Groupings and Pairings in Anonymous Networks

Jérémie Chalopin[1], Shantanu Das[2], and Nicola Santoro[3]

[1] LaBRI Université Bordeaux 1, Talence, France,
chalopin@labri.fr,
[2] School of Information Technology and Engineering, University of Ottawa, Canada,
shantdas@site.uottawa.ca,
[3] School of Computer Science, Carleton University, Canada,
santoro@scs.carleton.ca

Abstract. We consider a network of processors in the absence of unique identities, and study the k-*Grouping* problem of partitioning the processors into groups of size k and assigning a distinct identity to each group. The case $k = 1$ corresponds to the well known problems of *leader election* and *enumeration* for which the conditions for solvability are already known. The grouping problem for $k \geq 2$ requires to break the symmetry between the processors *partially*, as opposed to problems like leader election or enumeration where the symmetry must be broken completely (i.e. a node has to be distinguishable from all other nodes). We determine what properties are necessary for solving these problems, characterize the classes of networks where it is possible to solve these problems, and provide a solution protocol for solving them.
For the case $k = 2$ we also consider a stronger version of the problem, called *Pairing* where each processor must also determine which other processor is in its group. Our results show that the solvable class of networks in this case varies greatly, depending on the type of prior knowledge about the network that is available to the processors. In each case, we characterize the classes of networks where *Pairing* is solvable and determine the necessary and sufficient conditions for solving the problem.

1 Introduction

Consider a distributed system consisting of a network of n processors and suppose we want to partition the n nodes of the network into uniquely identified groups, each consisting of k nodes, where k divides n. This problem, called k-*Grouping*, is of simple resolution if the nodes have unique identifiers. However, in absence of distinct nodes identities (i.e., in an *anonymous* network), the solution of the k-*Grouping* problem becomes difficult, if at all possible. The goal of this paper is to understand under what conditions this problem is solvable in such a setting.

Notice that when $k = 1$, the grouping problem is equivalent to the well known *Node-Labelling* and *Enumeration* problems, where each node has to be assigned a distinct label (ranging from 1 to n, in case of Enumeration). The 1-*Grouping* problem is also computationally equivalent to the *Leader Election* problem, where one of the nodes has to become distinguished from all others. Although a natural extension to these problems, the k-Grouping problem for $k > 1$ has never been studied before, to the best of our knowledge.

For the leader election problem, it is known that the solvability of the problem depends on the (presence or absence) of symmetry between the nodes in the network. However, even if election is not solvable in a given network, it may be still possible to solve the grouping problem in that same network. In fact, the k-grouping problem for $k \geq 2$ requires to break the symmetry between the nodes only *partially*, as opposed to problems like leader election or enumeration where the symmetry must be broken completely (i.e. a node has to be distinguishable from all other nodes). Hence our investigation focuses on the computational aspects of *partial symmetry-breaking*; more precisely, our interest is in determining what conditions are necessary for solving these problems and in characterizing the solvable instances. A case of particular interest is when $k = 2$, called the *Matching* problem in which the nodes of the network are to be grouped in pairs[4].

It is interesting to note that the solvability of these problems depends not only on the symmetry of the network but also on what information is initially available to the nodes of the network, for instance, whether they know the topology or the size of the network or whether they have a map of the network.

We are also interested in a stronger version of the grouping problem, which we call *k-Relating*, where each node must also determine which other nodes have been grouped with it. Specifically, each node should be able to compute a path between itself and any other processor in its group. In the case $k = 2$, this problem is called *Pairing.* and each node must know a path to the other node it is paired with.

Related Results: The study of computability in an anonymous network of processors, has been a subject of intense research starting from the pioneering work of Angluin [1] who studied the problem of establishing a "center" in the network. This work was extended by Johnson and Schneider [10] and later by Yamashita and Kameda who gave a complete characterization of graphs where the leader election problem is solvable [16] and of graphs where any arbitrary function can be computed [17]. Boldi *et al.* [2] characterized labelled networks based on the election problem, whereas Boldi and Vigna [3] have studied the problem of general computability in directed graphs using the concepts of graph fibrations [4] and coverings, (which we also use in the present paper). Others have studied the computability issues in specific topologies or restricted to special functions (see [11] for a survey of such results). Sakamoto [15] studied the effects of initial conditions of the processors on computability in anonymous networks, while Flocchini *et al.* [8] investigated the impact of *sense of direction* on computability in anonymous networks.

[4] This problem is un-related to the distributed client-server match-making problem studied in the literature [14], where nodes are already divided into clients and servers and the network is not anonymous.

Mazurkiewicz [13] gave an algorithm (in the *local-computation* model) for the distributed enumeration problem, i.e. for numbering the nodes of an undirected graph G with integers from 1 to $|V(G)|$. They showed that it is possible to do this only when the graph G is "unambiguous". Godard et al. [9] translated this property in terms of coverings of simple graphs. Chalopin and Métivier [6] later adapted the Mazurkiewicz algorithm to the message passing model and showed that the enumeration problem is solvable in a symmetric directed graph G, if and only if G is *symmetric-covering-prime*.

Our Results: We first consider the k-*Grouping* problem and provide a complete characterization of its solvability. First of all, we show that the knowledge of the *exact* size of the network is necessary for solving the problem. Then we determine the necessary and sufficient condition for solving the k-*Grouping* problem, when such knowledge is available. For the case $k = 1$, this characterization corresponds precisely (as it should) to that given in [3, 16] for the leader election problem. We then present an algorithm (Algorithm 1) that solves the k-*Grouping* problem using a simple extension to the Mazurkiewicz algorithm. As part of our solution, we introduce a deterministic procedure with explicit termination, that computes the minimum base of any given network in the message-passing system. Our solution is able to detect if k-*Grouping* is solvable for any given k in any given network and reports failure when the problem is not solvable in that network.

