A Distributed Prime Sieving Algorithm based on *Scheduling by Multiple Edge Reversal*

Gabriel Paillard, Christian Lavault
Laboratoire d'Informatique de Paris Nord
Institut Galilée - Université Paris XIII
Villetaneuse - 93430, France
E-mail: {Gabriel.Paillard,Christian.Lavault}@lipn.univ-paris13.fr
Telephone: +33-1-49-40-40-73, Fax: +33-1-48-26-07-12

Felipe França
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ - Caixa Postal 68511
21941-972, Rio de Janeiro - RJ, Brazil
E-mail: felipe@cos.ufrj.br
Telephone: +55-21-2562-8663, Fax: +55-21-2562-8676

***Abstract*— In this article, we propose a fully distributed algorithm for finding all primes in an given interval $[2..n]$ (or (L, R), more generally), based on the SMER — *Scheduling by Multiple Edge Reversal* — multigraph dynamics. Given a multigraph $\mathcal{M}$ of arbitrary topology, having N nodes, the SMER-driven system is defined by the number of directed edges (arcs) between any two nodes of $\mathcal{M}$, and by the global period length of all "arc reversals" in $\mathcal{M}$. In the domain of prime numbers generation, such a graph method shows quite elegant, and it also yields a totally new kind of distributed prime sieving algorithms of an entirely original design. The maximum number of steps required by the algorithm is at most $n + \sqrt{n}$. Although far beyond the $O(n/\log\log n)$ steps required by the improved sequential "wheel sieve" algorithms, our SMER-based algorithm is fully distributed and of linear (step) complexity. The message complexity of the algorithm is at most $n\Delta_N + \sqrt{n}\Delta_N$, where Δ_N denotes the maximum "multidegree" of the arbitrary multigraph $\mathcal{M}$, and the space required per process is linear.**

Keywords: Distributed prime sieving, resource sharing, multigraph dynamics, scheduling by edge reversal, scheduling by multiple edge reversal.

I. Introduction

Since the sieve of Eratosthenes (more than 2000 years ago), the classical problem of generating all prime numbers from a given interval (L, R) gave rise to several sieving algorithms (from the basic Eratosthenes sieve to fancy ones), implemented in various settings (e.g., sequential and distributed environments).

In its most common form, the sieve of Eratosthenes is a simple device for finding all primes up to some positive integer n. Start with an array of $n - 1$ "ones", corresponding to the numbers from 2 to n. The first one corresponds to "2", so the ones in locations $4, 6, 8$, and so on, are all changed to zeros. The next one is in the position "3", and any one in locations $6, 9, 12$, etc., must be changed into zeros. (Entries that are already zeros are left unchanged.) The sieving process continues in this fashion. If the next entry one corresponds to "p", any entry one at locations $2p$, $3p$, $4p$ is changed to zero, and so on. However, when p is so large that $p^2 > n$, this process may stop. This exit point can be readily detected by noticing that sieving by p leads to no changes of ones to zeros. (At this point, the one entries in the list correspond to the primes not exceeding n, while the zero entries correspond to the composites.)

In passing through the list of the multiples of p, the sieve starts from the initial number p and sequentially adds p until it arrives at a number exceeding n. Thus, the arithmetic operations in the sieve are all additions. The number of steps in the sieve of Eratosthenes is proportional to $\sum_{p\leq n} n/p = n\ln\ln n + O(n)$, where p runs over primes (see [1, Thm. 427]). The number of steps needed per number up to n is proportional to $\ln\ln n$.

The largest computer limitation on sieves is the enormous amount of space they can consume; and it is sometimes necessary to segment the array $[2..n]$. For the segmented ordinary sieve of Eratosthenes, the space required is $O(\sqrt{n})$. However, if the length of a segment drops below $\sqrt{n}$, its efficiency begins to deteriorate. The time it takes to sieve a segment of length ℓ with the primes up to $\sqrt{n}$ is proportional to $\ell\ln\ln n + \pi(\sqrt{n}) + O(\ell)$, where $\pi(x)$ denotes the number of primes up to x. Note that, since $\pi(\sqrt{n}) \sim 2\sqrt{n}/\ln n$ by the prime number theorem, this term may be much larger that the "main term" $\ell\ln\ln n$ when ℓ is small. (See [2] for more details.)

Clearly, the main drawback of the practical sieve of Eratosthenes is clearly the fact that it obliges to go through *all* the entries of the multiples of *each* number during the sieving process. Let the current entry one correspond to "p", then any entry one at locations $2p$, $3p$, $4p$ is changed to zero, and so on, until the exit point when $p^2 > n$. Now, the basic sieve of Eratosthenes proceeds in the same way on *any other* entry one at locations 4, 6, 8, etc., 6, 9, 12, etc. (resp.), and so on until the exit point. In other words, 6 is "generated" twice (from 2 and 3), 12 is "generated" twice (from 2 and 3), etc. Certainly, the entries that are already zeros are left unchanged, but each entry must nevertheless be checked throughout the sieving process.

Thus, the most current and obvious idea consists in trying to prevent all numbers from been sieved "too many times":

Support has been provided by CAPES (Brazilian Research Agency) under contract 1522/00-0.

sieving the multiples of any given number more than once must be avoided, as much as possible. All efficient sieving algorithms are based on similar techniques. The complexity $n \ln \ln n$ of the sieve of Eratosthenes may be somewhat reduced by several clever arguments that are carried out by above methods. Such sieve algorithms improve on the complexity of Eratosthenes and achieve a linear [3], [4] or even a sublinear (step) complexity [4], [5]. So far, the best algorithm known is the "wheel sieve", designed in 1981 [5], [6]; it requires only $O(n/\log\log n)$ steps to find the set of primes in $[2..n]$ (with $n > 4$), where each step is either for bookkeeping or an addition with integers at most n. Basically, the algorithm rests on the central result about the number of primes in arithmetic progressions. More precisely, Dirichlet's theorem states that if a, b are coprime integers ($(a, b) = 1$) and $b > 0$, then the arithmetic progression $\{a, a + b, a + 2b, \ldots\} = \{a \bmod (b)\}$ contains infinite primes (see [1, Thm. 15]). Besides, the total number of additions (in the most operations consuming step) is $O\left(n^{3/4} \ln n\right)$. (See [2] for more details on the analysis of the "wheel sieve".)

