Interval Routing & Layered Cross Product: Compact Routing Schemes for Butterflies, Mesh of Trees and Fat Trees

Tiziana Calamoneri[1] and Miriam Di Ianni[2]

[1] Dip. di Scienze dell'Informazione, Università di Roma "la Sapienza"
via Salaria 113, I-00198 Roma, Italy.
calamo@dsi.uniroma1.it

[2] Istituto di Elettronica, Università di Perugia
via G.Duranti 1/A, I-06123 Perugia, Italy.
diianni@istel.ing.unipg.it

Abstract. In this paper we propose compact routing schemes having space and time complexities comparable to a 2-Interval Routing Scheme for the class of networks decomposable as Layered Cross Product (LCP) of rooted trees. As a consequence, we are able to design a 2-Interval Routing Scheme for butterflies, meshes of trees and fat trees using a fast local routing algorithm. Finally, we show that a compact routing scheme for networks which are LCP of general graphs cannot be found by any only using shortest paths information on the factors.

1 Introduction

The information needed to route messages in parallel and distributed systems must be somehow stored in each node of the network. The simplest solution consists of a complete routing table – stored in each node u – that specifies for each destination v at least one link incident to u and lying on a path from u to v. The required space of such a solution is $\Theta(n \log \delta)$, where δ is the node degree and n the number of nodes in the network. Efficiency considerations lead to store *shortest paths* information. For better and fair use of network resources, storing, for each entry of the routing table, as many outgoing links as necessary to describe *all* shortest paths in the network should be aimed. Due to limited storage space at each processor, a linear increase of the routing table size in n is not acceptable. Research has then focused on identifying classes of network topologies whose shortest paths information can be succinctly stored, assuming that some "short" labels can be assigned to nodes and links at preprocessing time.

In the *Interval Routing Scheme* (in short, IRS) [7, 12], node-labels belong to the set $\{1, \dots, n\}$, while link-labels are pairs of node-labels representing cyclic intervals of $[1, \dots, n]$. A message with destination v arriving at a node u is sent by u onto an incident link whose label $[v_1, v_2]$ is such that $v \in [v_1, v_2]$. Such an approach allows one to achieve an efficient memory occupation. An IRS is said *optimum* if the route traversed by each message is a shortest path from its source to its destination. It is said *overall optimum* if a message can be routed along *any* shortest path. In [6, 12] optimum IRSs have been designed for particular network topologies. In [3, 11, 13] it has been proved the existence of networks that do not admit any optimum IRS. *Multi-label Interval Routing Schemes* were introduced [7] to extend the model in order to allow more than one interval to be associated to each link: a k-IRS is a scheme associating at most k intervals to each link. A message whose destination is node v is sent onto a link labeled $(I_1, \ldots, I_k)$ if $v \in I_i$ for some $1 \leq i \leq k$. In [2] a technique for proving lower bounds on the minimum k allowed was developed and in [4] it has been used to construct n-node networks for which any optimal k-IRS requires $k = \Theta(n)$. It was proved that for some well known interconnection networks, such as shuffle exchange, cube connected cycle, butterfly and star graph, each optimal k-IRS requires $k = \Omega(n^{1/2-\epsilon})$ – for proper values of ϵ – to store one shortest path for each pair [5]. Of course, this lower bound still holds to store *any* shortest path.

In this paper, after providing the necessary preliminary definitions (Section 2), we propose overall optimum compact routing schemes (Section 3) based on the same leading idea as the Multi-label Interval Routing for all networks which are Layered Cross Product (LCP) [1] of rooted trees (in short, T-networks). For many commonly used interconnection networks falling in this definition no overall optimum compact routing scheme was known. Among them we recall three widely studied topologies: butterflies, mesh of trees and fat trees. Our compact routing scheme requires as much space and time as those required by a 2-IRS. The achievement is particularly meaningful for butterflies because of the result in [5]. Finally, in Section 4, we give a negative result by proving that the knowledge of shortest paths on the factors of a network could be not enough to compute shortest paths on it.