Building on the above results, in section 4.1, we investigate the *Pairing* problem under three different types of prior information that may be available to the processors in the network, and we provide an almost complete characterization of its solvability. The types of prior knowledge we consider are: (i) a complete map of the network[5]; (ii) just the number of nodes; (iii) only an upper bound on the number of nodes. We determine sufficient conditions for solving the *Pairing* problem under all three different types of knowledge.

Finally, in section 4.2, we determine necessary conditions for solving the *Pairing* problem in each of the different cases. We show that when a complete map is available or, when only an upper-bound on n is known, the sufficient conditions we have established for these two cases are necessary too; that is, our characterization is complete in case (i) and (iii). In case (ii), when the nodes have prior knowledge of the exact size of the network, there is a still gap between the necessary and sufficient conditions.

2 The Model and the Definitions

2.1 Directed Graphs

A directed graph(digraph) $D = (V(D), A(D), s_D, t_D)$ possibly having muliple arcs and self-loops, is defined by a set $V(D)$ of vertices, a set $A(D)$ of arcs and by two maps s_D and t_D that assign to each arc two elements of $V(D)$: a source and a target (in general, the subscripts will be omitted). A digraph D is strongly connected if for all vertices $u, v \in V(D)$, there exists

[5] The map is unanchored i.e. a node may not know its location in the map.

a path between u and v. A *symmetric* digraph D is a digraph endowed with a symmetry, that is, an involution $Sym : A(D) \to A(D)$ such that for every $a \in A(D), s(a) = t(Sym(a))$. In a symmetric digraph, the mirror of a path $P = (a_0, a_1, \ldots, a_p)$ is the path $(Sym(a_p), Sym(a_{p-1}), \ldots, Sym(a_0))$. In this paper, we will only consider strongly connected symmetric digraphs.

A digraph homomorphism γ between the digraph D and the digraph D' is a mapping $\gamma\colon V(D) \cup A(D) \to V(D') \cup A(D')$ such that if u, v are vertices of D and a is an arc such that $u = s(a)$ and $v = t(a)$ then $\gamma(u) = s(\gamma(a))$ and $\gamma(v) = t(\gamma(a))$. We consider digraphs where the vertices and the arcs are labelled with labels from a recursive label set L and such digraphs will be denoted by (D, λ), where $\lambda\colon V(D) \cup A(D) \to L$ is the labelling function. A homomorphism from (D, λ) to (D', λ') is a digraph homomorphism from D to D' which preserves the labelling, i.e., such that $\lambda'(\gamma(x)) = \lambda(x)$ for every $x \in V(D) \cup A(D)$.

2.2 The Message-Passing Network Model

We represent a point-to-point message passing network by a connected symmetric digraph $\mathbf{G}$ without self-loops and multiple arcs. The vertices represent processors and if there is a (bidirectional) communication link between processors corresponding to some vertices u and v, there is an arc a_{uv} from u to v, an arc a_{vu} from v to u and $Sym(a_{uv}) = a_{vu}$. The initial state of the processors is encoded by a vertex labelling function $\lambda^V : V(\mathbf{G}) \to \Sigma$, where Σ is a set with a total order $<_\Sigma$. In particular, if all vertices have the same label i.e. $\lambda^V(v) = \lambda^V(v'),\ \forall v, v' \in V(G)$, then the network is anonymous.

We assume the presence of a local orientation λ^A on the network: for each vertex u (of degree d), there exists an injective mapping λ^A_u that associates a unique number $\lambda^A_u(v) \in [1, d]$ to each neighbor v of u. This local orientation defines a labelling on the arcs of $\mathbf{G}$ as follows. For any pair of neighboring nodes $\{u, v\}$ in $\mathbf{G}$, $\lambda^A(a_{uv}) = (\lambda^A_u(v), \lambda^A_v(u))$ and $\lambda^A(a_{vu}) = (\lambda^A_v(u), \lambda^A_u(v))$. From this construction, one can notice that for any arc $a \in (\mathbf{G}, \lambda)$, if $\lambda^A(a) = (p, q)$, then $\lambda^A(Sym(a)) = (q, p)$.

The labelled digraph $(\mathbf{G}, \lambda)$ would be called a *network*, if and only if it satisfies each of the following: (i) There does not exist any arc $a \in A(\mathbf{G})$ such that $s(a) = t(a)$ (i.e. no self loops), (ii) There does not exist two distinct arcs $a, a' \in A(\mathbf{G})$ such that $s(a) = s(a')$ and $t(a) = t(a')$ (i.e. no parallel arcs), and (iii) $\lambda = (\lambda^V, \lambda^A)$, where $\lambda^V : V(\mathbf{G}) \to \Sigma$ and λ^A is a local orientation on $\mathbf{G}$, as defined above.

The vertices of the network $(\mathbf{G}, \lambda)$ would be called nodes or, processors. Each processor v in the network represents an entity that is capable of performing computation steps, sending messages on any outgoing arcs, and receiving any message that was sent on any of the incoming arcs. Notice that the entity can distinguish among the arcs due to the presence of local orientation. The following procedure calls are available to the entity at a node v : *Send*$< M, p >$ and *Receive*$< M, p >$, to send (respectively receive) the message M on the communication link labelled by p. Every entity executes the same algorithm provided to it which consists of a sequence of computation steps interspersed with procedure calls of the two types mentioned above. Each of the steps of execution may take an unpredictable (but finite) amount of time (i.e. we consider fully asynchronous systems).

