Distributed Maximal Matching: Greedy is Optimal

Juho Hirvonen
juho.hirvonen@cs.helsinki.fi

Jukka Suomela
jukka.suomela@cs.helsinki.fi

Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki
P.O. Box 68, FI-00014 University of Helsinki, Finland

ABSTRACT

We study distributed algorithms that find a maximal matching in an anonymous, edge-coloured graph. If the edges are properly coloured with k colours, there is a trivial greedy algorithm that finds a maximal matching in $k-1$ synchronous communication rounds. The present work shows that the greedy algorithm is optimal in the general case: if A is a deterministic distributed algorithm that finds a maximal matching in anonymous, k-edge-coloured graphs, then there is a worst-case input in which the running time of A is at least $k-1$ rounds.

If we focus on graphs of maximum degree Δ, it is known that a maximal matching can be found in $O(\Delta + \log^* k)$ rounds, and prior work implies a lower bound of $\Omega(\text{polylog}(\Delta) + \log^* k)$ rounds. Our work closes the gap between upper and lower bounds: the complexity is $\Theta(\Delta + \log^* k)$ rounds. To our knowledge, this is the first linear-in-Δ lower bound for the distributed complexity of a classical graph problem.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*

Keywords

distributed algorithms, lower bounds, maximal matching

1. INTRODUCTION

In the study of deterministic distributed graph algorithms, there are two parameters that are commonly used to describe the computational complexity of a graph problem: n, the number of nodes in the graph, and Δ, the maximum degree of the graph. For a wide range of problems, the complexity is well-understood as a function of n—at least if $n \gg \Delta$—but understanding the complexity as a function of Δ is one of the major open problems in the area. For example, the maximal matching problem can be solved in $O(\Delta + \log^* n)$ rounds [15], while the best lower bound is $\Omega(\text{polylog}(\Delta) + \log^* n)$ [10–12, 14].

The present works gives the first tight lower bound that is linear in Δ for a classical graph problem. In particular, we study the problem of finding a *maximal matching in anonymous, edge-coloured graphs*. If the edges are k-coloured, the problem can be solved in $O(\Delta + \log^* k)$ rounds with an adaptation of a simple deterministic algorithm [15]. It is well-known that the complexity is $\Omega(\log^* k)$ rounds [14]; we close the case by proving a lower bound of $\Omega(\Delta)$ rounds.

1.1 Related Work

For many graph problems, the state-of-the-art algorithms are extremely fast even if the network is very large—provided that Δ is small. For example, the following problems can be solved in $O(\Delta + \log^* n)$ synchronous communication rounds (assuming $O(\log n)$-bit node identifiers):

- maximal matching [15],
- vertex colouring with $\Delta + 1$ colours [3, 9],
- edge colouring with $2\Delta - 1$ colours [15].

There are also problems that can be solved in $O(\Delta)$ rounds, independently of n (even in anonymous networks without unique identifiers):

- maximal matching in 2-coloured graphs [6],
- maximal edge packing [2],
- 2-approximation of minimum vertex cover [2].

For each of these problems, the dependence on n in the running time is well-understood if $n \gg \Delta$. In particular, Linial's [14] lower bound shows that maximal matching, vertex colouring, and edge colouring require $\Omega(\log^* n)$ rounds, even if $\Delta = 2$.

However, we do not yet understand the dependence on Δ. For example, the best known lower bound for the maximal matching problem is *logarithmic* in Δ [10–12], while the above upper bounds are *linear* in Δ.

Some polylog(Δ) upper bounds are known for graph problems. For example, good approximations of fractional matchings can be found in polylog(Δ) rounds [11]; however, this does not seem to yield a deterministic polylog(Δ)-time algorithm for any of the above problems. Hańćkowiak et al.'s [7] algorithm finds a maximal matching in polylog(n) rounds, avoiding the linear dependence on Δ; however, it comes at the cost of a non-optimal dependence on n.

It is easy to come up with an artificial problem with the complexity of $\Theta(\Delta)$—for example, find nodes u for which

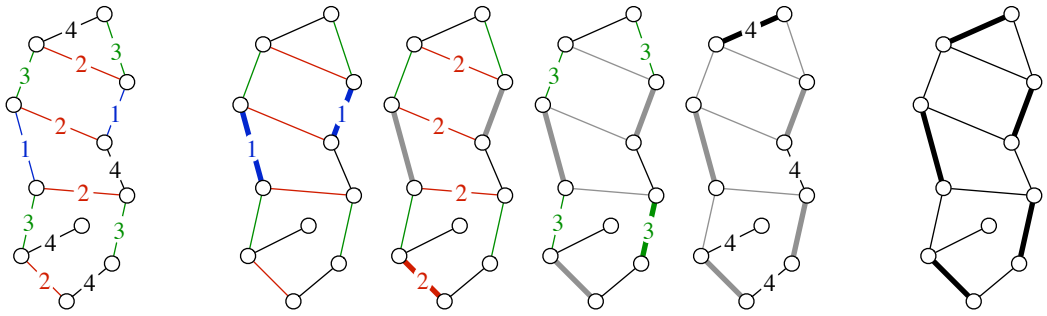


Figure 1: Greedy algorithm for $k = 4$; the thick edges indicate matching M.

there exists a node v such that the distance from u to v is smaller than the degree of v—but so far no such tight results are known for classical graph problems such as maximal matchings. The lower-bound result by Kuhn and Wattenhofer [13] comes close, but it only applies to a restricted family of algorithms.

1.2 Greedy Maximal Matching

We will focus on the task of finding a maximal matching in an edge-coloured graph, using a deterministic distributed algorithm in a network of anonymous nodes (see Section 2 for formally precise definitions and e.g. the survey [16] for more background information).

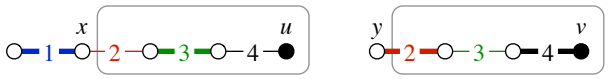
If the graph is edge-coloured with k colours, there is a very simple greedy algorithm that solves the problem in k steps: We start with an empty matching $M \leftarrow \emptyset$. Then, in step i we consider all edges of colour i in parallel. If an edge $\{u, v\}$ is of colour i, and neither u nor v is matched, we add $\{u, v\}$ to M; see Figure 1.

To analyse the exact running time of the greedy algorithm, we need to fix the model of computation. As usual, each node is a computational entity and there is an edge between two nodes if the nodes can exchange messages with each other—the same graph is both the problem instance and the network topology. Throughout this work, the running time is defined to be the number of synchronous communication rounds. Initially, each node knows the colours of its incident edges. In every round, each node in parallel (1) sends a message to each of its neighbours, (2) receives a message from each of its neighbours, and (3) updates its own state. After each round, a node can stop and announce its *local output*: whether it is matched and with which neighbour.

With these definitions, it is straightforward to verify that the running time of the greedy algorithm is at most $k - 1$ communication rounds. To see this, note that the first step of the greedy algorithm does not require any communication: if a node has an incident edge of colour 1, it is matched along this edge. Hence we have the following lemma.

LEMMA 1. *Let k be a positive integer. There exists a deterministic distributed algorithm with running time $k - 1$ that finds a maximal matching in any anonymous, k-edge-coloured graph.*

We can also easily verify that the analysis is tight, i.e., the worst-case running time of the greedy algorithm described above is exactly $k-1$ rounds. The following figure illustrates a worst-case input for $k = 4$; the construction is straightforward to generalise. In the greedy algorithm u is unmatched while v is matched. However, radius-2 neighbourhoods of u and v are indistinguishable; in order to produce a different output, we must propagate information over distance $k-1 = 3$: from x to u and from y to v. Hence any faithful implementation of the greedy algorithm requires at least $k - 1$ communication rounds.