Along the same lines, the above results show that a clever prime number generator can be considered as an excellent benchmark to test new architectures (sequential and parallel). Bearing that in mind, the first parallel implementation of the practical sieve of Eratosthenes was realized in a Flex/32 shared-memory multiprocessor in 1987 [7]. From another point of view, a variant of the sieve of Eratosthenes was also distributively implemented in a unidirectional ring of size N. The distributed sieve algorithm designed in [8] is also based on the central result shown in the theorem of Dirichlet stated above. To eliminate multiples of the first n prime numbers $p_1, ..., p_n$, ($p_1 = 2$), the general form of the numbers is

$$\prod_{i=1}^{n} p_i x + y, \qquad 0 < y < \prod_{i=1}^{n} p_i, \ \left(y, \prod_{i=1}^{n} p_i\right) = 1.$$

In order to cancel the multiples of 2, 3, and 5 before the writing and sieving phases, numbers of the following form are generated: $30x + 1,\ 30x + 7,\ 30x + 11,\ 30x + 13,\ 30x + 17,\ 30x + 19,\ 30x + 23,\ 30x + 29$.

Each processor generates the "quasi-primes" numbers of one of the preceding forms, and tests the local primality of the numbers it receives. The prime numbers are dynamically generated and stored in the local memory by each processor. When an integer has performed a complete cycle on the ring without being eliminated by a processor, it is a prime number and it is stored by the generating processor. (Note that the general current form of a message is $\langle 30k + p_j \rangle$, where p_j corresponds to the jth prime (except 1) and $k = 1, 2, 3, \ldots$ The first prime found is $p_4 = 7$, the second one is 11, the third one is 13, etc., and $(30, p_j) = 1$ for all integers in the arithmetic progression $\{p_j \bmod (30)\}$.)

The present paper presents a new kind of fully distributed algorithm that finds all primes by sieving in an given interval $[2..n]$. Our algorithm takes advantage of some powerful graph properties provided by the "scheduling by multiple edge reversal" (SMER) method. This mechanism makes it possible to design a completely new class of sieve algorithms. No explicit arithmetic operations is required in this new class. The mere fundamental operations in the algorithm are only the send-receipt event of a message (arc reversal) and the comparison.

In Section 2 and 3, the framework of the "scheduling by edge reversal" (SER) and the "scheduling by multiple edge reversal" (SMER) mechanisms are both introduced. Section 4 is devoted to the design of our distributed algorithm for sieving primes by using the "SMER-based" method. The complexity analysis of the algorithm is achieved in Section 5. In Section 6, some remarks are introduced and a new conjecture is offered. The final Section 7 draws a short conclusion and offers some perspectives.

II. Scheduling by edge reversal (SER)

Consider a neighborhood-constrained system composed by a set of *processing elements* (PEs) and a set of *atomic shared resources* represented by a connected directed graph $G = (V, E)$, where V is the set of PEs and E the set of its directed edges (or arcs), setting up the access topology. The latter is defined in the following way: an arc exists between any two nodes *if and only if* the two corresponding PEs share at least one atomic resource. SER works as follows: starting from any acyclic orientation ω on G, there is at least one *sink* node, i.e., a node such that all its arcs are directed to itself; all sink nodes are allowed to operate while other nodes remain idle.

This obviously ensures mutual exclusion at any access made to shared resources by sink nodes. After operation, a sink node will reverse the orientation of its arcs, becoming a *source* and thus releasing the access to resources to its neighbors. A new acyclic orientation is defined and the whole process is then repeated for the new set of sinks. Let $\omega' = g(\omega)$ denote this greedy operation. SER can be regarded as the endless repetition of the application of $g(\omega)$ upon G.

Assuming that G is finite, it is easy to see that eventually a set of acyclic orientations will be repeated defining a period of length P. This simple dynamics ensures that no deadlocks or starvation will ever occur since in every acyclic orientation there exists at least one sink, i.e., one node allowed to operate. Also, it is proved that inside any period, every node operates exactly the same constant number of times (M) [9].

SER is a fully distributed graph dynamics in which the sense of time is defined by its own operation, i.e., the synchronous behavior is found in the particular case when every node in G takes the very same exact time to operate and also the same amount of time to reverse arcs. Another interesting observation to be made here is that any topology G will have its own set of possible SER dynamics [10].

As an example of SER's applicability, consider Dijkstra's paradigmatic Dining Philosophers problem [11] under heavy load, i.e., in the case philosophers are either "hungry" or "eating" (no "thinking" state). Such system can be represented by a set $\{P_1, \ldots, P_N\}$ of N PEs, in which each PE shares a resource both with its previous PE and its subsequent

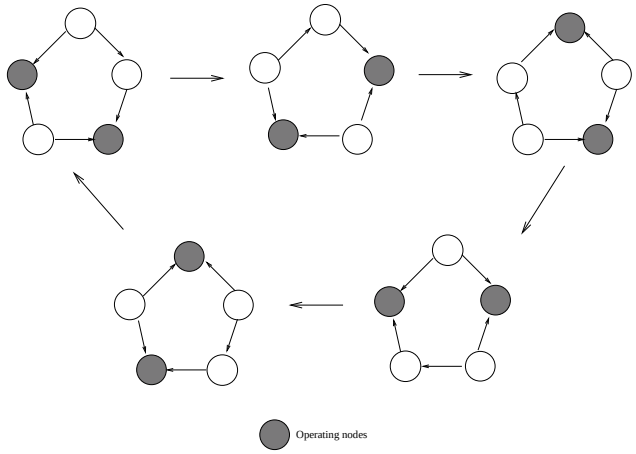

Fig. 1. Dining philosophers

PE. Thus, taking the original configuration where $N = 5$ and setting an acyclic orientation over the 5 nodes ring, the resulting SER dynamics where $P = 5$ and $M = 2$ is illustrated in Figure 1.