For any path $P = (a_1, a_2, \dots, a_j)$ in the network $(\mathbf{G}, \lambda)$, the sequence of arcs labels corresponding to it is denoted by $\Lambda(P) = (\lambda^A(a_1), \lambda^A(a_2), \dots, \lambda^A(a_j))$. For any sequence of edge-labels α, we define the function T_α for a network $(\mathbf{G}, \lambda)$ as follows. A node $u = T_\alpha(v)$ if and only if there is path P from v to u in G whose label-sequence $\Lambda(P)$ is α. Notice that if λ is a local orientation then there can be at most one node of this kind and then $T_\alpha(v)$ is a mapping.

Each processor, at the beginning of computation would have the same knowledge about the network. As in [16] we will focus on three different kinds of initial knowledge that may be available to the processors:

[UB] Knowledge of an upper bound on n, the size of G,
[ES] Knowledge of the exact value of n, the size of G
[MP] Knowledge of a map (i.e. an isomorphic copy) of the labelled graph $(\mathbf{G}, \lambda)$.

2.3 Problems and Definitions

Informally speaking, the problem of k-*GROUPING* is to partition the nodes of the network into groups of k nodes, where nodes in the same group are identified by a common label assigned to them.

k-GROUPING: Given the network represented by $(\mathbf{G}, \lambda)$, compute at each node v the value $LABEL(v)$ where $LABEL : V(G) \to \mathbb{N}$ satisfies the condition that for each $v \in V(G)$, $|\{u \in V(G) : LABEL(u) = LABEL(v)\}| = k$.

In the particular case, where $k = 1$, this problem corresponds to the well-studied problems of naming/enumeration and election. For the case $k = 2$, we call it the *MATCHING* problem where the nodes of the network are matched-up in pairs such that nodes in a pair share the same label. Notice that the nodes matched to each-other may not be adjacent and in general, a node may not know which other node it has been matched with. A more difficult version of MATCHING (or, 2-GROUPING) is the *PAIRING* problem which involves forming pairs among the nodes of the graph, such that each node v knows a path leading to the other node it is paired with, denoted by $Pair(v)$. This is defined formally as:

PAIRING: Given a network represented by $(\mathbf{G}, \lambda)$, compute at each node v the sequence of edge-labels representing a path from node v to the node $pair(v)$, where the function $pair : V(G) \to V(G)$ is such that (i) $pair(v) = u \Leftrightarrow pair(u) = v$, (ii) $pair(u) = pair(v) \Leftrightarrow u = v$, and (iii) $pair(v) \neq v$ for any $v \in V(G)$.

The generalized version of the *Pairing* problem, called *k-relating*, $k \geq 1$, is not considered in the present paper.

Definition 1. *For each of the above problems, we say that the problem is* solvable *on a given instance* $(\mathbf{G}, \lambda)$*, under the knowledge* MP*(respectively* ES *or,* UB*) if* *there exists a deterministic (distributed) algorithm A such that every execution of algorithm A on $(\mathbf{G}, \lambda)$, succeeds in solving the problem (i.e. produces the required output), when provided with the appropriate input according to* `MP`*(respectively* `ES` *or,* `UB`*).*

We are interested in generic solution protocols for the problems, i.e. algorithms which, when executed on any given instance $(\mathbf{G}, \lambda)$, always terminates within a finite time, either successfully solving the problem, or reporting failure to do so.

Definition 2. *We say that an algorithm A is* effective *for a given problem, under the knowledge* `MP`*(respectively* `ES` *or,* `UB`*) if for every instance $(\mathbf{G}, \lambda)$ of the problem, the algorithm A succeeds in solving the problem if and only if the problem is solvable on $(\mathbf{G}, \lambda)$ under the knowledge* `MP`*(respectively* `ES` *or,* `UB`*)*

2.4 Fibrations and Coverings

The notions of fibrations and coverings were defined by Boldi and Vigna in [4]. We present the main definitions and properties here, that are going to be used in this work.

A *fibration* between the digraphs D and D' is a homomorphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = t(a')$ there exists a unique arc a in $A(D)$ such that $t(a) = v$ and $\varphi(a) = a'$. The fibre over a vertex v (resp. an arc a) of D' is the set $\varphi^{-1}(v)$ of vertices of D (resp. the set $\varphi^{-1}(a)$ of arcs of D).

An *opfibration* between the digraphs D and D' is a homomorphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = s(a')$ there exists a unique arc a in $A(D)$ such that $s(a) = v$ and $\varphi(a) = a'$.

A *covering projection* is a fibration that is also an opfibration. If a covering projection $\varphi : D \to D'$ exists, D is said to be a *covering* of D' via φ and D' is called the base of φ. A symmetric digraph D is a *symmetric covering* of a symmetric digraph D' via a homomorphism φ if D is a covering of D' via φ such that $\forall a \in A(D), \varphi(Sym(a)) = Sym(\varphi(a))$. A digraph D is *symmetric-covering-minimal* if there does not exist any graph D' not isomorphic to D such that D is a symmetric covering of D'.

Property 1 ([4]). Given two non-empty strongly connected digraphs D, D', each covering projection φ from D to D' is surjective; moreover, all the fibres have the same cardinality. This cardinality is called the number of sheets of the covering.

The notions of fibrations and of coverings extend to labelled digraphs in an obvious way: the homomorphisms must preserve the labelling. Given a labelled symmetric digraph (G, λ), the minimum base of (G, λ) is defined to be the labelled digraph (H, λ_H) such that (i) (G, λ) is a symmetric covering of (H, λ_H) and (ii) (H, λ_H) is symmetric covering minimal.

The above definition is equivalent to that given in [12, 4] where the minimum base is defined using the degree refinement technique that is related to techniques used for minimizing deterministic automata.