Naturally, if our goal is to find a maximal matching, there is a wide range of possible algorithms, and in many special cases we already know how to beat the greedy algorithm. However, we show that *in the general case, the greedy algorithm is optimal*. The main contribution is summarised in the following theorem.

THEOREM 2. *Let k be a positive integer. A deterministic distributed algorithm that finds a maximal matching in any anonymous, k-edge-coloured graphs requires at least $k - 1$ communication rounds.*

We prove Theorem 2 in Section 3. The lower bound holds even if we allow arbitrarily large messages and unbounded local computations, while the matching upper bound is achieved by a simple algorithm that uses only small messages, little memory, and trivial state transitions.

1.3 Special Cases

Let us now return to the case of bounded-degree graph. If $k \gg \Delta$, we can use Cole–Vishkin [4] style colour reduction techniques to considerably speed up the algorithm. For example, a straightforward adaptation of Panconesi and Rizzi's [15] algorithm finds a maximal matching in $O(\Delta + \log^* k)$ rounds.

Linial's [14] result gives us the lower bound of $\Omega(\log^* k)$; however, so far it has not been known whether $\Omega(\Delta)$ rounds is required. Our result now settles this question. The maximum degree of a k-edge-coloured graph is at most k, and we have the following corollary.

COROLLARY 1. *A deterministic distributed algorithm that finds a maximal matching in an anonymous edge-coloured graph of maximum degree Δ requires $\Omega(\Delta)$ communication rounds.*

Incidentally, our lower-bound construction is a d-regular graph with $d = k - 1$, and hence this work shows that we need d rounds even in the seemingly simple case of d-regular

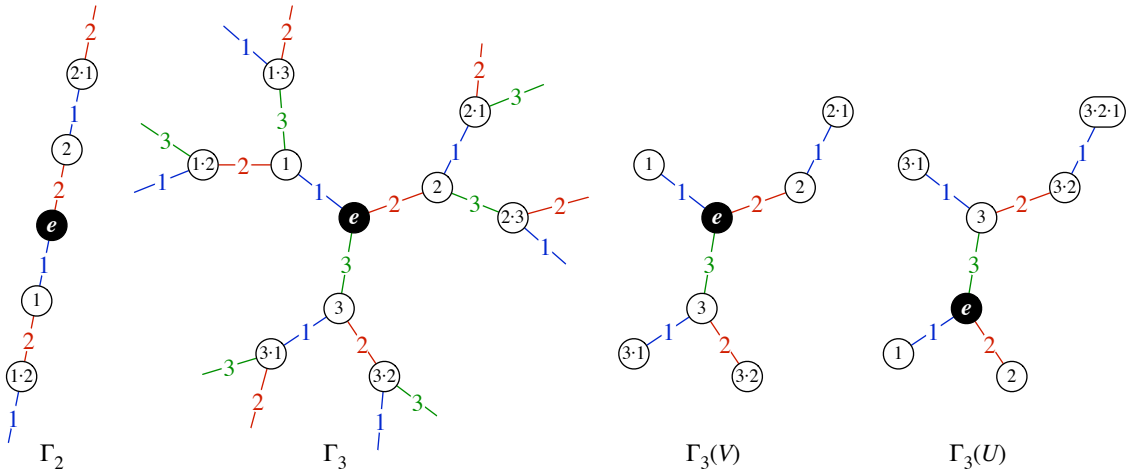


Figure 2: In this example, $V = \{e, 1, 2, 2{\cdot}1, 3, 3{\cdot}1, 3{\cdot}2\} \subseteq G_3$ is a 3-colour system and $U = \bar{3}V$. For example, $V[1] = U[1]$ and $V = V[2] \neq U[2] \neq U$.

graphs (assuming $d \geq 2$). Note that in a regular graph, an optimal fractional matching (edge packing) is trivial to find, and none of the existing lower bounds [10–12] apply—previously, we have not even had polylogarithmic-in-Δ lower bounds for such graphs.

Also note that if we study d-regular graphs with $d = k$, the problem becomes trivial: the edges of colour 1 form a perfect matching and we can solve the problem in constant time. The case of $d = k-1$ is the first non-trivial case, and it is already sufficiently rich to show that the greedy algorithm is optimal.

1.4 Future Work

Our lower-bound result covers the case of anonymous networks, including the widely-studied *port-numbering model* [1, 17] and its weaker variants [18] such as the *broadcast model* [2].

What remains open is the case of networks in which nodes have *unique identifiers*; however, the recent work [5] that bridges the gap between anonymous networks and networks with unique identifiers will likely find applications here as well.

2. PRELIMINARIES

In our lower-bound construction, we will need to manipulate edge-coloured trees, and certain group-theoretic concepts turn out to be useful.

2.1 Group G_k

Throughout this text, k is a positive integer. We use the shorthand notations $X + x = X \cup \{x\}$ and $X - x = X \setminus \{x\}$ for a set X, and $[i] = \{1, 2, \ldots, i\}$ for an integer i.

We define the group

$$G_k = \langle 1, 2, \ldots, k \mid 1^2, 2^2, \ldots, k^2 \rangle.$$

That is, the generators of group G_k are $1, 2, \ldots, k$, and we have the relations $c^2 = cc = e$ for each $c \in [k]$; we use e to denote the identity element, and we use the multiplicative notation xy or $x \cdot y$ for the group operation. Group G_k is the free product of k cyclic groups of order two, a.k.a. the group generated by k involutions, the universal Coxeter group, or the free Coxeter group.

Let Γ_k be the Cayley graph of G_k with respect to the generators $[k]$; see Figure 2 for an illustration. In Γ_k, we have a node for each element $x \in G_k$, and there is an edge of colour $c \in [k]$ from $x \in G_k$ to $y \in G_k$ if $y = xc$. As each generator is its own inverse, there is an edge of colour c from x to y iff there is an edge of colour c from y to x; hence we can interpret Γ_k as an undirected graph. It can be verified that Γ_k is a k-regular k-edge-coloured tree; Γ_k is countably infinite if $k \geq 2$.

In the reduced form, an element $x \in G_k$ is a product $x = c_1 c_2 \cdots c_\ell$ such that $c_i \in [k]$ and $c_{i-1} \neq c_i$. The reduced form is unique; it corresponds to the sequence of edge colours along the unique path from e to x in Γ_k. We use the length of the path to define the norm $|x| = \ell$.

We use the shorthand notation $\bar{x} = x^{-1}$ for the inverse of $x \in G_k$. If $x \in G_k - e$, there is a unique $c \in [k]$ such that $|xc| = |x| - 1$; we say that c is the *tail* of x, in notation $\operatorname{tail}(x) = c$. We also define $\operatorname{head}(x) = \operatorname{tail}(\bar{x})$ and $\operatorname{pred}(x) = x \operatorname{tail}(x)$ for each $x \in G_k - e$.

We make the following observations: If $x, y \in G_k$, then $|\bar{x}y|$ is the length of the unique path from x to y in Γ_k; in particular, $d(x, y) = |\bar{x}y|$ defines a metric on G_k. If $|\bar{x}y| = 1$, nodes x and y are connected with an edge of colour $\bar{x}y$. We have $|\bar{x}| = |x|$ for all $x \in G_k$ and $|xy| \equiv |x| + |y| \mod 2$ for all $x, y \in G_k$. The equality $|xy| = |x| + |y|$ holds iff $x = e$, $y = e$, or $\operatorname{tail}(x) \neq \operatorname{head}(y)$.

If $V \subseteq G_k$ and $x \in G_k$, we define $xV = \{xv : v \in V\}$. If $V \subseteq G_k$, $f \colon V \to X$, and $x \in G_k$, we also define the function $xf \colon xV \to X$ as follows: $(xf)(y) = f(\bar{x}y)$ for each $y \in xV$. That is, $(xf)(xv) = f(v)$ for each $v \in V$.

