

Edge-Disjoint Spanning Trees on Star-Product Networks

Daniel Hwang, Georgia Institute of Technology | Mentor: Laura Monroe, Los Alamos National Laboratory (HPC/USRC)

Star-Product Networks

REPRESENTING NETWORKS AS GRAPHS

A network can be represented by a set of nodes connected by another set of wires. This can be formalized by the mathematical structure of a **graph** (V, E) , which is a set of vertices V combined with a set of edges E between them. Instead of constructing a big graph from scratch, we can take two smaller graphs and combine them by using a graph product. For this presentation, we are interested in the star product.

CARTESIAN AND STAR PRODUCT NETWORKS

Given two graphs, which we call the **structure graph** $G_s = (V_s, E_s)$ and the **supernode** $G_n = (V_n, E_n)$, we define their **Cartesian product** $G_s \times G_n$ to be the graph with vertex set $V_s \times V_n$ and edge set

$$\begin{aligned} & \{((x, y_1), (x, y_2)) \mid x \in V_s, (y_1, y_2) \in E_n\} \\ & \cup \{((x_1, y), (x_2, y)) \mid (x_1, x_2) \in E_s, y \in V_n\}. \end{aligned}$$

Intuitively, we place a copy of G_n at each vertex of V_s (each connected by the edges in E_n) and then connect these copies by replacing each edge of E_s with a set of edges connecting corresponding vertices in V_n . We can similarly construct the star product by placing a copy of G_n at each vertex of V_s and then connect copies $\{x_1\} \times V_n$ and $\{x_2\} \times V_n$ by using an arbitrary bijection between the vertices $f_{(x_1, x_2)}: V_n \rightarrow V_n$.

Formally, the **star product** $G_s * G_n$ is defined as the graph with vertex set $V_s \times V_n$ and edge set

$$\begin{aligned} & \{((x, y_1), (x, y_2)) \mid x \in V_s, (y_1, y_2) \in E_n\} \cup \\ & \{((x_1, y), (x_2, f_{(x_1, x_2)}(y))) \mid (x_1, x_2) \in E_s, y \in V_n\}. \end{aligned}$$

See Figure 1 for an example of this construction.

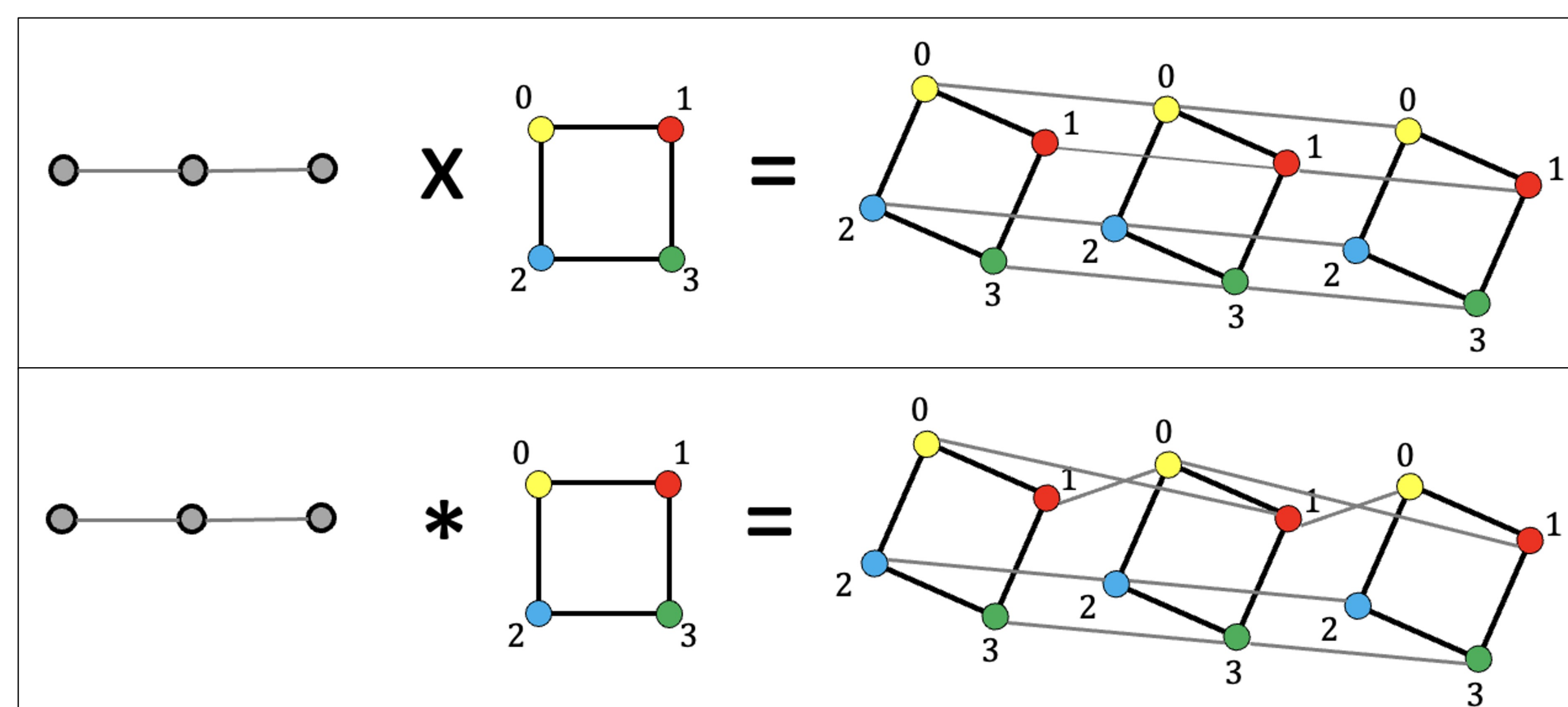


Figure 1. A quick demonstration of the Cartesian product and the star product.