III. Scheduling by multiple edge reversal (SMER)

SMER is a generalization of SER in which pre-specified access rates to atomic resources are imposed to processes in a distributed resource-sharing system represented by a multigraph $\mathcal{M} = (V, \mathcal{E})$. Differently from SER, in the SMER dynamics a number of arcs can exist between any two nodes i and j ($i, j \in V$): there can exist $e_{i,j}$ undirected edges, $e_{i,j} \geq 0$, connecting nodes i and j (such nodes are called "neighbors").

Let r_i denote the "reversibility" of node i, i.e., the number of arcs that shall be reversed by i towards each of its neighboring nodes, indiscriminately, at the end of the operation (access to the shared resources). Node i is called an r-sink if at least r_i arcs are directed to itself from each of its neighbors. Each r-sink node i operates by reversing r_i arcs towards its neighbors, next a new set of r-sinks operates in turn, and so on. Similarly to sinks under SER, only r-sink nodes are allowed to operate under SMER. Unlike as in SER, nodes may operate more than once consecutively.

Let $\mu_0, \mu_1, \ldots$ be the sequence of orientations produced by SMER over $\mathcal{M}$ from the initial orientation μ_0. As infinite sequences are of our interest (originally motivated by the Dining Philosophers with rates (DPPr) problem [12]), let a_s^{ij} denote the greatest multiple of $\gcd(r_i, r_j)$, the greatest common divisor of r_i and r_j, which does not exceed the number of edges oriented from i to j in $\mu_s, s \geq 0$. Orientations μ_s such that $f_{ij} = a_s^{ij} + a_s^{ji}$, $s \geq 0$, remains constant as a consequence of the two terms changing by a certain multiple of $\gcd(r_i, r_j)$ (arcs reversed between neighboring nodes i and j), are called *legal*. Let $\mathcal{M}^{i,j}$ be the submultigraph of $\mathcal{M}$ induced by a pair of neighboring nodes i and j. Moreover, let μ_0^{ij}, μ_1^{ij}, ...the sequence of orientations of $\mathcal{M}^{i,j}$ produced by SMER from μ_0^{ij}. The following Lemma 1 states a basic topology constraint towards the definition of the multigraph $\mathcal{M}$.

Lemma 1: ([12], [13])
If $\max\{r_i, r_j\} \leq e_{i,j} \leq r_i + r_j - 1$, then the use of SMER from μ_0^{ij} on $\mathcal{M}^{i,j}$ solves the instance of DPPr given by neighbor nodes i and j, r_i and r_j if and only if $f_{ij} = r_i + r_j - \gcd(r_i, r_j)$. In this case, the sequence $\mu_0^{ij}, \mu_1^{ij}, \ldots$ includes all orientations of $\mathcal{M}^{ij}$ that are legal for i and j given μ_0^{ij}.

Also, it is important to know that there is always at least one SMER solution for any target system's topology having arbitrary pre-specified reversibilities at any of its nodes [13]. Note also that, according to Lemma 1, by having $e_{i,j} = r_i + r_j - 1$, either i or j is in an r-sink condition, independently of $\mu_s, s \geq 0$. It may also be seen that, between all pairs of neighboring nodes i and j in $\mathcal{M}$, any SMER dynamics produces *one unique* period, given by the relation $P_{i,j} = (r_i + r_j)/\gcd(r_i, r_j)$ [12], [14]. This periodic property of SMER can be observed in Figure 2 and Figure 3, where $P_{i,j} = 8$ and $P_{i,j} = 11$ respectively, where nodes in $\mathcal{M}$ share values that are pairwise coprime integers: such pairs (r_i, r_j) (i.e, the numbers of pairwise "reversible" arcs) have no common divisors (but 1). In Figure 5, we can observe a situation where the application of SMER results in a deadlock. This happens whenever, at any cycle in $\mathcal{M}$, the sum of the "reversibilities" of all nodes in that cycle is equal or less than the sum of all oriented arcs in either clockwise or counterclockwise directions [12].

IV. A distributed prime sieving algorithm based on SMER

Recall that, from Lemma 1, the relation $f_{i,j} = r_i + r_j - \gcd(r_i, r_j)$ yields the largest multiple of $\gcd(r_i, r_j)$ below $e_{i,j}$ at each given pair (i, j) of nodes. Using this property, we can state the following lemma.

Lemma 2: For all pair of neighboring nodes i and j, connected by $e_{ij} = r_i + r_j - 1$ arcs, of a given arbitrary multigraph $\mathcal{M}$ under SMER, such that $\gcd(r_i, r_j) > 1$, at least one of such e_{ij} arcs remains static after a period of length $P_{i,j}$ is reached. Otherwise ($\gcd(r_i, r_j) = 1$), no static arcs are observed.

Proof: Consider two nodes i and j of the multigraph $\mathcal{M} = (V, \mathcal{E})$, with their corresponding "reversibilities" r_i and r_j. Two cases may arise. First, if r_i and r_j are coprime, then $f_{i,j} = r_i + r_j - 1 = e_{i,j}$. By contrast, when r_i and r_j have at least a proper common divisor, $\gcd(r_i, r_j) \geq 2$, and then $f_{i,j} \neq e_{i,j}$. Note that, conversely, whenever all arcs are reversed within the SMER period of $\mathcal{M}$, then $f_{i,j} = e_{i,j}$ and thus, there exists a pair of nodes (i, j), in $\mathcal{M}$, such that $(r_i, r_j) = 1$. (Figure 4 illustrates Lemma 2.)[1] ■

Now, from Section 3 and Lemma 2, we can apply the following sieve algorithm distributively to a given interval $[2..n]$, at the end of the algorithm, all the remaining integers from the interval are primes.