2.2 Colour Systems

A non-empty set $V \subseteq G_k$ is a *k-colour system* if $v \in V - e$ implies $\operatorname{pred}(v) \in V$. That is, a colour system is a prefix-closed subset; put otherwise, we can start from any $v \in V$ and walk towards e in Γ_k without leaving V. We define the set of edges

$$E(V) = \big\{ \{\operatorname{pred}(v), v\} : v \in V - e \big\}.$$

Let $\Gamma_k(V)$ be the graph with the node set V and the edge set $E(V)$. Now $\Gamma_k(V)$ is a connected subgraph of the tree Γ_k; see Figure 2 for an example. Observe that if $\mathcal{T}$ is any k-edge-coloured tree, then we can construct a k-colour system $V \subseteq G_k$ such that $\mathcal{T}$ and $\Gamma_k(V)$ are isomorphic.

The following lemma is straightforward to verify.

LEMMA 3. *If V is a k-colour system and $u \in V$, then $\bar{u}V$ is a k-colour system. Moreover, $x \mapsto \bar{u}x$ is an isomorphism from $\Gamma_k(V)$ to $\Gamma_k(\bar{u}V)$ that preserves adjacencies and edge colours.*

For a colour system V and integer h, we define $V[h] = \{v \in V : |v| \le h\}$. Similarly, if $f\colon V \to X$, we define that $f[h]\colon V[h] \to X$ is the restriction of f to $V[h]$. Note that $V[h]$ is a colour system. The set $u((\bar{u}V)[h]) \subseteq V$ consists of all nodes that are within distance h from $u \in V$ in $\Gamma_k(V)$.

Let $C(V, v) = \{\bar{u}v : \{u, v\} \in E(V)\}$ denote the set of colours incident to $v \in V$ in $\Gamma_k(V)$. Note that

$$C(V, v) = \{c \in [k] : vc \in V\} = (\bar{v}V)[1] - e.$$

The degree of v is $\deg(V, v) = |C(V, v)|$, and colour system V is said to be d-regular if $\deg(V, v) = d$ for all $v \in V$.

If V is a colour system and $c \in C(V, e)$, we define

$$\mathrm{prune}(V, c) = \{v \in V - e : \mathrm{head}(v) \neq c\} + e.$$

Observe that $U = \mathrm{prune}(V, c)$ is a colour system. Moreover, if V is d-regular, then $\deg(U, u) = d$ for all $u \in U - e$ and $\deg(U, e) = d - 1$.

2.3 Distributed Algorithms

For the purposes of our lower-bound result, it is sufficient to define formally what a distributed algorithm A outputs if we apply it in $\Gamma_k(V)$, where V is a colour system.

We already gave an informal definition of a distributed algorithm in Section 1.2. In particular, we assumed that the nodes are anonymous (they do not have unique identifiers), and initially each node knows the colours of the incident edges. Put otherwise, initially a node $v \in V$ knows precisely $(\bar{v}V)[1]$. Now if we let the nodes exchange all information that they have, after the first round each node $v \in V$ can reconstruct $(\bar{v}V)[2]$, and recursively, after r rounds each node knows precisely $(\bar{v}V)[r+1]$. We will use this as our definition of a distributed algorithm.

Assume that A is a function that associates a *local output* $A(V, v)$ with any colour system V and node $v \in V$. Then we say that A is a *distributed algorithm with running time r* if $(\bar{u}U)[r+1] = (\bar{v}V)[r+1]$ implies $A(U, u) = A(V, v)$.

2.4 Algorithms for Maximal Matchings

We say that a distributed algorithm A *finds a maximal matching* in colour system V if

(M1) we have $A(V, v) \in C(V, v) + \bot$ for each $v \in V$,
(M2) if $A(V, v) = c \neq \bot$ then $vc \in V$ and $A(V, vc) = c$,
(M3) if $A(V, v) = \bot$ and $c \in C(V, v)$ then $A(V, vc) \neq \bot$.

The interpretation is that $A(V, v) = \bot$ if v is unmatched and $A(V, v) = c \in C(v)$ if v is matched along the edge of colour c; see Figure 3 for an illustration. Property (M2) ensures that the output is consistent, and property (M3) ensures that the matching is maximal.

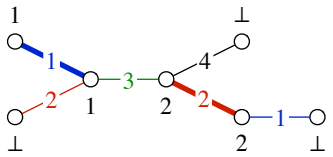


Figure 3: Encoding of a maximal matching.

3. LOWER BOUND

Let us first cover the case of $k \le 2$.

LEMMA 4. *Let $k \in \{1, 2\}$. A deterministic distributed algorithm that finds a maximal matching in any anonymous, k-edge-coloured graphs requires at least $k - 1$ communication rounds.*

PROOF. The case of $k = 1$ is trivial. Let us then focus on the case of $k = 2$. Define the 2-colour systems $T = \{e, 1\}$, $U = \{e, 2\}$, and $V = \{e, 1, 2\}$. Now $A(T, 1) = 1$ and $A(U, 2) = 2$ for any distributed algorithm A. However, we must have either $A(V, 1) \neq 1$ or $A(V, 2) \neq 2$, even though $(\bar{1}T)[1] = (\bar{1}V)[1]$ and $(\bar{2}U)[1] = (\bar{2}V)[1]$. □

The rest of this work contains the proof of the following theorem that covers the case of $k \ge 3$.

THEOREM 5. *Let $k \ge 3$ be an integer, and let $d = k - 1$. Assume that A is a distributed algorithm that finds a maximal matching in any d-regular k-colour system. Then there are two d-regular k-colour systems U and V such that $U[d] = V[d]$, $A(U, e) \neq \bot$, and $A(V, e) = \bot$.*

In particular, the running time of A is at least $d = k - 1$. Theorem 2 follows.

3.1 Overview of the Proof

For the rest of this work, choose k, d, and A as in the statement of Theorem 5, and let r be the running time of A. All colour systems are k-colour systems.

Sections 3.2–3.7 introduce a number of concepts that we will use to present our lower-bound construction. After that, we prove Theorem 5 by induction; the base case is in Section 3.8, and the inductive step in Section 3.9.

3.2 Templates and Colour Pickers

An *h-template* is a pair (T, τ) where $T \subseteq G_k$ is an h-regular colour system and $\tau\colon T \to [k]$ associates a *forbidden colour* $\tau(t) \in [k] \setminus C(T, t)$ with each $t \in T$. The set of *free colours* is

$$F(T, \tau, t) = [k] \setminus (C(T, t) + \tau(t))$$

for each $t \in T$.

Let b be an integer with $0 \le b \le d - h$. A *b-colour picker* for (T, τ) is a function P that associates a subset $P(t) \subseteq F(T, \tau, t)$ of size b with each node $t \in T$. That is, a b-colour picker chooses b free colours for each node. Figure 4 gives an example with $h = 2$, $b = 1$, $d = 4$, and $k = 5$; a 2-template is an infinite path and a 1-colour picker chooses exactly one free colour for each node.

Let P and Q be colour pickers for (T, τ). We say that P and Q are *disjoint* if $P(t) \cap Q(t) = \emptyset$ for all $t \in T$. If P and Q are disjoint colour pickers for (T, τ), we can construct a colour picker R by setting $R(t) = P(t) \cup Q(t)$ for each $t \in T$.

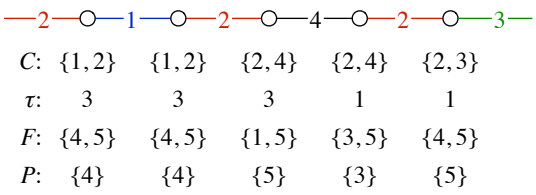


Figure 4: A 2-template and a 1-colour picker.