Given a labelled digraph (G, λ_G) and its minimum base (H, λ_H), the quantity $q = |V(H)|/|V(G)|$ is called the *symmetricity* (see [16]) of the labelled digraph (G, λ_G). This quantity is same as the number of sheets of the covering projection φ from (G, λ_G) to (H, λ_H).

The following property says that if (G, λ_G) is a covering of (H, λ_H), then from any execution of an algorithm on (H, λ_H), one can build an execution of the algorithm on (G, λ_G). This is the counterpart of the lifting lemma that Angluin gives for coverings of simple graphs [1] and the proof can be found in [4, 6].

Property 2. If (G, λ_G) is a covering of (H, λ_H) via φ, then any execution of an algorithm $\mathcal{A}$ on (H, λ_H) can be lifted up to an execution on (G, λ_G), such that at the end of the execution, for any $v \in V(G)$, v would be in the same state as $\varphi(v)$.

In particular, if we consider a synchronous execution of an algorithm $\mathcal{A}$ on (G, λ), then this execution is obtained by lifting up the synchronous execution of $\mathcal{A}$ on the minimum base (H, λ). As a result of the above property we have the following additional property, which is useful for proving impossibility results.

Property 3. Consider two labelled digraphs (G_1, λ_1) and (G_2, λ_2) that both cover the same labelled digraph (H, λ_H) via φ_1 and φ_2 respectively. For any algorithm $\mathcal{A}$, there exist executions of $\mathcal{A}$ on (G_1, λ_1) and (G_2, λ_2) such that at the end of these executions, any vertex $v_1 \in V(G_1)$ would be in the same state as any vertex $v_2 \in \varphi_2^{-1}(\varphi_1(v)) \subset V(G_2)$ provided that the vertices are given the same input initially.

3 Solving the k-Grouping problem

3.1 Conditions for solvability

Throughout the rest of this paper, we shall assume that the values of k and n are such that k divides n, which is a necessary condition for solving the problems that we consider.

Lemma 1. *For solving the* k-Grouping *problem for a given k in a network $(\mathbf{G}, \lambda)$, (i) knowledge of the exact size of the network, is necessary (i.e. the knowledge [*`UB`*] is not sufficient) and (ii) q must divide k, where q is the symmetricity of $(\mathbf{G}, \lambda)$.*

Proof Omitted.

3.2 Solution Protocol

We give below an algorithm *Grouping(n, k)* for solving the k-Grouping problem in a network of size n. The algorithm computes the minimum base (H, λ_H) of the network $(\mathbf{G}, \lambda)$, using the sub-procedure *Enumerate* which is based on Chalopin and Métivier's version of the Mazurkiewicz enumeration algorithm. However, we modify the algorithm to obtain a pseudo-synchronous algorithm that labels the vertices of the network with integers from 1 to $|V(H)|$, such that all nodes that map to the same vertex v in H share the same label. This enables us to compute the minimum base (H, λ_H) based on the labelling. (Note that computing the minimum-base of a digraph is a fundamental problem which is related to state-minimization of automata and also to graph-partitioning and isomorphism detection [5, 7]. However the known solutions are not directly applicable in the present model.)

Algorithm 1: Grouping(n,k)

```
$(H, num) :=$ Enumerate($n$) ;
$q := n/|V(H)|$ ;
$x := n/k$ ;
if $q$ divides $k$ and $k$ divides $n$ then
    return (num modulo x) + 1 ;
else
    Terminate with failure ;
```

Procedure `Enumerate(`$\hat{n}$`)` at node v

```
$n(v) := 1$ ;
$N(v) := \emptyset$ ;
$M(v) := \{(1, \lambda(v), \emptyset\}$ ;
for $\hat{n}^4$ iterations do
    for $p$ := 1 to $d_G(v)$ do send $< (n(v), M(v)), p >$ via port $p$ ;
    for $p$ := 1 to $d_G(v)$ do
        receive $< (x, M), q >$ via port $p$ ;
        $N(v) := N(v) \setminus \{(\_, p, \_)\} \cup \{(x, p, q)\}$ ;
        $M(v) := M(v) \cup M \cup \{(n(v), \lambda(v), N(v))\}$ ;
    if $\exists (n(v), l, N) \in M(v) \mid (\lambda(v), N(v)) \prec (l, N)$ then
        $n(v) := 1 + \max\{x \mid \exists (x, l, N) \in M(v)\}$ ;
        $M(v) := M(v) \cup \{(n(v), \lambda(v), N(v))\}$ ;
$Map :=$ Construct-Graph($M(v)$) ;
return $(Map, n(v))$ ;
```

During the procedure *Enumerate*, the state of each processor $v_i \in V(G)$ is represented by $(\lambda^V(v_i),\ c(v_i))$, where $c(v_i) = (n(v_i), N(v_i), M(v_i))$ represents the following information obtained during the computation:

- $n(v_i) \in \mathbb{N}$ is the *number* assigned to v_i by the algorithm.

– $N(v_i)$ is the *local view* of v_i, i.e., the information the vertex v_i has about its neighbours. This contains elements of the form (n_j, p_j, q_j) where n_j is the number assigned to a neighbor v_j and the arc from v_i to v_j is labelled (p_j, q_j).
– $M(v_i)$ is the *mailbox* of v_i containing all of the information received by v_0 at previous computation steps. Formally, it is a set of elements of the form (n_j, l_j, N_j) where n_j, l_j, and N_j are respectively the number, the initial label and the local view of some node at some previous step of the algorithm.

As in the original algorithm of Mazurkiewicz [13], we need a total order on the local views. Given two local views N_1 and N_2, we shall say that $N_1 \prec N_2$ if the maximum element for the lexicographic order of the symmetric difference $N_1 \triangle N_2 = N_1 \cup N_2 \setminus N_1 \cap N_2$ belongs to N_2. We will also say that $(l_1, N_1) \prec (l_1, N_1)$ if $l_1 <_\Sigma l_2$ or if $l_1 = l_2$ and $N_1 \prec N_2$.