Let $\mathcal{M} = (V, \mathcal{E})$ be an arbitrary multigraph having N nodes. For the sake of simplicity, the distributed algorithm is

[1]More details on the validity of the SMER approach for any multigraph can be found in [12].

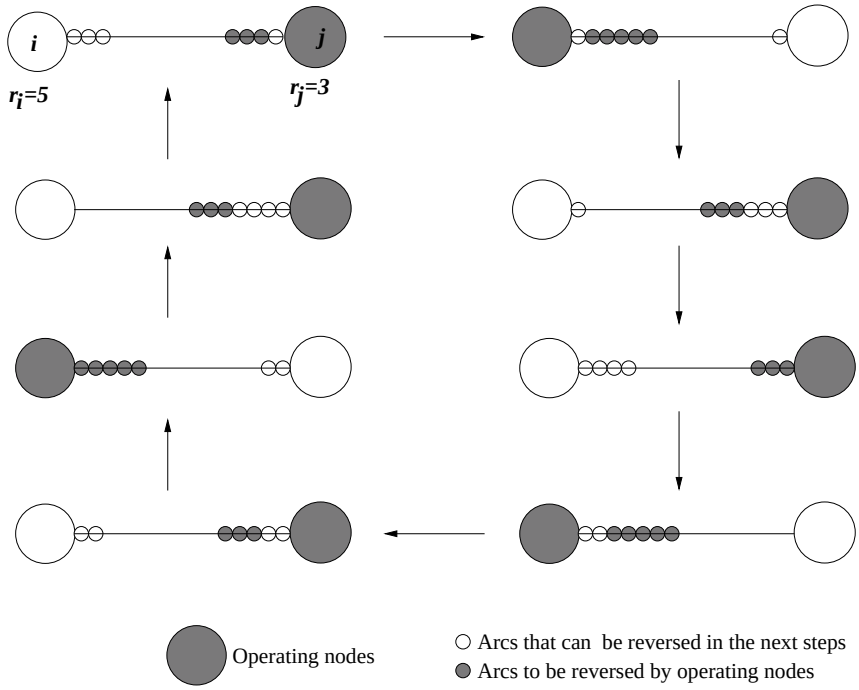

Fig. 2. An example of SMER, with period $P_{i,j} = 8$. Oriented arcs are represented by tokens.

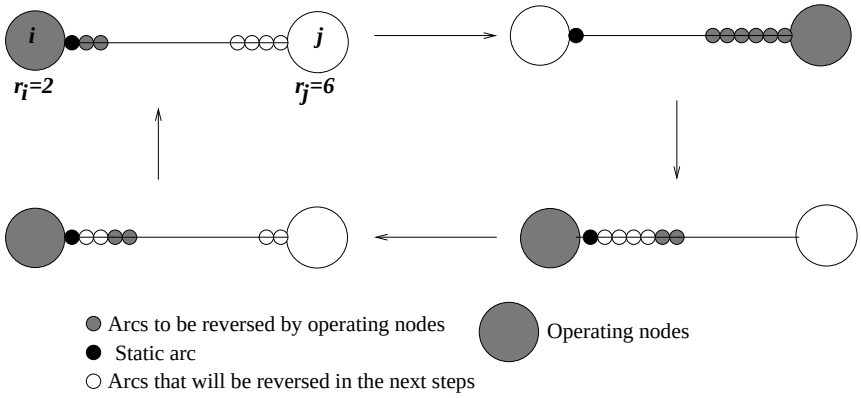

Fig. 4. Pair of nodes (i, j) s.t. r_i and r_j have a common divisor

actually assumed to sieve the restricted interval $I = \{2\} \cup \{$odd integers from $[3..n]\}$ (according to the parity of n). Such a SMER-based sieve algorithm is called "Semi-SMER", indeed, by contrast with the SMER dynamics described in Section 3 (which considers all neighbourhood of any given node). The Semi-SMER assumes that every r_i takes exclusive values from the above restricted interval I.

The procedure Semi-SMER(n) is designed for any current node process $i \in V$, and it uses local variables, defined as follows:

- *Neigh*$_i$ denotes the set of neighbors of process i, and the number of incoming arcs oriented from every $j \in$ *Neigh*$_i$ to the current process i is denoted by the variable *incoming*$_i[j]$.
- $e_i[j]$ denotes the number of undirected edges (both outgoing and incoming arcs) connecting every pair of neighbors (i, j) in $\mathcal{M}$. (See Figure 2.)
- $a_i[j]$ denotes the number of incoming arcs oriented from each $j \in$ *Neigh*$_i$ to i in the initial orientation. (See Figure 4.)
- $r_i[j]$ denotes the required number of arcs that shall be reversed by i towards every $j \in$ *Neigh*$_i$, indiscriminately; i.e., $r_i[j]$ represents the "reversibility" of node i as defined in Section 3 [12]. Each variable $r_i[j]$ and $r_j[i]$ is initially set to an exclusive value in J and in I, respectively, where $J = \{2\} \cup \{$odd integers from