3.3 Extensions

Let (T, τ) be an h-template and let P be a b-colour picker for (T, τ). We will define a relation $\rightsquigarrow$ between G_k and T recursively as follows; see Figure 5 for an illustration.

(i) We have $e \rightsquigarrow e$, $c \rightsquigarrow c$ for each $c \in C(T, e)$, and $c \rightsquigarrow e$ for each $c \in P(e)$.

(ii) Assume that $x \rightsquigarrow t$ and $x \neq e$.
We have $xc \rightsquigarrow tc$ for each $c \in C(T, t) - \mathrm{tail}(x)$, and $xc \rightsquigarrow t$ for each $c \in P(t) - \mathrm{tail}(x)$.

We make the following observations.

(a) If $x \rightsquigarrow t_1$ and $x \rightsquigarrow t_2$, we have $t_1 = t_2$.

(b) If $x \rightsquigarrow t$ and $x \neq e$, we have $\mathrm{tail}(x) \in C(T, t) \cup P(t)$.

(c) If $x \rightsquigarrow t$, $x \neq e$, and $\mathrm{tail}(x) \in C(T, t)$, we have $\mathrm{pred}(x) \rightsquigarrow t\,\mathrm{tail}(x)$.

(d) If $x \rightsquigarrow t$, $x \neq e$, and $\mathrm{tail}(x) \in P(t)$, we have $\mathrm{pred}(x) \rightsquigarrow t$.

(e) If $x \rightsquigarrow t$ and $c \in C(T, t)$, we have $xc \rightsquigarrow tc$.

(f) If $x \rightsquigarrow t$ and $c \in P(t)$, we have $xc \rightsquigarrow t$.

(g) If $x \rightsquigarrow t$ and $c \in [k] \setminus (C(T, t) \cup P(t))$, there is no $t' \in T$ with $xc \rightsquigarrow t'$.

(h) If $x \rightsquigarrow t$ then $|x| \geq |t|$.

(i) For each $t \in T$ there exists an x such that $x \rightsquigarrow t$.

Let $X = \{x \in G_k : x \rightsquigarrow t \text{ for some } t \in T\}$. Define the function $p\colon X \to T$ as follows: for each $x \in X$, let $p(x)$ be the unique element with $x \rightsquigarrow p(x)$. Let $\xi = \tau \circ p$. We say that (X, ξ, p) is the *P-extension* of (T, τ), in notation, $\mathrm{ext}(T, \tau, P) = (X, \xi, p)$.

Remark 1. We can interpret extensions as universal covering graphs [1] as follows. First, consider the edge-coloured tree $\mathcal{G} = \Gamma_k(T)$. Then modify $\mathcal{G}$ as follows: for each $t \in T$ and $c \in P(t)$, add a self-loop of colour c from t to itself. Now $\mathcal{G}$ is an edge-coloured multigraph; then we construct the universal covering graph $\mathcal{T}$ of $\mathcal{G}$ (i.e., we "unfold" all self-loops of $\mathcal{G}$). Graph $\mathcal{T}$ is an edge-coloured tree; it can be verified that $\mathcal{T}$ is isomorphic to $\Gamma_k(X)$.

3.4 Properties of Extensions

Let us first prove that an extension is a template.

Lemma 6. *Assume that (T, τ) is an h-template, P is a b-colour picker for (T, τ), and $(X, \xi, p) = \mathrm{ext}(T, \tau, P)$. Then X is an $(h+b)$-regular colour system, and (X, ξ) is an $(h+b)$-template. For each $x \in X$ we have $C(X, x) = C(T, p(x)) \cup P(p(x))$.*

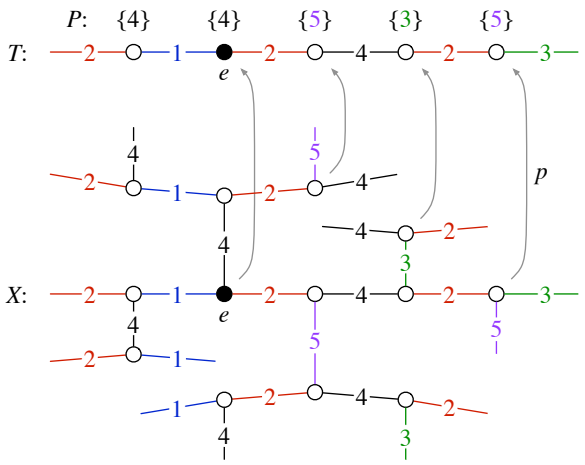


Figure 5: Here T is a 2-template and P is a 1-colour picker. The arrows illustrate the relation $\rightsquigarrow$ between X and T, and hence also function p. In this case, X is a 3-regular colour system.

Proof. Each $x \in X - e$ has $\mathrm{pred}(x) \in X$; hence X is a colour system. If $x \in X$ and $c \in [k]$, we have $xc \in X$ iff $c \in C(T, p(x)) \cup P(p(x))$; hence $C(X, x) = C(T, p(x)) \cup P(p(x))$ and $\deg(x) = h + b$. It follows that X is $(h+b)$-regular. By assumption,

$$\xi(x) = \tau(p(x)) \notin C(T, p(x)) \cup P(p(x)) = C(X, x);$$

that is, ξ associates a valid forbidden colour with each $x \in X$, and we conclude that (X, ξ) is an $(h+b)$-template. □

Next, we observe that an extension has a high degree of symmetry.

Lemma 7. *Let $(X, \xi, p) = \mathrm{ext}(T, \tau, P)$, $x, y \in X$, and $p(x) = p(y)$. Then $\bar{x}X = \bar{y}X$, $\bar{x}\xi = \bar{y}\xi$, and $\bar{x}p = \bar{y}p$.*

Proof. Let $w \in \bar{x}X$. Assume that $w = c_1 c_2 \cdots c_\ell$, where $c_i \in [k]$, and define $w_i = c_1 c_2 \cdots c_i$. We have $w_i \in \bar{x}X$ and $xw_i \in x\bar{x}X = X$ for all i; let $t_i = p(xw_i)$.

With these definitions, $xw_i \rightsquigarrow t_i$ for all $i = 0, 1, \ldots, \ell$. We will prove by induction that $yw_i \rightsquigarrow t_i$ for all i. The base case of $i = 0$ is trivial. Now assume that $xw_i \rightsquigarrow t_i$ and $yw_i \rightsquigarrow t_i$. As we have $xw_ic_{i+1} \rightsquigarrow t_{i+1}$, there are two possibilities. If $c_{i+1} \in C(T, t_i)$, then $t_{i+1} = t_ic_{i+1}$ and $yw_ic_{i+1} \rightsquigarrow t_ic_{i+1}$. Otherwise $c_{i+1} \in P(t_i)$, $t_{i+1} = t_i$ and $yw_ic_{i+1} \rightsquigarrow t_i$. In both cases $yw_{i+1} \rightsquigarrow t_{i+1}$.

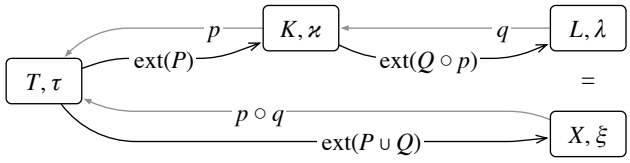
It follows that $yw \rightsquigarrow t_\ell$, and hence $yw \in X$ with $p(yw) = t_\ell = p(xw)$. We have shown that $w \in \bar{x}X$ implies $w = \bar{y}yw \in \bar{y}X$ and

$$(\bar{y}p)(w) = (\bar{y}p)(\bar{y}yw) = p(yw) = p(xw) = (\bar{x}p)(w).$$

By symmetry, $w \in \bar{y}Y$ implies $w \in \bar{x}X$. Finally, $\bar{x}p = \bar{y}p$ implies $\bar{x}\xi = \bar{y}\xi$. □

We also show that the order in which we extend does not affect the end result. If we have two disjoint colour pickers P and Q, we can first apply P and then Q, or vice versa, and we obtain the same result as if we used the colour picker $t \mapsto P(t) \cup Q(t)$ directly; in this sense, extensions commute.