2 Definitions and Preliminary Results

Point to point communication networks are usually represented by graphs, whose nodes stand for processors and edges for communication links. We always represent each edge $\{u, v\}$ by the pair of (oriented) arcs, (u, v) and (v, u).

An *l-layered graph*, $G = (V^1, V^2, \ldots, V^l, E)$ consists of l layers of nodes; V^i is the (non-empty) set of nodes in layer i, $1 \leq i \leq l$; every edge in E connects vertices of two adjacent layers. In particular a rooted tree T of height h is a h-layered graph, layer i defined either as the set of nodes having distance $i-1$ from the root or as the set of nodes having distance $h-i$ from the root. From now on, we call T a *root-tree* or a *leaf-tree* according to whether the first or the second way of defining layers is chosen. In Fig. 1, T_1 is a root-tree, while T_2 is a leaf-tree.

Let $G_1 = (V_1^1, V_1^2, \ldots, V_1^l, E_1)$ and $G_2 = (V_2^1, V_2^2, \ldots, V_2^l, E_2)$ be two l-layered graphs. Their *Layered Cross Product* (LCP for short) [1] $G^1 \times G^2$ is an l-layered graph $G = (V^1, V^2, \ldots, V^l, E)$ where V^i is the cartesian product of V_1^i and V_2^i, $1 \leq i \leq l$, and a link $((a,\alpha),(b,\beta))$ belongs to E if and only if $(a,b) \in E_1$ and $(\alpha,\beta) \in E_2$.

Many common networks are LCP of trees [1]. Among theml: the *butterfly* with N inputs and N outputs is the LCP of two N-leaves complete binary trees (Fig. 1.a), the *mesh of trees* of size $2N$ is the LCP of two N-leaves complete binary trees with paths of length $\log N$ attached to their leaves (Fig. 1.b), the *fat tree* of height h [10] is the LCP of a complete binary tree and a complete quaternary tree, both of height h (Fig. 1.c).

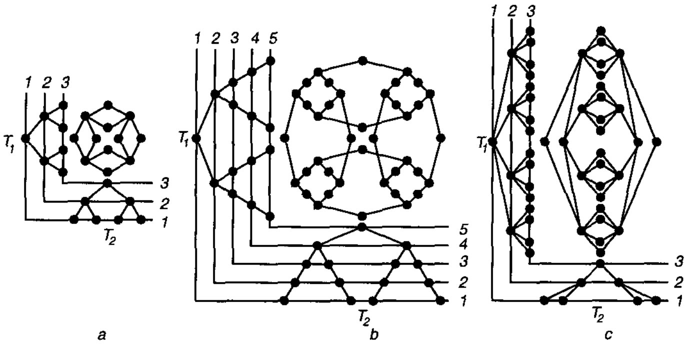

Fig. 1. Butterfly, Mesh of Trees and Fat-tree as LCP of rooted trees.

Fact 21 *Let $(a,\alpha),(b,\beta) \in V(G_1 \times G_2)$. Any shortest path from (a,α) to (b,β) is never shorter than a shortest path from a to b in G^1 and a shortest path from α to β in G^2.*

Fact 22 *If G is the LCP of either two root-trees or two leaf-trees then G is a tree.*

Observe that the LCP of two trees could be also not connected. This is not a restriction to our discussion, since we deal with connected networks that are the LCP of trees and not with *any* LCP of trees.