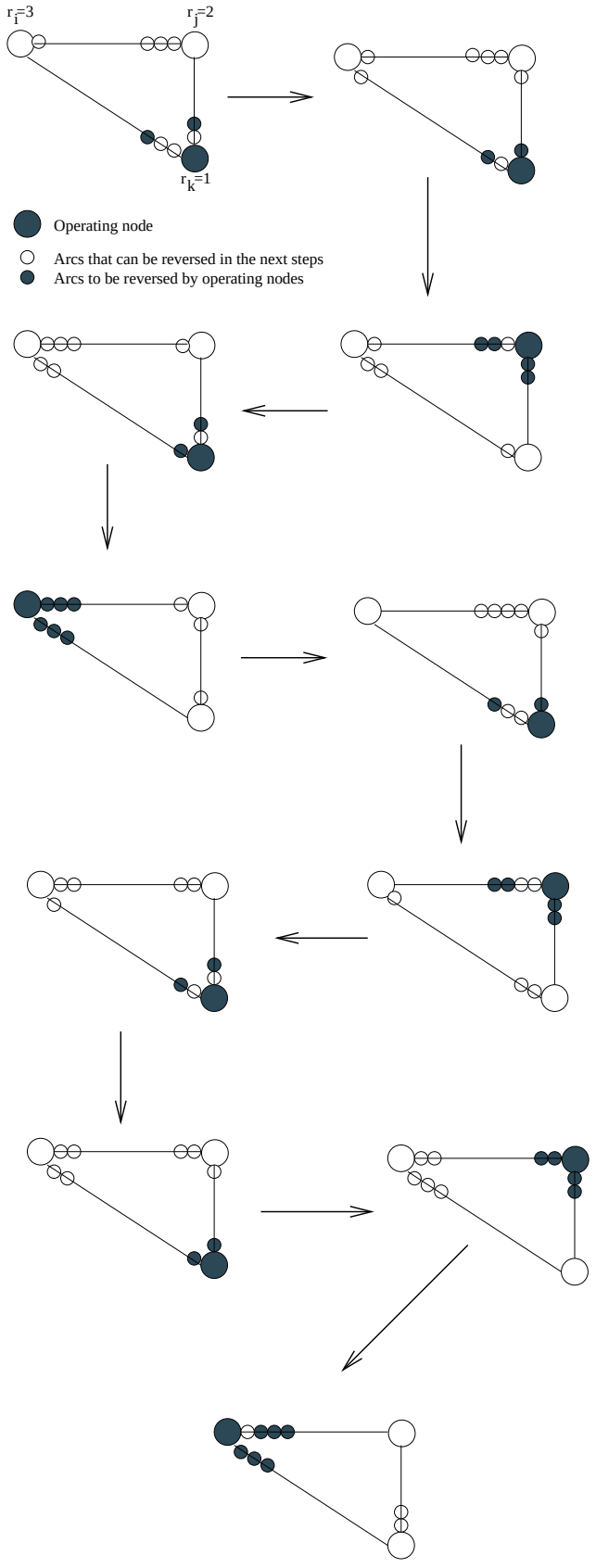

Fig. 3. An example of SMER, with period $P_{i,j} = 11$. Oriented arcs are represented by tokens.

$[3..\sqrt{n}]\}$, and I is defined before. (See Figure 4.)

- Process i also maintains the boolean variables $rev\_arc_i[j]$ and $end\_period\_i[j]$. If, at the end of the Semi-SMER period, $rev\_arc_i[j]$ is true for all $j \in Neigh_i$, then $r_i[j]$ and $r_j[i]$ are coprime.

The first six iterations of the semi-SMER are illustrated in Figure 10, Section Appendix, where the black tokens represent the arcs that remain static during the execution of the algorithm. The sieve interval is $[2..20]$ and $N = 10$. To make the understanding of the figure easier, each node $\leq \sqrt{n}$ corresponds to each other node $> \sqrt{n}$.

Procedure Semi-SMER(n)
var
  $prime$: boolean **init** true;
  $incoming_i[j]$: integer;
  $rev\_arc_i[j]$: boolean **init** false;
  $end\_period_i[j]$: boolean **init** false;
Begin
  If $r_i[j] \leq r_j[i]$ **Then**
    $a_i[j] = r_i[j]$;
    $incoming_i[j] = r_i[j]$;
    $e_i[j] = r_i[j] + r_j[i] - 1$;
  Else
    $a_i[j] = r_i[j] - 1$;
    $incoming_i[j] = a_i[j]$;
    $e_i[j] = r_i[j] + r_j[i] - 1$;
  EndIf
  While not $end\_period_i[j]$
    If $incoming_i[j] \geq r_i[j]$
    Then send message $\langle r_i[j] \rangle$ to $j \in Neigh_i$;
      $incoming_i[j] = incoming_i[j] - r_i[j]$;
      ($\star$ The flipping arcs process is triggered $\star$)
    Else
      receive $\langle r_j[i] \rangle$ from $j \in Neigh_i$;
      $r_i[j] = incoming_i[j] + r_j[i]$;
    EndIf
    If $incoming_i[j] = 0$ **Then** $rev\_arc_i[j] = true$;
    EndIf
    If $incoming_i[j] = a_i[j]$ **Then** $end\_period_i[j] = true$;
    EndIf
  EndWhile
  Forall $j \in Neigh_i$
    If $rev\_arc_i[j] =$ false **Then**
      If $r_i[j] \leq r_j[i]$ **Then** $prime \wedge true$;
      else $prime \wedge false$;
      EndIf
    EndIf
  EndFor
  Return $\mathcal{P}$ ($\star$ $\mathcal{P}$ is the set of primes $\star$)
end.

V. Analysis of the Algorithm

In order to sieve all primes from the interval $[2..n]$, the only fundamental operations explicitly used in the algorithm Semi-

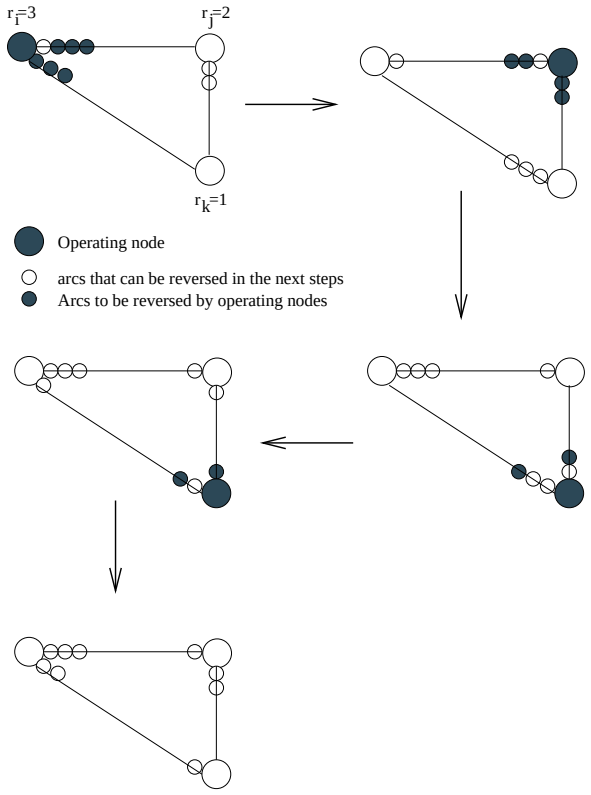

Fig. 5. An example of SMER, with a deadlock.