LEMMA 8. *Assume that (T,τ) is a template and P and Q are disjoint colour pickers for (T,τ). Let $R(t) = P(t) \cup Q(t)$ for each $t \in T$. Let $(K,\kappa,p) = \mathrm{ext}(T,\tau,P)$, $(L,\lambda,q) = \mathrm{ext}(K,\kappa,Q \circ p)$, and $(X,\xi,r) = \mathrm{ext}(T,\tau,R)$. Now $X = L$, $\lambda = \xi$, and $p \circ q = r$.*



PROOF. Let $x = c_1 c_2 \cdots c_\ell$, where $c_i \in [k]$, and define $x_i = c_1 c_2 \cdots c_i$. We prove by induction that if $x_i \in X$, we also have $x_i \in L$ with $p(q(x_i)) = r(x_i)$, and if $x_i \notin X$, we also have $x_i \notin L$.

The base case $i = 0$ is trivial: $p(q(e)) = p(e) = e = r(e)$ and $e \in X \cap L$. Now assume that $x_i \in X \cap L$ and $p(q(x_i)) = r(x_i)$. There are four cases depending on c_{i+1}:

(a) Assume that $c_{i+1} \in C(T, r(x_i)) = C(T, p(q(x_i)))$. Then $c_{i+1} \in C(K, q(x_i))$, $x_{i+1} \in X \cap L$, and

$$\begin{aligned} &p(q(x_{i+1})) = p(q(x_i c_{i+1})) \\ &= p(q(x_i)c_{i+1}) = p(q(x_i))c_{i+1} = r(x_i)c_{i+1} \\ &= r(x_i c_{i+1}) = r(x_{i+1}). \end{aligned}$$

(b) Assume that $c_{i+1} \in P(r(x_i)) = P(p(q(x_i))) \subseteq R(r(x_i))$. Then $c_{i+1} \in C(K, q(x_i))$, $x_{i+1} \in X \cap L$, and

$$\begin{aligned} &p(q(x_{i+1})) = p(q(x_i c_{i+1})) \\ &= p(q(x_i)c_{i+1}) = p(q(x_i)) = r(x_i) \\ &= r(x_i c_{i+1}) = r(x_{i+1}). \end{aligned}$$

(c) Assume that $c_{i+1} \in Q(r(x_i)) = Q(p(q(x_i))) \subseteq R(r(x_i))$. Then $c_{i+1} \in (Q \circ p)(q(x_i))$, $x_{i+1} \in X \cap L$, and

$$\begin{aligned} &p(q(x_{i+1})) = p(q(x_i)) = r(x_i) \\ &= r(x_i c_{i+1}) = r(x_{i+1}). \end{aligned}$$

(d) Otherwise $x_{i+1} \notin X$ and $x_{i+1} \notin L$. As a consequence, $x_{i+j} \notin X$ and $x_{i+j} \notin L$ for all $j > 1$.

In conclusion, we have $X = L$, $p \circ q = r$, and $\lambda = \tau \circ p \circ q = \tau \circ r = \xi$. □

3.5 Realisations

Let (T,τ) be an h-template. Define a $(d-h)$-colour picker P by setting $P(t) = F(T,\tau,t)$ for each $t \in T$. Let $(V,g,p) = \mathrm{ext}(T,\tau,P)$. We say that (V,p) is the *realisation* of template (T,τ), in notation, $(V,p) = \mathrm{real}(T,\tau)$.

Intuitively, V is a concrete problem instance—it is always a d-regular colour system, and hence we can apply algorithm A to V. Templates can be seen as compact, schematic representations of problem instances.

Lemma 7 has the following corollary.

COROLLARY 2. *Let $(V,p) = \mathrm{real}(T,\tau)$. If $u, v \in V$ and $p(u) = p(v)$, then $\bar{u}V = \bar{v}V$. In particular, $A(V,u) = A(V,v)$.*

Put otherwise, if (T,τ) is a template with the realisation (V,p), each node $t \in T$ represents an *equivalence class* $p^{-1}(t) \subseteq V$ of nodes with identical outputs. For each $t \in T$, we define $A(T,\tau,t) = A(V,v)$ where $v \in p^{-1}(t)$; by Corollary 2, this does not depend on the choice of v.

We define $M(T,\tau) = \{\{u,v\} \in E(T) : A(T,\tau,u) = A(T,\tau,v) = \bar{u}v\}$. Note that $M(T,\tau)$ is always a matching in the tree $\Gamma_k(T)$, but the matching is not necessarily maximal. If $S \subseteq T$, we also define $M(T,S,\tau) = \{\{u,v\} \in M(T,\tau) : u, v \in S\}$, the restriction of $M(T,\tau)$ to S.

Lemma 8 has the following corollary; it shows that a template and its extensions have the same realisations.

COROLLARY 3. *Let*

$$\begin{aligned} (K,\kappa,p) &= \mathrm{ext}(T,\tau,P), \\ (X,r) &= \mathrm{real}(T,\tau), \\ (L,q) &= \mathrm{real}(K,\kappa). \end{aligned}$$

Then $X = L$, $p \circ q = r$, and $A(K,\kappa,x) = A(T,\tau,p(x))$ for all $x \in K$.

The following lemma is yet another application of the symmetry that we have in extensions: if a template has free colours (i.e., $h < d$), then an algorithm produces a perfect matching in the realisation of the template.

LEMMA 9. *Assume that (T,τ) is an h-template with $h < d$. Then $A(T,\tau,t) \neq \perp$ for all $t \in T$.*

PROOF. Let $(V,p) = \mathrm{real}(T,\tau)$, $t \in T$, and $v \in p^{-1}(t)$. If $h < d$, there exists a $c \in F(T,\tau,t)$. Let $u = vc$; we have $p(u) = p(v) = t$, $c \in C(V,v)$, and

$$A(V,u) = A(V,v) = A(T,\tau,t).$$

Now $A(T,\tau,t) = \perp$ would contradict property (M3). □

3.6 Zero-Templates

Let $Z = \{e\}$ be the colour system with only one node. For each $c \in [k]$, let $\hat{c}$ denote the function $\hat{c}\colon Z \to [k]$ that maps $\hat{c}(e) = c$. Now $(Z,\hat{c})$ is a 0-template for each $c \in [k]$.

If A is the greedy algorithm, we have $A(Z,\hat{1},e) = 2$ and $A(Z,\hat{3},e) \neq 2$. The following lemma generalises this observation.

LEMMA 10. *There are distinct colours $c_1, c_2, c_3 \in [k]$ such that $A(Z,\hat{c}_1,e) = c_2$ and $A(Z,\hat{c}_3,e) \neq c_2$.*

PROOF. For each $c \in [k]$, let $h(c) = A(Z,\hat{c},e)$. By Lemma 9, we have $h(c) \in [k]$ for each $c \in [k]$. Moreover, $h(c) \in [k] - \hat{c}(e) = [k] - c$. Hence we have a function $h\colon [k] \to [k]$ that does not have any fixed points.

First, assume that $h(h(1)) \neq 1$. Then we can choose $c_1 = h(1)$, $c_2 = h(h(1))$, and $c_3 = 1$.