3 Designing Compact Routing Schemes for T-networks

Let $G = T_1 \times T_2$, where both T_1 and T_2 are trees. In [12] an IRS for trees has been shown. Thus, consider the two IRSs for T^1 and T^2 and let $\mathcal{L}_1$ and $\mathcal{I}_1$, $\mathcal{L}_2$ and $\mathcal{I}_2$ be the node- and link-labelings of an IRS for T^1 and T^2, respectively. A node $(u_1,u_2) \in V(G)$ is labeled with a triple (a,α,l) if $\mathcal{L}_1(u_1) = a$, $\mathcal{L}_2(u_2) = \alpha$ and l is the layer of both u_1 in T_1 and u_2 in T_2 (and of (u_1,u_2) in G). Similarly, a link $((u_1,u_2),(v_1,v_2)) \in E$ is labeled with a triple (I_1,I_2,l), where $I_1 = \mathcal{I}_1((u_1,v_1))$, $I_2 = \mathcal{I}_2((u_2,v_2))$ and l is the layer of (v_1,v_2). It is possible to rename all nodes according to their node-labeling $\mathcal{L}$; therefore, in the following we speak about *labeling* to mean link-labeling and we refer to nodes themselves to mean their node-labels.

To complete the definition of the compact routing scheme for G, we must describe algorithm $\mathcal{A}$ stored in each node (a, α, l_a) used to route a packet onto a shortest path connecting the current node (a, α, l_a) itself to the destination (t, τ, l_t). Informally, at each step $\mathcal{A}$ tries to take the greedy choice: if both shortest path factors $P_1(a,t)$ and $P_2(\alpha, \tau)$ move towards the same level then $\mathcal{A}$ moves in that direction, that is, it chooses link $((a, \alpha, l_a), (b, \beta, l_b))$. Otherwise, if $a = t$ (or $\alpha = \tau$), then P_1 (or P_2) is null, so the other path must be followed[1]. Finally, if $P_1(a,t)$ and $P_2(\alpha, \tau)$ go towards opposite levels, $\mathcal{A}$ follows the path going away from level l_t. We will prove that in this way a shortest path on G is always used. Now, we are ready to describe algorithm $\mathcal{A}$ formally.

Algorithm $\mathcal{A}$

(**input**: t, τ, l_t; **output**: one outgoing link labeled (I_1, I_2, l))
{ The labeling of the outgoing links, a, α and l_a are known constants in each node.}

if $a = t$ **and** $\alpha = \tau$ **then** extract the packet
else if there exists a link labeled (I_1, I_2, l) s.t. $t \in I_1$ **and** $\tau \in I_2$ **then** choose it
    else if there exists a link having label (I_1, I_2, l) s.t.
        $t \in I_1$ **and** ($\alpha = \tau$ **or** $|l_a - l_t| < |l - l_t|$) **then** choose it
      else if there exists a link having label (I_1, I_2, l) s.t.
          $\tau \in I_2$ **and** ($a = t$ **or** $|l_\alpha - l_\tau| < |l - l_\tau|$) **then** choose it;

Notice that algorithm $\mathcal{A}$, stored in each node, does not increase the asympthotic space complexity with respect to 2-IRS and runs in $\mathcal{O}(\delta)$ time, δ being the maximum node degree. As a consequence, if $\mathcal{A}$ is able to route packets to their destinations we have designed a compact routing scheme. In the following, we first show the optimality (Thm. 1) and then its overall optimality (Thm. 2).

Theorem 1. *If (a, α, l_a) transmits a packet, whose destination is (t, τ, l_t), to an adjacent node (b, β, l_b), then (b, β, l_b) belongs to a shortest path from (a, α, l_a) to (t, τ, l_t).*

Proof. If G is the LCP of either two root-trees or two leaf-trees the statement is trivially true because, from Fact 22, there exists a unique path between any pair of nodes. It must necessarily be the Layered Cross Product of the corresponding paths in the factors. Thus, links labeled (I_1, I_2, l) such that $t \in I_1$ and $\tau \in I_2$ can always be used. Therefore, from now on we shall always suppose that T_1 is a root-tree and T_2 is a leaf-tree.

The proof considers the truth of the **if**-conditions in $\mathcal{A}$.