Procedure `Construct-Graph(`M`)`

```
$n_{\max} := \max\{x \mid \exists(x, l, N) \in M\}$ ;
$V(H) := \{v_i \mid 1 \le i \le n_{\max}\}$ ;
$A(H) := \emptyset$ ;
for $i := 1$ to $n_{\max}$ do
    $(\lambda_H(v_i), N_i) := \max_\prec\{(l, N) \mid (i, l, N) \in M\}$ ;
    foreach $(j, p, q) \in N_i$ do
        $A(H) := A(H) \cup \{a_{ijpq}\}$ ;
        $s(a_{ijpq}) = v_i$ ;
        $t(a_{ijpq}) = v_j$ ;
        $\lambda_H(a_{ijpq}) = (p, q)$ ;
return $(H, \lambda_H)$ ;
```

Lemma 2. *During the execution of algorithm* Enumerate($\hat{n}$) *on a network* $(\mathbf{G}, \lambda_G)$ *of size* $\le \hat{n}$*, the map constructed by Procedure* Construct-graph *represents the minimum base* (H, λ_H) *of* $(\mathbf{G}, \lambda_G)$*.*

Once the minimum base of $(\mathbf{G}, \lambda_G)$ has been constructed, it is quite straightforward to solve k-Grouping as shown in Algorithm 1. Notice that the algorithm always succeeds if q divides k which is the necessary condition for solving k-Grouping. Hence we have the following results:

Theorem 1. *Under the knowledge* [`ES`], k-Grouping *is solvable (for any k that divides n) in the network* $(\mathbf{G}, \lambda)$ *if and only if q divides k, where q is the symmetricity of* $(\mathbf{G}, \lambda)$*.*

Corollary 1. *When the size of the network is known,* Matching *is solvable in* $(\mathbf{G}, \lambda)$ *if and only if the symmetricity of* $(\mathbf{G}, \lambda)$ *is either* 1 *or* 2*.*

4 Solving the Pairing problem

4.1 Sufficiency Conditions and Solutions

Lemma 3. *If k-Grouping is solvable in* $(\mathbf{G}, \lambda)$ *for* $k = |G|/2$*, then Pairing is also solvable in* $(\mathbf{G}, \lambda)$*.*

Combining the results of Lemma 1 and Lemma 3, we know that Pairing is solvable in $(\mathbf{G}, \lambda)$ if the symmetricity q divides $n/2$. However, since q always divides n, the above condition is equivalent to the condition that 2 divides n/q (i.e. the size of the minimum base). This gives us the following corollary:

Corollary 2. *Pairing is solvable in* $(\mathbf{G}, \lambda)$ *if it is solvable in* (H, λ_H)*, the minimum base of the network (or equivalently, if* H *has even size).*

In the minimum base (H, λ_H), each vertex is uniquely labelled. Thus, Pairing is solvable in (H, λ_H) if and only if H has even number of vertices. From a solution for Pairing in (H, λ_H), we can easily construct a corresponding solution for $(\mathbf{G}, \lambda_G)$. (We only need to ensure that if a node u is paired to v, using the label sequence α, then v should be paired to u using the inverse sequence of α.) In case H has an odd number of vertices, then some node in $\mathbf{G}$ should be paired with another node having the same label (which is possible if there is a symmetric arc joining them).

Theorem 2. *Under the knowledge* [`UB`]*,* Pairing *is solvable for* $(\mathbf{G}, \lambda_G)$ *having minimum base* (H, λ_H) *and symmetricity* q*, if any one of the following holds:*

(i) (H, λ_H) *has an even number of vertices (i.e.* n/q *is even) or,*
(ii) (H, λ_H) *contains a symmetric self-loop (i.e. an arc* a*, s.t.* $Sym(a) = a$*).*

Let us now consider the case when the exact value of the network size is known.

Theorem 3. *Under the knowledge* [`ES`]*,* Pairing *is solvable for the network* $(\mathbf{G}, \lambda_G)$ *having minimum base* (H, λ_H) *and symmetricity* q*, if one of the following holds:*

(i) (H, λ_H) *has an even number of vertices (i.e.* n/q *is even),*
(ii) there exists a symmetric self-loop in (H, λ_H) *(i.e., a self-loop whose label has the form* (p, p)*),*
(iii) the minimum base has $2|V(H)|$ *arcs , i.e.,* $|A(H)| = 2n/q$*,*
(iv) $q = 4$ *and there exists a self-loop in* (H, λ_H)*,*
(v) $q = 2$ *and there exists two distinct arcs* $a, a' \in A(H)$ *such that* $s(a) = s(a')$ *and* $t(a) = t(a')$*.*

Proof. Suppose that the size of (H, λ_H) is odd and that it does not contain any symmetric self-loop (otherwise, from Lemma 2, we already know that it is possible to solve Pairing). Since (H, λ_H) does not contain any symmetric self-loop, for each arc $a \in A(H)$, there exists $a' \neq a$ such that $Sym(a) = a'$.

