,y)$, a TISP with two distinct vertices, then we finally have $k_2=\{x,y\}, \{(x,y),x,y\}$ and we are guaranteed $x, y$ will not vanish and the reduction will correctly terminate. But if this is not the case and no distinct vertex exists, then an illegal 2-path, which is a ring, could be performed, upon which no vertex is an end vertex, and a deadlock could occur. Therefore, we have to insert, in this case, a leader electing process for a bidirectional ring with no sense of direction, which
in our model would not change the order of number of messages sent.

We do not go down into details concerning the data management in the processor. We assume every processor stores during the reduction all information concerning his changing neighborhood, which finally includes two incident edges $e_1, e_2$ and two adjacent vertices, by means of $H(e)$ for all $\cup_{i \in I} E_i$ such that, in some $G_i$, $e$ is incident to it. therefore, its physical edges are divided into two sets, $E(H(e_1)), E(H(e_2))$.

The reduction process terminates when the network is reduced to $k_2=\{x,y\}, \{e\}$. Then $x$ starts the construction process. This process reconstructs $G$ by a sequence of graphs $G_k', G_{k-1}', ... G_0'$, where $G'_i$ is obtained from $G_{i+1}'$ by a single parallel or series construction. We have $G_k'=G_k$ and $G_0'=G_0=G$, but the order of constructions, because the process is distributed, is not necessarily the same as the reductions order. In each construction, two edges $e_1, e_2 \in G_i'$ replace one $(v_1, v_2)$ edge $e \in G_{i+1}'$. The vertex which is responsible for such a construction is $v_1$ which is left to $v_2$.

The left right order is established during the construction as follows. After $x$ makes some parallel constructions (at least one, $G_{k-1}'=G_k$) it performs a series construction by sending a message to a vertex $v \neq y$. This vertex recognizes itself as the destination of the message and starts construction. If the message was received on $e_1$, all edges in $E(H(e_1))$ are left edges; otherwise they are right edges.

The left-right order is well defined and no vertex can receive construction messages along edges both in $E(H(e_1))$ and $E(H(e_2))$, otherwise $G = (V, E, x, y)$ is not TTSP (Fig. 3.1).

As in the reduction, no messages are sent in a parallel construction; Also, a complete 2-path $p=(v_1, v_2, ..., v_k)$ is triggered to start construction by its left end vertex $v_1$, which sends messages to $v_k, ..., v_2$ in that order. The right end vertex, $v_k$, is guaranteed to be already in the construction process.

In this way, each vertex receives exactly one construction message, and $|V|$ messages are required to complete the construction process. Therefore, the whole SPRC requires $O(|V|)$ messages.
a. The number of different simple $v_1, v_2$ paths in $H(e)$ is $n(e)$.

b. The number of simple edge disjoint $v_1, v_2$ paths in $H(e)$ is $m(e)$.

c. The probability of $v_1, v_2$ not becoming disconnected during $[0,t]$ in $H(e)$ is $p(e)$.

**Proof:** Omitted.

The functions $p, n, m$ can be distributively computed. Each vertex $v$ computes the value of its incident edges in $\bigcup_{i \in I} E_i$ (until it becomes of degree two (if $v \neq x,y$) in a series reduction). In a two path the end vertices compute the series reductions of the whole path. Therefore, we have the following theorem.

**Theorem:** When the distributed reduction terminates with $G_k = k_2 = \{x,y\}, \{e\}$ the solutions to problems 1, 2 and 3 are given by $n(e), m(e),$ and $p(e),$ respectively.

**Proof:** Recall from Lemma 2.1 d. that $H(e) = G$. Then, from Lemma 4.1 the proof follows.

### 4.2 Minimum Weight Spanning Tree

We restrict ourselves to a general description of the solution. The reduction computation is performed analogously to the computations in 4.1. We are given a weight function $\mathcal{W}: E \rightarrow \mathbb{Q}$ and define recursively using the sequence $G_0, \ldots G_k$ a function $w: \bigcup_{i \in I} E_i \rightarrow \mathbb{Q}$ as follows: $\forall e \in E_0, w(e) = \mathcal{W}(e)$. If $e$ is the result of a parallel reduction of $e_1$ and $e_2$ then $w(e) = \min(w(e_1), w(e_2))$ and if $e$ is the result of a series reduction of $e_1$ and $e_2$ then $w(e) = \max(w(e_1), w(e_2))$.