1. There exists a link (I_1, I_2, l) such that $\tau \in I_2$ and $a = t$. Let $\langle \alpha, \beta, \beta_1, \ldots, \beta_k, \tau \rangle$ be the shortest path in T_2. Then, by definition of LCP, there exist $b, b_1, \ldots, b_k$ in T_1 such that $\langle a, b, b_1, \ldots, b_k, a \rangle$ is a path (crossing the same links more than

[1] When we say that $\mathcal{A}$ follows a path P_i (either $i = 1$ or $i = 2$), we mean that $\mathcal{A}$ follows an edge on G whose i-th factor belongs to P_i.

once) such that $\langle(a,\alpha,l_a),(b,\beta,l_b),(b_1,\beta_1,l_{b_1}),\ldots,(b_k,\beta_k,l_{b_k}),(a,\tau,l_a)\rangle$ is a path in G starting at (a,α,l_a) and ending at (a,τ,l_a). Fact 21 ensures that this is a shortest path in G. The case with $t \in I_1$ and $\alpha = \tau$ is symmetric.

2. There exists a link labeled (I_1, I_2, l_b) such that $t \in I_1$ and $\tau \in I_2$. W.l.o.g., suppose $l_a < l_t$: b is a child of a and β is the father of α (Fig. 2). Moreover, since b belongs to a shortest path from a to t, t is a descendant of b. Two cases are possible:

   - τ is an ancestor of α (Fig. 2.a). Then, the shortest paths from a to t in T_1 and from α to τ in T_2 have the same length $l_t - l_a$. It is easy to see that in G there exists a unique shortest path from (a,α,l_a) to (t,τ,l_t). Furthermore, all links in such path are labeled (I_1, I_2, l) such that $t \in I_1$ and $\tau \in I_2$. Thus, (b,β,l_b) belongs to a path of length $l_t - l_a$ and for Fact 21 it is the shortest path.
   - α and τ have a common ancestor (Fig. 2.b). Let γ be the nearest common ancestor and let l_c be its layer. By the definition of layers, and since T_1 is a root-tree and T_2 is a leaf-tree, it must be $l_t < l_c$.

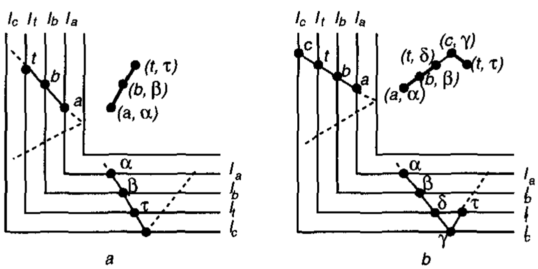

Fig. 2. $l_a < l_t$; a. τ is an acestor of α; b. α and τ have a common ancestor.

Let c be one of the descendants of t at layer l_c in T_1 and δ be the node at layer l_t belonging to the shortest path from α to γ in T_2. There exists a path from (a,α,l_a) to (t,δ,l_t) in G whose links are always labeled (I_1,I_2,l) such that $t \in I_1$ and $\tau \in I_2$ and having length equal to the length of the shortest path from α to δ in T_2 (this is easily proved by induction on the length). The path in G from (t,δ,l_t) to (t,τ,l_t) chosen by $\mathcal{A}$ is a shortest one because of case 1. Since the path from (a,α,l_a) to (t,τ,l_t) passing through (b,β,l_b) has the same length as the path from α to τ in T_2, it is a shortest one.

3. No outgoing link from (a,α,l_a) is labeled (I_1,I_2,l) such that $t \in I_1$ and $\tau \in I_2$; furthermore, neither $a = t$ nor $\alpha = \tau$. Then, one shortest path factor moves towards increasing layers while the other shortest path factor moves towards decreasing layers.
   Suppose first $l_t > l_a$, that is, t is closer than a to the leaves in T_1 while τ is closer than α to the root in T_2.