Second, assume that $h(h(1)) = 1$. Let $c \in [k] - \{1, h(1)\}$. If $h(c) = h(1)$, we can choose $c_1 = h(1)$, $c_2 = 1$, and $c_3 = c$. If $h(c) \neq h(1)$, we can choose $c_1 = 1$, $c_2 = h(1)$, and $c_3 = c$. □

3.7 Compatible Templates and Critical Pairs

Let $h \geq 1$. We say that templates (S,σ) and (T,τ) are *h-compatible* if

(C1) $S[h] = T[h]$,
(C2) $\sigma[h-1] = \tau[h-1]$.

We emphasise that h-compatible templates are not necessarily h-templates.

We say that (S,σ) and (T,τ) form an *h-critical pair* if they are h-compatible h-templates and they satisfy the following additional properties:

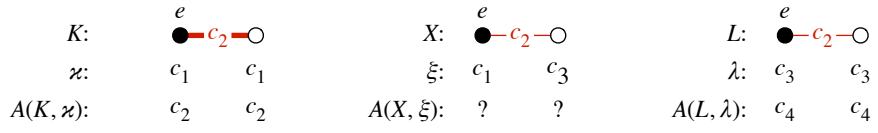


Figure 6: A 1-critical pair.

(C3) $A(T, \tau, e) \notin C(T, e)$,
(C4) $A(S, \sigma, s) \in C(S, s)$ for each $s \in S$.

If $h < d$, Lemma 9 and property (C3) imply that $A(T, \tau, e) \in F(T, \tau, e)$. Property (C4) implies that $M(S, \sigma)$ is a perfect matching in $\Gamma_k(S)$, while property (C3) implies that $M(T, \tau)$ cannot be a perfect matching in $\Gamma_k(T)$.

Remark 2. A reader familiar with Linial's neighbourhood graphs [14] may want to interpret h-compatible templates as adjacent nodes in an h-neighbourhood graph.

3.8 Base Case

In this section we show that there exists a 1-critical pair. Choose $c_1, c_2, c_3 \in [k]$ as in Lemma 10 and let $c_4 = A(Z, \hat{c}_3, e)$. Note that $c_4 \neq c_2$; however, we may have $c_4 = c_1$.

Let $K = L = X = \{e, c_2\}$. Define $\kappa(e) = \kappa(c_2) = \xi(e) = c_1$ and $\lambda(e) = \lambda(c_2) = \xi(c_2) = c_3$. Now (K, κ), (L, λ), and (X, ξ) are 1-templates; the construction is illustrated in Figure 6.

If $p(e) = p(c_2) = e$ and $P(e) = c_2$, we have

$$(K, \kappa, p) = \operatorname{ext}(Z, \hat{c}_1, P),$$
$$(L, \lambda, p) = \operatorname{ext}(Z, \hat{c}_3, P).$$

Therefore $A(K, \kappa, v) = c_2$ for each $v \in K$ and $A(L, \lambda, v) = c_4$ for each $v \in L$.

Now we construct 1-templates (S_1, σ_1) and (T_1, τ_1) as follows:

(i) If $A(X, \xi, e) \neq c_2$, we choose $(S_1, \sigma_1) = (K, \kappa)$ and $(T_1, \tau_1) = (X, \xi)$.

(ii) If $A(X, \xi, e) = c_2$, we choose $(S_1, \sigma_1) = (\bar{c}_2 X, \bar{c}_2 \xi)$ and $(T_1, \tau_1) = (\bar{c}_2 L, \bar{c}_2 \lambda)$.

LEMMA 11. *Templates (S_1, σ_1) and (T_1, τ_1) form a 1-critical pair.*

PROOF. We have $S_1[1] = T_1[1] = K = L = X = \{e, c_2\}$, verifying property (C1). To verify (C2), note that case (i) implies $\sigma_1(e) = \tau_1(e) = c_1$ and case (ii) implies $\sigma_1(e) = \tau_1(e) = c_3$. To verify property (C3), observe that $A(T_1, \tau_1, e) \neq c_2$ while $C(T_1, e) = \{c_2\}$. To verify property (C4), observe that $A(S_1, \sigma_1, s) = c_2$ and $C(S_1, s) = \{c_2\}$ for all $s \in S_1$. □

3.9 Inductive Step

Now assume that (S_h, σ_h) and (T_h, τ_h) form an h-critical pair, where $1 \leq h < d$. In this section, we will construct an $(h+1)$-critical pair (S_{h+1}, σ_{h+1}) and (T_{h+1}, τ_{h+1}).

Recall that Lemma 9 implies that $A(S_h, \sigma_h, s) \neq \perp$ for all $s \in S_h$ and $A(T_h, \tau_h, t) \neq \perp$ for all $t \in T_h$. We define two colour pickers as follows; see Figures 7 and 8 for illustrations.

(i) Define a 1-colour picker Q for (T_h, τ_h) as follows. Let $t \in T_h$. If $A(T_h, \tau_h, t) \in F(T_h, \tau_h, t)$, we choose $Q(t) = \{A(T_h, \tau_h, t)\}$. Otherwise we choose an arbitrary free colour $c \in F(T_h, \tau_h, t)$ and set $Q(t) = \{c\}$.

(ii) Define a 1-colour picker P for (S_h, σ_h) as follows. Let $s \in S_h$. If $|s| \leq h-1$, we have $s \in T_h$ and $F(S_h, \sigma_h, s) = F(T_h, \tau_h, s)$; hence we can choose $P(s) = Q(s)$. Otherwise we choose an arbitrary free colour $c \in F(S_h, \sigma_h, s)$ and set $P(s) = \{c\}$.

Let $(K, \kappa, p) = \operatorname{ext}(S_h, \sigma_h, P)$, $(L, \lambda, q) = \operatorname{ext}(T_h, \tau_h, Q)$, and $\chi = A(T_h, \tau_h, e)$. We make the following observations:

(a) (K, κ) and (L, λ) are $(h+1)$-templates,

(b) (K, κ) and (L, λ) are h-compatible,

(c) $\{e, \chi\} \in E(K)$ and $\{e, \chi\} \in E(L)$,

(d) $p(e) = p(\chi) = e$ and $q(e) = q(\chi) = e$,

(e) $\bar{\chi}K = K$, $\bar{\chi}\kappa = \kappa$, $\bar{\chi}L = L$, and $\bar{\chi}\lambda = \lambda$,

(f) $A(K, \kappa, v) \in C(K, v)$ for each $v \in K$, i.e., $M(K, \kappa)$ is a perfect matching in $\Gamma_k(K)$,

(g) $A(L, \lambda, v) \in C(L, v)$ for each $v \in L$, i.e., $M(L, \lambda)$ is a perfect matching in $\Gamma_k(L)$,

(h) $\{e, \chi\} \notin M(K, \kappa)$ but $\{e, \chi\} \in M(L, \lambda)$.

Now we will use (K, κ) and (L, λ) to construct a new $(h+1)$-template (X, ξ); refer to Figure 7. Let $K_1 = \operatorname{prune}(K, \chi)$, $L_1 = \chi \operatorname{prune}(\bar{\chi}L, \chi)$, and $X = K_1 \cup L_1$. Define $\xi \colon X \to [k]$ as follows: $\xi(v) = \kappa(v)$ for all $v \in K_1$ and $\xi(v) = \lambda(v)$ for all $v \in L_1$. We make the following observations:

(a) (X, ξ) is an $(h+1)$-template,

(b) (X, ξ), (K, κ), and (L, λ) are pairwise h-compatible,

(c) $(\bar{\chi}X, \bar{\chi}\xi)$, $(\bar{\chi}K, \bar{\chi}\kappa)$, and $(\bar{\chi}L, \bar{\chi}\lambda)$ are pairwise h-compatible.