SMER are comparisons and the send-receipt (arc reversals) of messages $\langle r_i[j] \rangle$ from i to all $j \in \textit{Neigh}_i$ (multicast). Besides, a send-receipt event is assumed to take one time unit, and one comparison operation to take $O(1)$ number of time slots.

Theorem 1: Given an oriented arbitrary multigraph $\mathcal{M} = (V, \mathcal{E})$ having N vertices, the maximum number of steps required by the algorithm Semi-SMER is at most $n + \lfloor \sqrt{n} \rfloor$. The message complexity of the algorithm achieves at most $n\Delta_N + \lfloor \sqrt{n} \rfloor \Delta_N$, where Δ_N denotes the maximum "multi-degree" of $\mathcal{M}$. The maximum space required per process is linear.

Proof: The "multidegree" $\deg_i$ of a vertex $i \in V$, is defined as $\deg_i = \#\textit{Neigh}_i = \sum_j e_{i,j}$, where $e_{i,j} \geq 0$ denotes the number of undirected "multi-edges" connecting node i and all its neighboring vertices.

The number of steps required by the algorithm is proportional to the period involved between any two nodes of $\mathcal{M}$ during the algorithm. Now, the largest period $P_{i,j}$ follows from Lemma 1 and [12]: $P_{i,j} = r_i + r_j$, when $(r_i, r_j) = 1$. Since $r_i \leq \lfloor \sqrt{n} \rfloor$ and $r_j \leq n$, for any pair of nodes (i, j), the procedure Semi-SMER(n) requires at most $n + \lfloor \sqrt{n} \rfloor$ steps.

Similarly, for any node of $\mathcal{M} \leq \sqrt{n}$, the number of exchanged messages in the **while** loop is proportional to their current period times $\deg_i = \#\textit{Neigh}_i$ for the current node process i. Therefore, the maximum message complexity of the algorithm is proportional to the largest period times $\Delta_N = \sup_{i \in V} \deg_i$, where Δ_N denotes the maximum "multidegree" of $\mathcal{M}$ ($1 \leq \Delta_N \leq N - 1$). Hence, the message complexity achieves at most $n\Delta_N + \lfloor \sqrt{n} \rfloor \Delta_N$.

Finally, the maximum amount of memory space required per process is $O(n)$. ■

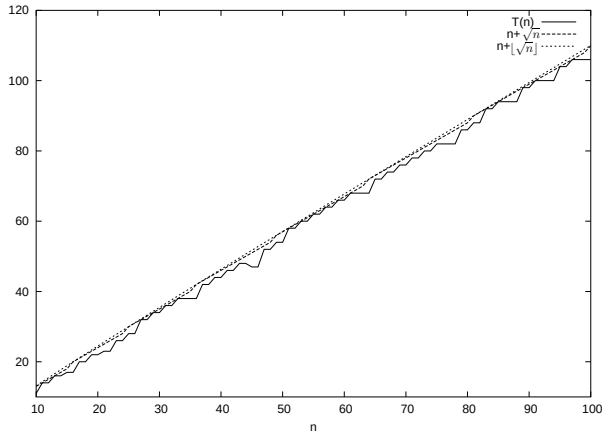

Fig. 6. $n + \lfloor \sqrt{n} \rfloor$, $n + \sqrt{n}$ and $T(n)$, from $n = 10$ up to $n = 100$

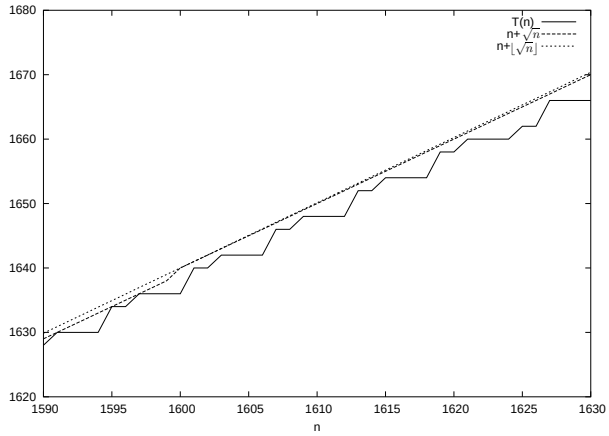

Fig. 7. $n + \lfloor \sqrt{n} \rfloor$, $n + \sqrt{n}$ and $T(n)$, from $n = 1590$ up to $n = 1630$

VI. Remarks and Conjecture

Actually, it stems from various experimental results that the number of steps $T(n)$ executed by the algorithm stays always very close from the maximum number of steps predicted by Theorem 1. More precisely, we have $T(n) = n + \lfloor \sqrt{n} \rfloor - \varphi(n)$, where $\varphi(n)$ is a positive non periodic arithmetic function, with rather small fluctuations for $n \geq 4$. The amplitude of $\varphi(n)$ rests in the narrow strip $0 \leq \varphi(n) < 5$ for "almost every" $n \geq 4$, *but* for very widely spaced out high "peaks" at a few integer values ξ's. The first of these large peaks occurs at $\xi_1 = 514$, with $\varphi(\xi_1) = 6.6715\ldots$, the second one occurs at $\xi_2 = 4210$, with $\varphi(\xi_2) = 6.8845\ldots$, etc.