Suppose that (H, λ_H) has $2|V(H)|$ arcs. Then there exists exactly two simple cycles C, C' in (H, λ_H) where C' is the mirror of C. The preimage of a cycle in

(H, λ_H) is a set of disjoint cycles in $(\mathbf{G}, \lambda_G)$. If the preimage of C contains at least two cycles, then $(\mathbf{G}, \lambda_G)$ would be disconnected. Consequently, the preimage of C must be a single cycle of length $|C|.q$ in $(\mathbf{G}, \lambda_G)$. Moreover there does not exist any other cycle in $(\mathbf{G}, \lambda_G)$ different from the preimage of C or C'. Since $|V(H)|$ is odd, and $|V(G)| = q|V(H)|$ is even, we know that q is even. Let us fix a vertex $v = s(a_0)$ belonging to the cycle C. Then for each vertex $x \in \varphi^{-1}(v)$, we use the label sequence $\alpha(x) = \Lambda(C)^{q/2}$ to pair it with another vertex $y \in \varphi^{-1}(v)$. Now, the remaining vertices in $\mathbf{G}$ can be easily paired-up.

Suppose there exists a (non-symmetric) self-loop in (H, λ_H) and $q = 4$. Let a be such a self-loop and let $v = s(a) = t(a)$. As explained above, the preimage of a is a set of cycles and the sum of the lengths of these cycles must be 4. Since $(\mathbf{G}, \lambda)$ is a network that contains neither self-loop, nor multiple arcs, the preimage of a cannot contain cycles of length 1 or 2, and then, the preimage of a is a set of cycles of length 4. Consequently, we can associate to each vertex $x \in \varphi^{-1}(v)$ the label $\alpha(x) = \Lambda(aa)$. If $T_{\alpha(x)}(x) = x$, then it means that there exists a cycle of length 2 in $(\mathbf{G}, \lambda_G)$ which is impossible. Let us now consider $y = T_{\alpha(x)}(x)$. Since $\varphi(y) = v$, $\alpha(y) = \alpha(x)$ and consequently $T_{\alpha(y)}(y) = T_{\alpha(x)}(T_{\alpha(x)}(x)) = T_{\alpha(x)^2}(x) = T_{\Lambda(aaaa)}(x) = x$, since the preimage of a consists of cycles of length 4. Consequently all the vertices in $\varphi^{-1}(v)$ will be paired in $(\mathbf{G}, \lambda_G)$. For all the other vertices we proceed as before.

Suppose that $q = 2$ and that there exists two arcs $a, a' \in A(H)$ such that $s(a) = s(a')$ and $t(a) = t(a')$. Let $v = s(a)$ and consider the cycle $(a, Sym(a'))$ of length 2. The preimage of this cycle in $\mathbf{G}, \lambda_G$ is a set of cycles and the sum of the lengths of these cycles must be 4. As before, it implies that the preimage of this cycle consists of a set of cycles of length 4. Then, one can associate to each vertex $x \in \varphi^{-1}(v)$ the label $\alpha(x) = \Lambda(aSym(a'))$. Consider a vertex $x \in \varphi^{-1}(v)$, if $T_{\alpha(x)}(x) = x$, then it means that there exists a cycle of length 2 in $(\mathbf{G}, \lambda_G)$ which is impossible. Let us now consider $y = T_{\alpha(x)}(x)$. Since $\varphi(y) = v$, $\alpha(y) = \alpha(x)$ and consequently $T_{\alpha(y)}(y) = T_{\alpha(x)}(T_{\alpha(x)}(x)) = T_{\alpha(x)^2}(x) = T_{\Lambda(aSym(a')aSym(a'))}(x) = x$ and consequently all the vertices in $\varphi^{-1}(v)$ will be paired in $(\mathbf{G}, \lambda_G)$. For all the other vertices we proceed as before. □

Theorem 4. *Under the knowledge* [MP], Pairing *can be solved for* $(\mathbf{G}, \lambda_G)$ *whose minimum base is* (H, λ_H) *if one of the following holds:*

(i) (H, λ_H) *has an even number of vertices (i.e.* n/q *is even),*

(ii) there exists a vertex $v \in V(H)$ *and a closed path* $P(v, v)$ *in* (H, λ_H) *such that for any vertex* $u \in \varphi^{-1}(v)$, $T_\alpha(u) \neq u$ *and* $T_\alpha(T_\alpha(u)) = T_{\alpha^2}(u) = u$ *where* $\alpha = \Lambda(P)$ *is the sequence of labels corresponding to the path* $P(v, v)$.

The above result clearly indicates how to solve the Pairing problem, when provided with a map of the graph. Observe that the condition *(ii)* in Theorem 2 and the conditions *(ii)*, *(iii)*, *(iv)*, *(v)* in Theorem 3 are particular cases of the condition *(ii)* in Theorem 4.

4.2 Necessary conditions

We now show that most of the sufficient conditions presented in Section 4.1 are in fact, also necessary. First we state a general result about solving Pairing in networks whose minimum base has odd number of vertices.

Lemma 4. *If $(\mathbf{G}, \lambda_G)$ is a network whose minimum base (H, λ_H) has an odd number of vertices, then for any solution to the Pairing problem in $(\mathbf{G}, \lambda_G)$, by some algorithm $\mathcal{A}$, there exists $v \in V(H)$ such that each node $u \in \varphi^{-1}(v)$ is paired with another node $u' \in \varphi^{-1}(v)$.*

Theorem 5. *Under the knowledge* [`MP`], Pairing *cannot be solved for any network $(\mathbf{G}, \lambda_G)$ whose minimum base (H, λ_H) has the following properties:*

(i) *(H, λ_H) has an odd number of vertices, and*
(ii) *there does not exist a vertex $v \in V(H)$ and a closed path $P(v, v)$ in (H, λ_H) such that for every vertex $u \in \varphi^{-1}(v)$, $T_\alpha(u) \neq u$ and $T_\alpha(T_\alpha(u)) = T_{\alpha^2}(u) = u$ where $\alpha = \Lambda(P)$ is the sequence of labels of the arcs in $P(v, v)$.*

Notice that there are indeed networks (of even size) which satisfy the conditions of Theorem 5, and thus, Pairing is unsolvable in such networks even when a map of the network is available. One such example is shown in Figure 1.