(d) $(\bar{y}X, \bar{y}\xi)$ and $(\bar{y}K, \bar{y}\kappa)$ are $(h+1)$-compatible for any $y \in K_1$,

(e) $(\bar{y}X, \bar{y}\xi)$ and $(\bar{y}L, \bar{y}\lambda)$ are $(h+1)$-compatible for any $y \in L_1$.

Hence we have a family of $(h+1)$-compatible $(h+1)$-templates; however, we need to construct an $(h+1)$-critical pair.

LEMMA 12. *There is a node $y \in X$ such that $A(X, \xi, y) \notin C(X, y)$.*

PROOF. We say that an edge $\{u, v\}$ is *distant* if $|u| > r+1$ and $|v| > r+1$; otherwise it is *near*.

Set $M(K, \kappa)$ is a perfect matching in $\Gamma_k(K)$. Moreover, $\{e, \chi\} \notin M(K, \kappa)$; therefore we have either $\{u, v\} \subseteq K_1$ or $\{u, v\} \cap K_1 = \emptyset$ for each $\{u, v\} \in M(K, \kappa)$. It follows that $\bigcup M(K, K_1, \kappa) = K_1$. Let $K_3' \subseteq M(K, K_1, \kappa)$ consist of the edges that are distant, and let $K_2' = M(K, K_1, \kappa) \setminus K_3'$ consist of the edges that are near. Define $K_2 = \bigcup K_2'$ and $K_3 = \bigcup K_3'$; see Figure 7 for an illustration.

Set $M(L,\lambda)$ is a perfect matching in $\Gamma_k(L)$. Moreover, $\{e,\chi\} \in M(L,\lambda)$; this is the unique edge that joins L_1 and $L \setminus L_1$. Therefore we have $\bigcup M(L,L_1,\lambda) = L_1 - \chi$. Let $L_3' \subseteq M(L,L_1,\lambda)$ consist of the edges that are distant, and let $L_2' = M(L,L_1,\lambda) \setminus L_3'$ consist of the edges that are near. Define $L_2 = (\bigcup L_2') + \chi$ and $L_3 = \bigcup L_3'$.

It follows that

(a) K_3, K_2, L_2, and L_3 form a partition of X,

(b) $(\bar{v}K)[r+1] = (\bar{v}X)[r+1]$ and $(\bar{v}\kappa)[r+1] = (\bar{v}\xi)[r+1]$ for any $v \in K_3$,

(c) $(\bar{v}L)[r+1] = (\bar{v}X)[r+1]$ and $(\bar{v}\lambda)[r+1] = (\bar{v}\xi)[r+1]$ for any $v \in L_3$,

(d) $A(K,\kappa,v) = A(X,\xi,v)$ for any $v \in K_3$,

(e) $A(L,\lambda,v) = A(X,\xi,v)$ for any $v \in L_3$,

(f) $\{u,v\} \in K_3' \cup L_3'$ implies $\{u,v\} \in M(X,\xi)$,

(g) K_2 is a finite set with an even number of nodes,

(h) L_2 is a finite set with an odd number of nodes.

By a parity argument, there is a node $y \in K_2 \cup L_2$ such that $y \notin \bigcup M(X,\xi)$, i.e., $A(X,\xi,y) \notin C(X,y)$. □

Now choose y as in Lemma 12, and define (S_{h+1},σ_{h+1}) and (T_{h+1},τ_{h+1}) as follows:

(a) If $y \in K_1$, define $S_{h+1} = \bar{y}K$, $\sigma_{h+1} = \bar{y}\kappa$, $T_{h+1} = \bar{y}X$, and $\tau_{h+1} = \bar{y}\xi$.

(b) If $y \in L_1$, define $S_{h+1} = \bar{y}L$, $\sigma_{h+1} = \bar{y}\lambda$, $T_{h+1} = \bar{y}X$, and $\tau_{h+1} = \bar{y}\xi$.

LEMMA 13. *Templates (S_{h+1},σ_{h+1}) and (T_{h+1},τ_{h+1}) form an $(h+1)$-critical pair.*

PROOF. First, assume that $y \in K_1$. We have already observed that $(S_{h+1},\sigma_{h+1}) = (\bar{y}K,\bar{y}\kappa)$ and $(T_{h+1},\tau_{h+1}) = (\bar{y}X,\bar{y}\xi)$ are $(h+1)$-compatible. Moreover, we have

$$A(T_{h+1},\tau_{h+1},e) = A(X,\xi,y) \notin C(X,y) = C(T_{h+1},e),$$
$$A(S_{h+1},\sigma_{h+1},s) = A(K,\kappa,ys) \in C(K,ys) = C(S_{h+1},s)$$

for each $s \in S_{h+1}$. Hence (S_{h+1},σ_{h+1}) and (T_{h+1},τ_{h+1}) form an $(h+1)$-critical pair.

The case of $y \in L_1$ is analogous. □

By induction, there are d-templates (S_d,σ_d) and (T_d,τ_d) that form a d-critical pair. Theorem 5 follows by choosing $U = S_d$ and $V = T_d$.

4. ACKNOWLEDGEMENTS

We thank Mika Göös and anonymous reviewers for comments and feedback, and Petteri Kaski, Christoph Lenzen, Joel Rybicki, and Roger Wattenhofer for discussions. This work was supported in part by the Academy of Finland, Grants 132380 and 252018, the Research Funds of the University of Helsinki, and the Finnish Cultural Foundation.

A preprint of this work is available [8]. For presentation slides, see http://www.cs.helsinki.fi/jukka.suomela/mm-lb.

5. REFERENCES

[1] Dana Angluin. Local and global properties in networks of processors. In *Proc. STOC 1980*, pages 82–93. ACM Press, 1980.

[2] Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proc. SPAA 2010*, pages 294–302. ACM Press, 2010.

[3] Leonid Barenboim and Michael Elkin. Distributed $(\Delta+1)$-coloring in linear (in Δ) time. In *Proc. STOC 2009*, pages 111–120. ACM Press, 2009.

[4] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[5] Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. In *Proc. PODC 2012*. ACM Press, 2012.

[6] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. SODA 1998*, pages 219–225. SIAM, 1998.

[7] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. *SIAM J. Discrete Math.*, 15(1):41–57, 2001.

[8] Juho Hirvonen and Jukka Suomela. Distributed maximal matching: greedy is optimal, 2011. Manuscript, arXiv:1110.0367 [cs.DC].

[9] Fabian Kuhn. Weak graph colorings: Distributed algorithms and applications. In *Proc. SPAA 2009*, pages 138–144. ACM Press, 2009.

[10] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proc. PODC 2004*, pages 300–309. ACM Press, 2004.

[11] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. SODA 2006*, pages 980–989. ACM Press, 2006.

[12] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds, 2010. Manuscript, arXiv:1011.5470 [cs.DC].

[13] Fabian Kuhn and Roger Wattenhofer. On the complexity of distributed graph coloring. In *Proc. PODC 2006*, pages 7–15. ACM Press, 2006.

[14] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[15] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distrib. Comput.*, 14(2):97–100, 2001.

[16] Jukka Suomela. Survey of local algorithms. *ACM Comput. Surveys*. To appear.

[17] Masafumi Yamashita and Tsunehiko Kameda. Computing on anonymous networks: Part I—characterizing the solvable cases. *IEEE Trans. Parallel Distrib. Systems*, 7(1):69–89, 1996.

[18] Masafumi Yamashita and Tsunehiko Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Trans. Parallel Distrib. Systems*, 10(9):878–887, 1999.