The Figures 6 and 7 illustrate the variation of $T(n)$ for $n \geq 4$, compared to $n + \sqrt{n}$ and $n + \lfloor \sqrt{n} \rfloor$, respectively, for $n \geq 4$ (both curves bound $T(n)$ from above).

The last two Figures 8 and 9 point out the fluctuations of the error functions $\varphi(n) = n + \lfloor \sqrt{n} \rfloor - T(n)$ and $\varphi'(n) =$

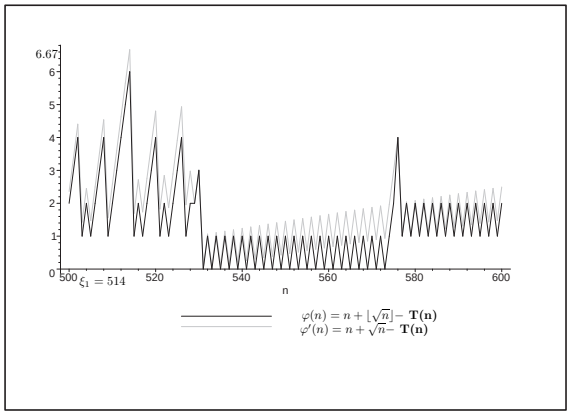

Fig. 8. $\varphi(n)$ and $\varphi'(n)$, from $n = 500$ up to $n = 600$

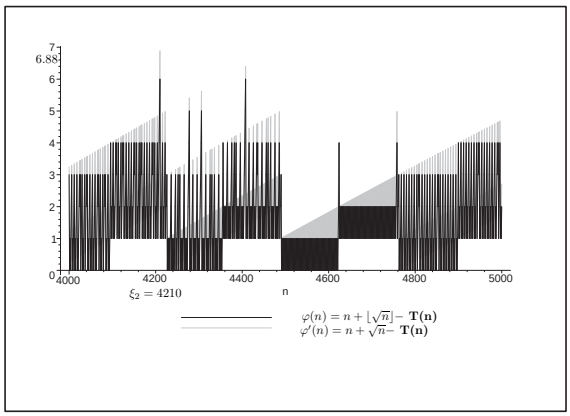

Fig. 9. $\varphi(n)$ and $\varphi'(n)$, from $n = 4000$ up to $n = 5000$

$n + \sqrt{n} - T(n)$, for $n \geq 4$.

The oscillating peaks of $\varphi(n)$ may been seen at various and widely spaced out integers : $\xi_1 = 514$, with $\varphi(\xi_1) = 6.6715\ldots$, $\xi_2 = 4210$, with $\varphi(\xi_2) = 6.8845\ldots$, etc. Though it fluctuates, the function $\varphi(n)$ has mean value $\overline{\varphi(n)} \approx 2.47\ldots$ for $n \geq 4$. More precisely, for all $n \geq 4$ and for all $0 \leq \epsilon_n < 1$, $\overline{\varphi(n)} = 2.47\ldots \pm \epsilon_n$.

Therefore, the average number of steps needed by the Semi-SMER algorithm is proportional to $n + \lfloor\sqrt{n}\rfloor - \overline{\varphi(n)}$, while the average number of messages of the algorithm is $(n + \lfloor\sqrt{n}\rfloor - \overline{\varphi(n)})\,\overline{\deg}_N$, where $\overline{\deg}_N = 1/N \sum_{i \in V} \deg_i$ denotes the "average multidegree" of $\mathcal{M}$ ($1 \leq \overline{\deg}_N \leq \Delta_N$).

Conjecture 1: Denote ξ_i (i positive integer) the integers where the function $\varphi(n)$ takes maximum values (the peaks of $\varphi(n)$), and $(\varphi(\xi_i))$ the sequence of such peaks.

(i) Within any sufficiently large interval of positive integers, the function $\varphi(n)$ gets equal to zero: $\exists n_0 \geq 4\ \forall n \geq n_0 \quad \inf_{n \in [n_0..n]}\{\varphi(n)\} = 0$. In other words, the curve of the function $T(n)$ touches that of $n + \lfloor\sqrt{n}\rfloor$ infinitely often.

(ii) The highest peaks of the sequence $(\varphi(\xi_i))$, are less and less frequent when the ξ_i's get large.

(iii) Though slightly fluctuating (tiny oscillations are observed), we conjecture the property that these peaks remain bounded in the range $6.6715\ldots \leq \varphi(\xi_i) < 7$ and that the sequence $(\varphi(\xi_i))$ is decreasing on the average. Thus $(\varphi(\xi_i))$ converges when $\xi_i \to \infty$.

VII. Conclusion and perspectives

Given an arbitrary multigraph $\mathcal{M}$ having N nodes, the paper introduces a totally new kind of SMER-based distributed sieve algorithm that generates all primes from a given interval $[2..n]$. This approach is the first attempt to use the SMER framework in the domain of basic number theory. Such a graph method approach shows fruitful and quite elegant indeed. Besides, it seems also general enough to compute some of the elementary arithmetic tools and arithmetic functions, as well as to address numbers of problems in fundamental combinatorial arithmetics. For example (*via* the gcd and inverse), the least common multiple of two integers, various inequalities, functional identities, asymptotic expansions or generating functions for e.g., Euler's totient function $\phi(n)$, Möbius function $\mu(n)$, divisor functions $d(n)$, $\sigma(n)$, $\omega(n)$, $\Omega(n)$ and similar functions.

Besides, various combinatorial problems, such as the elementary theory of partitions, factorization of integers, etc., and their applications in cryptography, are very likely to the reach of clever SMER-based algorithms, either parallel, distributed or even sequential.