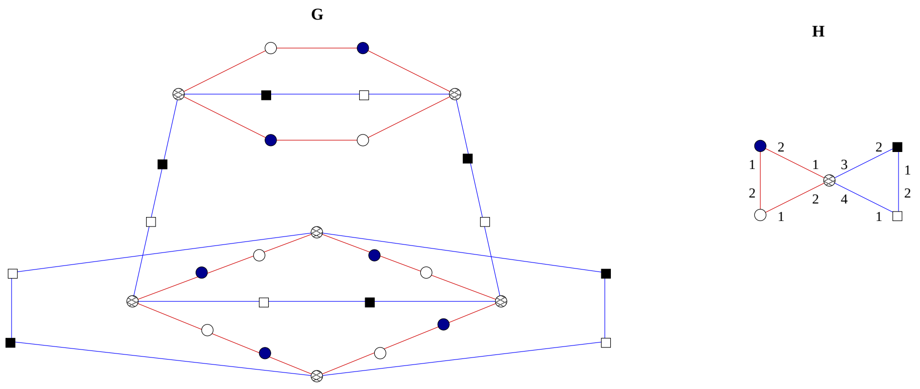

Fig. 1. A simple network, $\mathbf{G}$ and (its minimum base H) where Pairing is not solvable. Here each edge between two nodes represents a pair of arcs, one in each direction (For clarity, the edge labels have been removed from $\mathbf{G}$).

Theorem 6. *Under the knowledge* [`ES`], Pairing *is not solvable for any network $(\mathbf{G}, \lambda_G)$ having minimum base (H, λ_H) and symmetricity q, if all of the following hold:*

(i) *(H, λ_H) has an odd number of vertices,*

(ii) the minimum base has strictly more than $2|V(H)|$ *arcs , i.e.,* $|A(H)| > 2n/q$*,*

(iii) there does not exist any self-loop in (H, λ_H)*, and*

(iv) there does not exist two distinct arcs $a, a' \in A(H)$ *such that* $s(a) = s(a')$ *and* $t(a) = t(a')$*.*

The above result shows that there is a gap between the necessary and sufficient conditions for the case when the exact network size is known. In fact, if the minimum base of the network contains asymmetric self-loop and parallel arcs, then we do not know the exact conditions necessary for solving the Pairing problem. In these cases, it may be possible to characterize the networks in terms of the number of self-loops or parallel arcs in their minimum base, and thus minimize this gap between the necessary and sufficient conditions.

Theorem 7. *Under the knowledge* [UB], Pairing *is not solvable in any network* $(\mathbf{G}, \lambda_G)$ *having minimum base* (H, λ_H)*, if the following holds:*

(i) (H, λ_H) *has an odd number of vertices, and*

(ii) there does not exist any symmetric self-loop in (H, λ_H) *(i.e. an arc a such that Sym(a)=a).*

Proof. Let (H, λ_H) be any symmetric digraph with odd number of vertices and having no symmetric self-loops. If there is an algorithm $\mathcal{A}$ that solves *Pairing* in $(\mathbf{G}, \lambda_G)$ under the knowledge [UB] then this algorithm should work for every network $(\mathbf{G}', \lambda'_G)$ which covers (H, λ_H) (the algorithm cannot differentiate between two networks with a common minimum base, when only an upper bound on the network size is known). We shall now show that the algorithm $\mathcal{A}$ would fail for at least one network $(\mathbf{G}, \lambda_G)$ that covers (H, λ_H). First note that $|A(H)| \geq 2 \cdot |V(H)|$ because otherwise either G is disconnected or G is an odd-sized tree (where Pairing is not possible anyway).

If $|A(H)| \geq 2\ |V(H)|$, then there exists an arc $a \in A(H)$ such that $H' = H \setminus \{a, Sym(a)\}$ is strongly connected. (a could be either a self-loop or one of the pair of parallel arcs or, one of the arcs in a cycle). Let $u = s(a)$, $v = t(a)$, and $a' = Sym(a) \neq a$. Consider the connected digraph H' that is obtained from H by removing the arcs a and a'. Suppose $(\mathbf{G}', \lambda'_G)$ be a network of odd size, whose minimum base is (H', λ_H). Notice that it is always possible to construct such a G', if H' has no symmetric self-loops.

We now construct two networks $(\mathbf{G}_1, \lambda_{G1})$ and $(\mathbf{G}_2, \lambda_{G2})$ defined as follows. To construct $(\mathbf{G}_1, \lambda_{G1})$, we take 4 distinct copies $(\mathbf{G}'_0, \lambda'_0), \ldots, (\mathbf{G}'_3, \lambda'_3)$ of $(\mathbf{G}', \lambda'_G)$. We will denote by $u_{i1}, u_{i2}, \ldots$ (resp. $v_{i1}, v_{i2}, \ldots$) the vertices that corresponds to u (resp. v) in $(\mathbf{G}'_i, \lambda_i)$. We then add the arc a_{ij} with the same label as a (and the symmetric arc a'_{ij} with the same label as a') between u_{ij} and v_{rj}, $r = i+1 \mod 4$ for all i, j. To construct $(\mathbf{G}_2, \lambda_{G2})$, we do the same but we consider 8 distinct copies of $(\mathbf{G}', \lambda'_G)$. Clearly, the two graphs we have constructed are symmetric coverings of (H, λ_H). Thus, due to Lemma 4, there exists a vertex $v \in V(H)$, such that all nodes in $\varphi^{-1}(v)$ are paired among themselves, both in network $(\mathbf{G}_1, \lambda_{G1})$ and network $(\mathbf{G}_2, \lambda_{G2})$.