Notice that it is not possible to have both shortest paths factors moving towards the leaves. Thus, both shortest paths factors move towards the roots. Then, t and a have a nearest common ancestor c at layer l_c. On T_2, either τ is an ancestor of α (see Fig. 3) or τ and α have a nearest common ancestor δ at layer l_d (Fig. 4).

Consider the case of Fig. 3 first. Let b the father of a in T_1 and let I_1 be the label of (a,b) in T_1. Since $l_t - l_b > l_t - l_a$ and $t \in I_1$, $\mathcal{A}$ chooses one link labeled (I_1, I_2, l_b) and ending in (b, β, l_b). Let d be the node belonging to the shortest path from a to t at layer l_a in T_1. Node (b, β, l_b) lies on the path

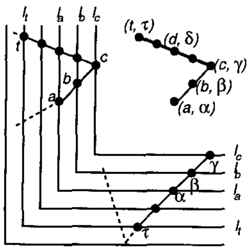

Fig. 3. The two shortest paths factors move towards the roots and τ is an ancestor of α while t and a have a common ancestor c.

from (a, α, l_a) to (d, α, l_a) – through a node (c, γ, l_c) – of the same length as the shortest path from a to d in T_1. From this last node, there is a single shortest path to (t, τ, l_t) constituted by links always labeled (I_1, I_2, l) such that $t \in I_1$ and $\tau \in I_2$. The length of such path from (a, α, l_a) to (t, τ, l_t) through (b, β, l_b) is equal to the length of the shortest path from a to t in T_1.

Consider now the situation in Fig. 4: both shortest paths factors move towards the roots, t and a have a nearest common ancestor c at layer l_c, and τ and α have a nearest common ancestor δ at layer l_d. To go from (a, α, l_a) to (t, τ, l_t) through a shortest path it is necessary to use either first a shortest path in T_1 till c or first a shortest path in T_2 till δ. Indeed, consider a path that alternates links whose first factor is from a shortest path in T_1 with links whose second factor is from a shortest path in T_2: let $(a, \alpha, l_a), (a_1, \alpha_1, l_1), \ldots, (a_k, \alpha_k, l_k)$ be the first fragment of such a path having the first factor as the shortest path in T_1, with $a_i \neq c$, $i = 1, \ldots, k$. If the next step is a link whose second factor belongs to a shortest path towards τ in T_2, such link ends necessarily in $(u, \alpha_{k-1}, l_{k-1})$, where u is a child of a_k in T_1. For the sake of symmetry, since u and a_{k-1} are at the same layer in the same subtree rooted at a_k (with $l_k > l_c$), the distance of $(u, \alpha_{k-1}, l_{k-1})$ from (t, τ, l_t) is equal to the distance of $(a_{k-1}, \alpha_{k-1}, l_{k-1})$ from (t, τ, l_t). Thus, in order to reach (t, τ, l_t) from (a, α, l_a) at least two steps have been wasted. Hence, necessarily one of the following is a shortest path in G:

- (Fig. 4.a) a path from (a, α, l_a) to (b, β, l_b) to (c, γ, l_c) to (h, α, l_a) to (t, η, l_t) to (f, δ, l_d) to (t, τ, l_t), where γ is a descendant of α at layer l_c in T_2, h and f are descendants of c at layer l_a and l_d, respectively, in T_1 and η is an ancestor of α at layer l_t in T_2. The length of such a path is $2(l_a - l_c) + (l_t - l_a) + 2(l_d - l_t) = 2(l_d - l_c) + l_a - l_t$
- (Fig. 4.b) a path from (a, α, l_a) to (d, δ, l_d) to (k, τ, l_t) to (a, θ, l_a) to (c, σ, l_c) to (h, θ, l_a) to (t, τ, l_t), where d and k are descendant of a at layers l_d and l_t, respectively, in T_1 and θ and σ are descendants of τ at layers l_a and l_c, respectively, in T_2. The length of such a path is $(l_d - l_a) + (l_d - l_t) + 2(l_t - l_c) = 2(l_d - l_c) + l_t - l_a$