We have chosen to demonstrate this example using the "totally left to right" construction process. (In this construction method, $|E_2|$ messages are required, which is still $O(|V|)$).

In this process we need four types of messages:

**<CT>:** *Construct tree.* The receiving vertex is on the chosen $x-y$ path of the tree.

This vertex chooses a minimal path on its right and sends on it a message **<CT>**.
4. EXAMPLES

In this abstract we do not give a formal description of the algorithms and we describe the SPRC computations and types of messages required only for a sample of the problems described in the introduction. We assume, that \( G = (V, E, x, y) \) is a two connected TTSP.

4.1 Connectivity and Reliability Computations

We want to compute the following:

1. Number of different simple \( x - y \) paths (for general graphs this problem is in \#P-complete [VAL]).

2. Number of simple edge disjoint \( x - y \) paths.

3. Probability of \( x - y \) not becoming disconnected in a given interval of time \([0, t]\) given a function \( P: E \rightarrow [0, 1] \) where \( P(e) \) is the probability that \( e \) will not fail during \([0, t]\).

We use the sequence \( G_0, \ldots, G_k \), as before and define three functions, \( p: \bigcup_{i \in I} E_i \rightarrow [0,1], n,m: \bigcup_{i \in I} E_i \rightarrow \mathbb{Z}_+ \) recursively as follows:

- If \( e \in E_0, p(e) = P(e) \), \( n(e) = m(e) = 1 \). If \( G_{j+1} \) is obtained from \( G_j \) by a parallel reduction, \( e \in E_{j+1} \) replaces \( e_1, e_2 \in E_j \). Then,
  \[ p(e) = 1 - (1 - p(e_1))(1 - p(e_2)), n(e) = n(e_1) + n(e_2), m(e) = m(e_1) + m(e_2). \]

- If the reduction is series, then:
  \[ p(e) = p(e_1) \cdot p(e_2); n(e) = n(e_1) + n(e_2); m(e) = \min(m(e_1), m(e_2)). \]

**Lemma 4.1:** Let \( j \in I, e \in E_j \) be a \((v_1, v_2)\) edge. Then
<CF, w>: Construct forward. The path along which such a message is sent has an edge that should not be a tree edge. This edge has weight \( w \) which was maximal on the path; \( w \) traversed the path by series reductions. Each vertex receiving such a message compares the weight of its right edge to \( w \). If it is not equal, it sends \(<CF, w>\) on the appropriate edge to its right which becomes a tree edge. Otherwise it sends:

<CD>: Construct and delete. The worst edge along the path had been discovered. It is not marked as a tree edge. The path has to be continued until meeting some main path, and from this point on the direction to the root is right, therefore the following message is sent along the path:

<CR>: Construct reverse.

We do not get into further details, and only demonstrate the idea in Figure 4.1. We note that, in fact, this process also establishes the directions to the root \( z \).

![Figure 4.1](image)

(===== denotes a tree edge)

5. CONCLUDING REMARKS

In this paper we have outlined a method for distributively solving a variety of reliability, routing and combinatorial optimization problems, some of which are known to be complete in \( \text{NP} \) or \( \#P \) for general graphs, using linear number of messages, when the network is described by a two connected series parallel graph.

In the cases of leader electing and maximum weight independent set, few modifications are required, but the details have not been presented.
The linear number of messages becomes possible due to a new assumption made concerning the distributed model, which we believe is a fair one.

REFERENCES

[SHI] Y. Shibata and M. Morita, "The Representation Tree of a Series Parallel Graph," TR-Mana University, Dept. of Computer Science, Japan.