Due to property 2, there exists an execution of $\mathcal{A}$ on $(\mathbf{G}_1, \lambda_{G1})$ (respectively $(\mathbf{G}_2, \lambda_{G2})$) where the each node in the pre-image of v computes the same label sequence α as computed by v in an execution of $\mathcal{A}$ on (H, λ_H). Consider the path $P(v, v)$ in (H, λ_H), which corresponds to the sequence α. Let $|P|_a$ (resp. $|P|_{a'}$) be the number of times the arc a (resp, a') appears in P and let $n_a = |P|_a - |P|_{a'}$.

CLAIM(1): n_a is of the form $4r_1 + 2$ for some integer r_1.
CLAIM(2): n_a is of the form $8r_2 + 4$ for some integer r_2.

To see why the first claim is true, consider the subgraph G'_i of G_1. There are an odd number of vertices in G'_i, which belong to the preimage of v. Thus, at least one of these nodes must be paired with a node in some other subgraph $G'_j, j \neq i$ of G_1. (Notice that whenever we traverse an arc belonging to the pre-image of a, we move from one subgraph G'_i to the next $G'_{i+1 \mod 4}$.) Thus, in this case, $j = i + n_a \mod 4$ and also $i = j + n_a \mod 4$. So, n_a must be of the form $4r_1 + 2$ for some integer r_1. This proves the first claim. For the second claim, we consider the graph $(\mathbf{G}_2, \lambda_{G2})$ where, using a similar argument we can show that n_a must be of the form $8r_2 + 4$ for some integer r_2.

Note that the two claims above cannot be true simultaneously. This implies that the algorithm $\mathcal{A}$ must fail for one of the networks $(\mathbf{G}_1, \lambda_{G1})$ or $(\mathbf{G}_2, \lambda_{G2})$. □

Due to the earlier results and Theorem 7, we have a complete characterization of those networks where *Pairing* is solvable when provided with an upper bound on the network size.

5 Conclusions and Open problems

In this paper, we studied the problem of k-*Grouping* which is a generalization of the node-enumeration or the election problem, in anonymous networks. In particular we also studied the problem of *Matching* or *Pairing* the nodes of the network. For the *Pairing* problem, the solvability depends on the amount of prior knowledge available. When an upper bound on the network size is known, it is possible to compute the minimum base for the network. We characterized the solvable instances of the *Pairing* problem in terms of the minimum base of the network. When the exact network size is known, the network can be represented by its minimum base and its symmetricity. In this case, the characterization presented in this paper is not complete and there is a gap between the necessary and sufficient conditions, which needs to be investigated. Another possible extension of this work would be to study the generalization of the *Pairing*(or 2-*Relating*) problem to other values of k (say for $k = 3, 4, 5, \dots$) or for arbitrary values of k. It would also be interesting to consider the problem of approximate k-groupings in the case when k does not divide n.

Acknowledgments: The authors would like to thank Paola Flocchini for the many helpful discussions and comments.

References

1. D. Angluin. Local and global properties in networks of processors. In *Proc. 12th ACM Symp. on Theory of Computing* (STOC '80), 82–93, 1980.
2. P. Boldi, B. Codenotti, P. Gemmell, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks: Characterizations. In *Proc. of 4th Israeli Symposium on Theory of Computing and Systems*(ISTCS'96), 16–26, 1996.
3. P. Boldi and S. Vigna. An effective characterization of computability in anonymous networks. In *Proc. 15th Int. Conference on Distributed Computing (DISC'01)*, 33–47, 2001.
4. P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Math.*, 243:21–66, 2002.
5. A. Cardon and M. Crochemore. Partitioning a Graph in $O(|A|log2|V|)$. *Theoretical Computer Science*,19:85–98, 1982.
6. J. Chalopin and Y. Métivier. A bridge between the asynchronous message passing model and local computations in graphs (*extended abstract*). In *Proc. of Mathematical Foundations of Computer Science, MFCS'05*, volume 3618 of *LNCS*, pages 212–223, 2005.
7. D.G. Corneil and C.C. Gotlieb. An Efficient Algorithm for Graph Isomorphism. *Journal of the ACM*, 17(1):51–74, 1970.
8. P. Flocchini, A. Roncato, and N. Santoro. Computing on anonymous networks with Sense of Direction. *Theoretical Computer Science*, 301, 355-379, 2003.
9. E. Godard, Y. Métivier, and A. Muscholl. Characterization of classes of graphs recognizable by local computations. *Theory of Computing Systems*, 37(2):249–293, 2004.
10. R.E. Johnson and F.B. Schneider. Symmetry and similarity in distributed systems. In *Proc. 4th Annual ACM Symp. on Principles of Distributed Computing* (PODC '85), 13–22, 1985.
11. E. Kranakis. Symmetry and computability in anonymous networks: A brief survey. In *Proc. 3rd Int. Conf. on Structural Information and Communication Complexity* (SIROCCO '97), 1–16, 1997.
12. F. T. Leighton. Finite common coverings of graphs. *J. Combin. Theory, Ser. B*, 33:231–238, 1982.
13. A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61(5):233–239, 1997.
14. S.J. Mullender and P.M.B. Vitanyi. Distributed match-making. *Algorithmica*, 3: 367–391, 1998.
15. N. Sakamoto. Comparison of initial conditions for distributed algorithms on anonymous networks. In *Proc. 18th ACM Symposium on Principles of Distributed Computing* (PODC '99), 173–179, 1999.
16. M. Yamashita and T. Kameda. Computing on anonymous networks: Parts I and II. *IEEE Trans. Parallel and Distributed Systems*, 7(1):69–96, 1996.
17. M. Yamashita and T. Kameda. Computing functions on asynchronous anonymous networks. *Mathematical Systems Theory*, 29:331–356, 1996.