Thus, the shortest path depends on the sign of $l_t - l_a$. Since in our hypothesis $l_t > l_a$, the shortest path is the first one, that is the path whose first step goes away from l_t. Since node (b, β, l_b) is such that $l_t - l_b > l_t - l_a$, then $\mathcal{A}$ chooses the first path.
When $l_t < l_a$, the same reasoning applies. Finally, the same discussion holds also if $l_t = l_a$. However, notice that in this case any of the two choices is possible.

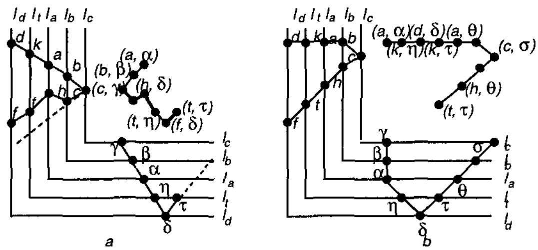

Fig. 4. The two shortest paths factors move towards the roots, t and a have a common ancestor c and τ and α have a common ancestor δ.

Theorem 2. *If a packet must be transmitted from node* (a, α, l_a) *to node* (t, τ, l_t) *and* $\langle (a, \alpha, l_a), (a_1, \alpha_1, l_{a_1}), (a_2, \alpha_2, l_{a_2}), \ldots, (t, \tau, l_t) \rangle$ *is any shortest path from* (a, α, l_a) *to* (t, τ, l_t), *then algorithm* $\mathcal{A}$ *can possibly use it.*

Proof. Again, thanks to Fact 22, we shall always suppose that T_1 is a root-tree and T_2 is a leaf-tree. The proof is divided according to the truth of the **if**-conditions in $\mathcal{A}$ and most of considerations done in the proof of Theorem 1 are used here.

1. There exists a link (I_1, I_2, l) such that $\tau \in I_2$ and $a = t$: then (cf. proof of Thm.1) G contains a path from (a, α, l_a) to (t, τ, l_t) of length equal to the path from α to τ in T_2. Let $k + 1$ be such length. Suppose that a path $\langle (a, \alpha, l_a), (a_1, \alpha_1, l_{a_1}), \ldots, (a_k, \alpha_k, l_{a_k}), (t, \tau, l_t) \rangle$ exists in G that is not found

by $\mathcal{A}$. Thus, α_1 does not belong to the shortest path from α to τ in T_2, that is, if link $((a, \alpha, l_a), (a_1, \alpha_1, l_{a_1}))$ is labeled (I_1, I_2, l_{a_1}), $\tau \notin I_2$. Hence, since by definition of LCP $p = \langle \alpha, \alpha_1, \ldots, \alpha_k, \tau \rangle$ must be a path in T_2, some α_i must be the same as some α_j, with $i \neq j$. But the distance between α and τ in T_2 is $k+1$, then p must be longer than $k+1$, an absurd. The case $t \in I_1$ and $\alpha = \tau$ is symmetric.