References

[1] G. Hardy and E. Wright, *An introduction to the theory of numbers*. Oxford: Clarendon Press, 1979.
[2] R. Crandall and C. Pomerance, *Prime Numbers: a computational perspective*. Springer Verlag, 2001.
[3] D. Gries and J. Misra, "A linear sieve algorithm for finding prime numbers," *Communications of the ACM*, vol. 21, no. 12, pp. 999–1003, 1978.
[4] H. Mairson, "Some new upper bounds on the generation of prime numbers," *Communications of the ACM*, vol. 20, no. 9, pp. 664–669, 1977.
[5] P. Pritchard, "A sublinear additive sieve for finding prime numbers," *Communications of the ACM*, vol. 24, no. 1, pp. 18–23, 1981.
[6] ——, "Explaining the wheel sieve," *Acta Informatica*, vol. 17, pp. 477–485, 1982.
[7] S. Bokhari, "Multiprocessing the sieve of Eratosthenes," *IEEE Computer*, vol. 20, no. 4, pp. 50–58, 1987.
[8] M. Cosnard and J.-L. Philippe, "Génération de nombres premiers en parallèle," *La lettre du transputer*, pp. 3–12, 1989.
[9] V. Barbosa and E. Gafni, "Concurrency in heavily loaded neighborhood-constrained systems," *ACM Transactions on Programming Languages and Systems*, vol. 11, no. 4, pp. 562–584, October 1989.
[10] V. Barbosa, *An atlas of edge-reversal dynamics*, ser. Research Notes in Mathematics. Boca Raton, Florida: Chapman & Hall/CRC, 2001.
[11] E. W. Dijkstra, "Hierarchical ordering of sequential processes," *Acta Informatica*, vol. 1, no. 2, pp. 115–138, 1971.
[12] V. Barbosa, M. Benevides, and F. França, "Sharing resources at nonuniform access rates," *Theory of Computing Systems*, vol. 34, no. 1, pp. 13–26, January 2001.
[13] F. França, "Scheduling weightless systems with self-timed boolean networks," in *Workshop on Weightless Neural Networks*, 1993, pp. 87–92.
[14] ——, "Neural networks as neighbourhood-constrained systems," Ph.D. dissertation, Imperial College, London, 1994.

Appendix: Example of the execution of a semi-SMER

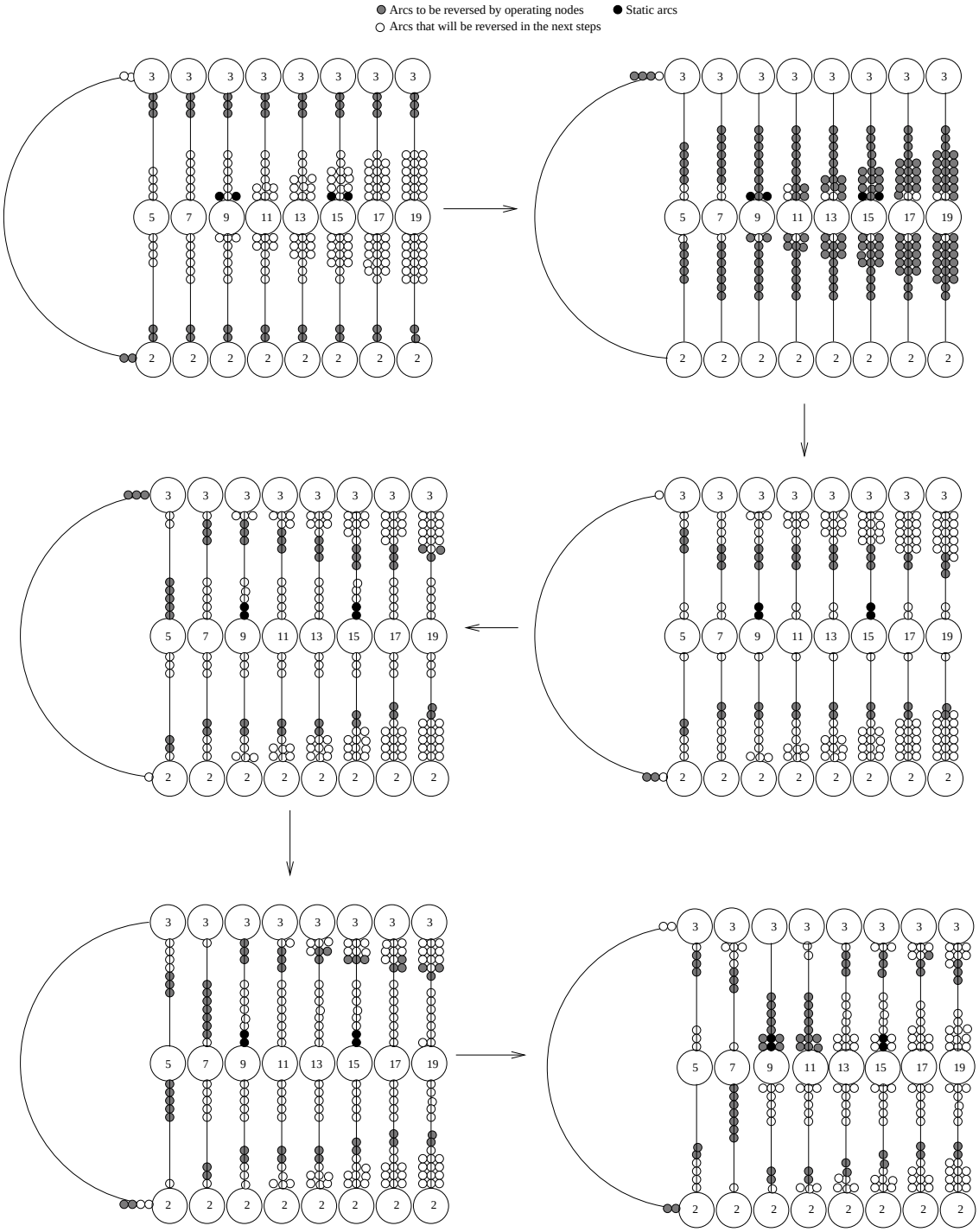

Fig. 10. The first six iterations of Semi-SMER for $n = 20$, nodes having reversibilities 2 and 3 are shown replicated in order to facilitate understanding of the Semi-SMER procedure.