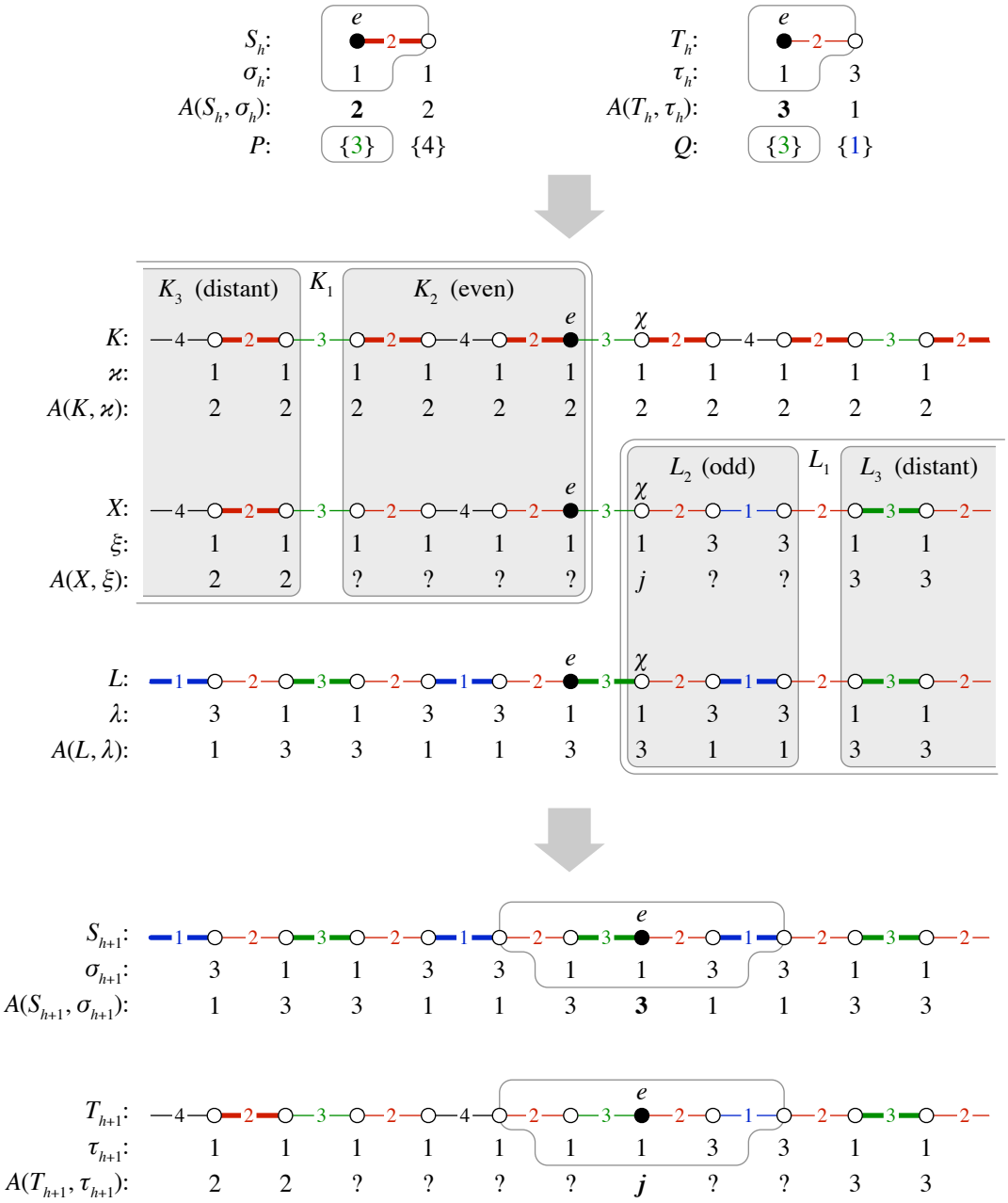


Figure 7: Inductive step. In this example, $h = 1$ and $\chi = 3$. We assume that $j \notin \{2, 3\}$, and thus we can choose $y = \chi$ in Lemma 12.

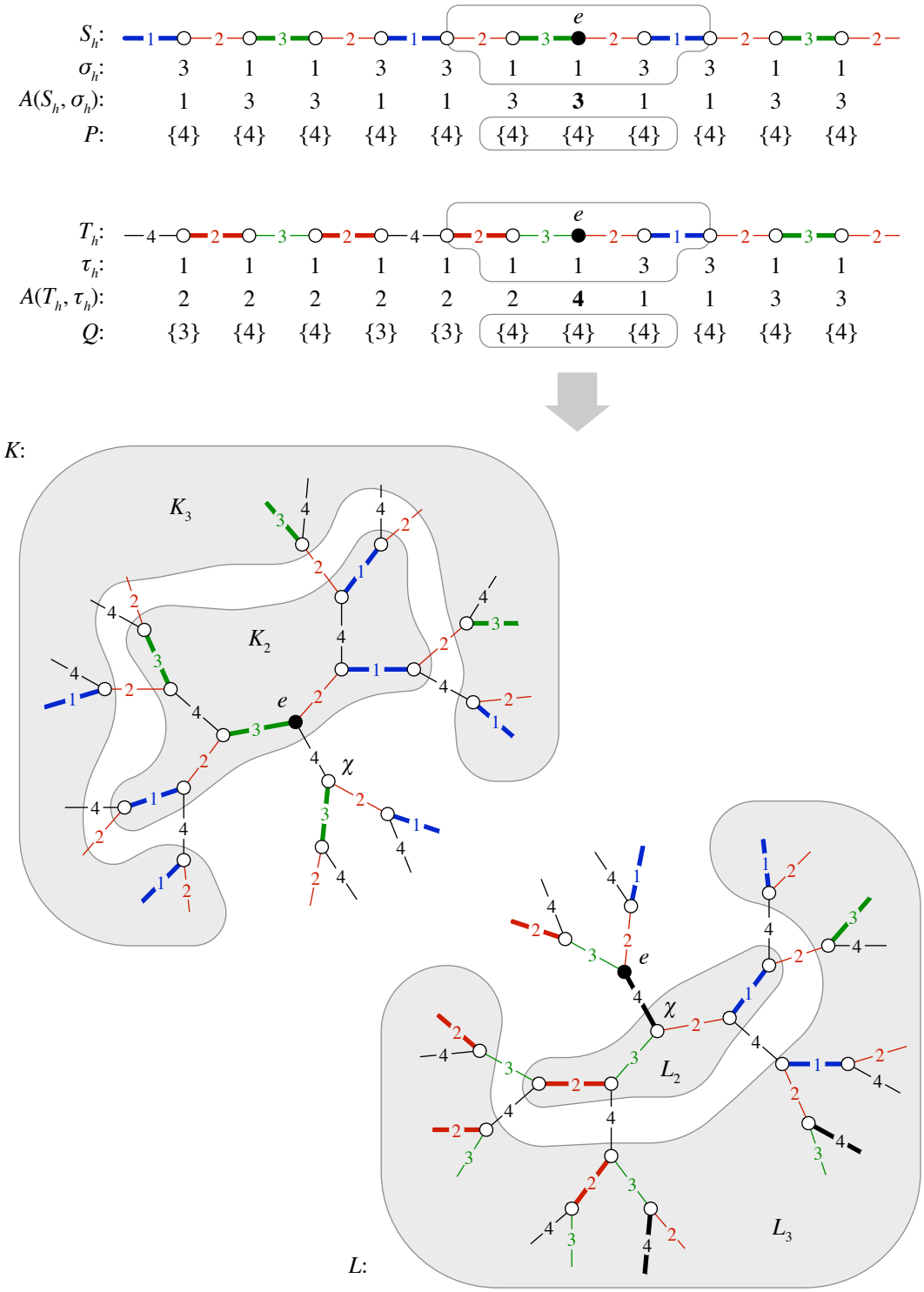


Figure 8: Inductive step. In this example, $h = 2$ and $\chi = 4$.