2. There exists a link labeled (I_1, I_2, l) such that $t \in I_1$ and $\tau \in I_2$. W.l.o.g., suppose $l_a < l_t$ and therefore b a child of a and β the father of α. Since b belongs to a shortest path from a to t, t is a descendant of b. Two cases are possible:
   - τ is an ancestor of α. This case is trivially proved, since there is in G a unique shortest path from (a, α, l_a) to (t, τ, l_t) and $\mathcal{A}$ finds that path.
   - α and τ have a nearest common ancestor γ at layer l_c. Recall that $l_t \leq l_c$ and a shortest path in G must be as long as the path from α to τ in T_2 (cf. proof of Thm.1). The proof of this case is similar to the one of case 1. of this theorem and thus omitted.
3. No outgoing link from (a, α, l_a) is labeled (I_1, I_2, l) such that $t \in I_1$ and $\tau \in I_2$. Suppose $l_t > l_a$. We have already proved that one of the following cases must occur:
   - both shortest paths factors move towards the roots, τ is an ancestor of α while t and a have a nearest common ancestor c at layer l_c: algorithm $\mathcal{A}$ chooses a link labeled (I_1, I_2, l) such that $t \in I_1$ and such link belongs to a path from (a, α, l_a) to (t, τ, l_t) having the same length as the path from a to t in T_1. Again, a reasoning very similar to that one of case 1. of this theorem applies.
   - both shortest paths factors move towards the roots, t and a have a nearest common ancestor c at layer l_c, τ and α have a nearest common ancestor δ at layer l_d. We have already proved that a shortest path from (a, α, l_a) to (t, τ, l_t) necessarily crosses either first a shortest path in T_1 to c or first a shortest path in T_2 to δ. This implies (cf. proof of Thm.1) that a shortest path in G must necessarily be a path from (a, α, l_a) to (c, γ, l_c) to (h, α, l_a) to (t, η, l_t) to (d, δ, l_d) to (t, τ, l_t), where γ is a descendant of α at layer l_c in T_2, h is a descendant of c at layer l_a in T_1 and η is an ancestor of α at layer l_t in T_2 (Fig. 4.a). Since $\mathcal{A}$ can choose any link $((a, \alpha, l_a), (b, \beta, l_b))$ labeled (I_1, I_2, l_b) such that $l_t - l_b > l_t - l_a$ and $t \in I_1$, then it is able to choose any path of this sort, and this proves the assertion.

The same reasoning applies when $l_t < l_a$ and when $l_t = l_a$.

4 LCP of General Graphs: Driving some Conclusions

In the previous section we have proposed a method to compute a compact routing scheme for all T-networks. In the proofs of correctness, we have strongly used the properties of the factors and having a unique (shortest) path between any couple of nodes. It is immediate to wonder whether our technique can be extended to networks which are the LCP of more general graphs. In this section we show a negative result in this direction. Namely, we prove a property of the LCP allowing one to deduce that the knowledge of node- and link-labels on factors (and therefore the knowledge of their shortest paths) gives not enough information to find shortest paths in their LCP.

Theorem 3. *There exist a layered graph $G = (V, E)$ LCP of $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a source node $(s, \sigma, l_s) \in V$, a destination node $(t, \tau, l_t) \in V$ and an edge $e \in E$ having the following properties: i. e is the layered cross product of two edges $e_1 \in E_1$ and $e_2 \in E_2$; ii. e belongs to a shortest path from (s, σ, l_s) to (t, τ, l_t); iii. neither e_1 belongs to a shortest path from s to t in G_1, nor e_2 belongs to a shortest path from σ to τ in G_2.*

Proof. The graph in the assertion is shown in Fig.5 in which the following convention is used: each edge in the drawing of G_1 and G_2 (and therefore of G) represents a simple chain whose length is determined by the difference of layers where its extremes lie.

Suppose the shortest path from s to t in G_1 passes through m and the shortest path from σ to τ passes through ν. Then, the following relations must hold:
$|l_s - l_t| + 2|l_t - l_m| < 2|l_s - l_d| + |l_s - l_t|$ and $2|l_s - l_n| + |l_s - l_t| < |l_s - l_t| + 2|l_t - l_r|$
that is $|l_m - l_t| < |l_s - l_d|$ and $|l_s - l_n| < |l_t - l_r|$.

Whenever one of the following inequalities hold
$|l_s - l_t| + 2|l_t - l_r| < 2|l_s - l_n| + |l_s - l_t| + 2|l_t - l_m|$
$2|l_s - l_d| + |l_s - l_t| < 2|l_s - l_n| + |l_s - l_t| + 2|l_t - l_m|$
either the path through (m, μ_1, l_m) and (r, ρ, l_r) (first inequality) or the path through (n_2, ν, l_n) and (d, δ, l_d) (second inequality) are shorter than the path through (n_2, ν, l_n) and (m, μ_2, l_m). That is, the shortest path passes through an edge – either $((m, \mu_1, l_m), (r, \rho, l_r))$ or $((n_2, \nu, l_n), (d, \delta, l_d))$ – whose factors are not on a shortest path.

As a consequence of the previous theorem, we can state the following fact:

Fact 41 *Let G be a network that is the LCP of any two graphs G_1 and G_2. The only knowledge of the compact routing schemes (i.e. a node- and link-labeling scheme) on G_1 and G_2 may not be sufficient to deduce a compact routing scheme for G.*

Anyway, the previous claim does not forbid one to find some special cases in which the particular structure either of the network itself or of its factors helps in defining a compact routing scheme.

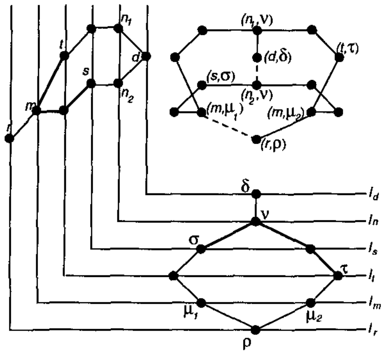

Fig. 5. Edges on a shortest path in G whose factors do not lie on any shortest path in G_1 and in G_2.

Acknowledgments: We thank Richard B. Tan for a careful reading of earlier drafts and for his helpful suggestions on improving the paper.

References

1. S. Even, A. Litman, "Layered Cross Product - A technique to construct interconnection networks", 4^{th} *ACM SPAA*, 60-69, 1992.
2. M. Flammini, J. van Leeuwen, A. Marchetti-Spaccamela, "The complexity of interval routing in random graphs", *MFCS 1995*, LNCS 969, 37-49, 1995.
3. P. Fraigniaud, C. Gavoille, "Optimal interval routing", *CONPAR*, LNCS 854, 785-796, 1994.
4. C. Gavoille, S. Perennes, "Lower bounds for shortest path interval routing", *SIROCCO'96*, 1996.
5. R. Kráľovič, P. Ružička, D. Štefankovič, "The complexity of shortest path and dilation bounded interval routing", *Euro-Par'97*, LNCS 1300, 258-265, 1997.
6. J. van Leeuwen, R.B. Tan, "Computer networks with compact routing tables", in: G. Rozemberg and A. Salomaa (Eds.) *The Book of L*, Springer-Verlag, Berlin, 298-307, 1986.
7. J. van Leeuwen, R.B. Tan, "Interval routing", *Computer Journal*, 30, 298-307, 1987.
8. J. van Leeuwen, R.B. Tan, "Compact routing methods: a survey", *Colloquium on Structural Information and Communication Complexity (SICC'94)*, 1994.
9. F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, Inc.,1992.
10. C.E. Leiserson, "Fat-Trees: universal networks for hardware-efficient supercomputing", *IEEE Trans. on Comp.*, 34, 892-901, 1985.
11. P. Ružička, "On efficiency of Interval Routing algorithms", in: M.P. Chytil, L. Janiga, V. Koubeck (Eds.), *MFCS 1988*, LNCS 324, 492-500, 1988.
12. N. Santoro, R. Khatib, "Routing without routing tables", *Tech. Rep. SCS-TR-6*, School of Computer Science, Carleton University, 1982. Also as: "Labelling and implicit routing in networks", *Computer Journal*, 28, 5-8, 1985.

3. S.S.H. Tse, F.C.M. Lau, "A lower bound for interval routing in general networks", *Tech. Rep. 94-04*, Dept. of Computer Science, The University of Hong Kong, 1994 (to appear in *Networks*).