EXISTING STAR PRODUCT NETWORK IMPLEMENTATIONS

Turns out a lot of existing network topologies are star products! The Mesh, Torus, and HyperX topologies are all Cartesian products, and more generally, the Slim Fly, BundleFly, and Chimera networks are star products, the latter of which was used in the DWave 2000Q. Recently, our team has constructed a new star product PolarStar, which takes the star product of PolarFly with the Paley graph or our own IQ graph.

PolarFly is a mathematically designed network of diameter 2 which asymptotically reaches the Moore maximum bound on the number of nodes on a network with fixed degree and diameter. By using PolarFly as its structure graph, PolarStar currently achieves the largest known diameter-3 network topologies for almost all radices and has been submitted for this year's R&D 100. Since star products are generalizations of Cartesian networks, they share some properties with Cartesian networks, but improved, such as modular structure, small diameter, flexibility in design, and as we will discuss, parallelism.

Edge-Disjoint Spanning Trees

WHAT ARE EDSTs AND WHY ARE THEY IMPORTANT?

An important question about any network is: how many disjoint paths can one take to travel between every node? One obvious application of this question is parallelism: if there are multiple paths available to travel between nodes on a network, one can maximize the number of messages that can be sent on the network at the same time. Moreover, the network's fault tolerance and bandwidth are improved at the same time. We formally define disjoint paths as **edge-disjoint spanning trees (EDSTs)**, which are edge-disjoint if they do not share any edge with each other and spanning trees if they connect all the vertices without forming any cycles. We now formalize our research question as:

Given any structure graph G_s and supernode G_n (both connected and simple), how many EDSTs can we find on their star product?

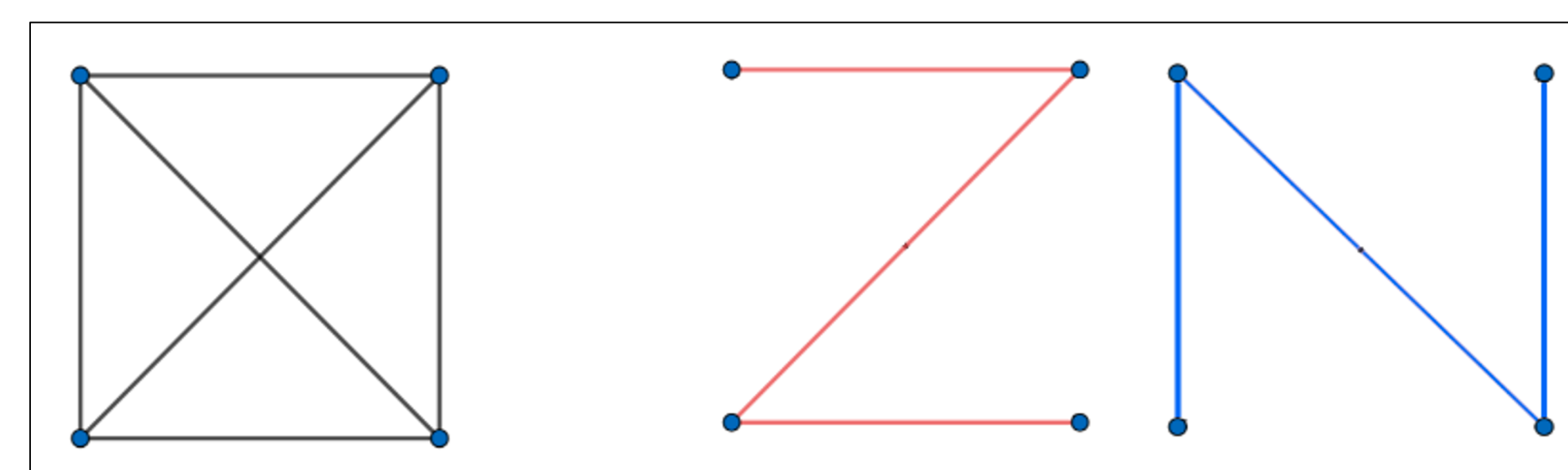


Figure 2. The complete graph K_4 with its two EDSTs on the right.

CONSTRUCTING EDSTs

If the structure graph G_s has t_s EDSTs and the structure graph G_n has t_n EDSTs, **we can construct $t_s + t_n - 2$ EDSTs without any restrictions**, and we go over these constructions now. Let X_1, X_2, \dots, X_{t_s} be the set of EDSTs on G_s and let Y_1, Y_2, \dots, Y_{t_n} be the set of EDSTs on G_n . For our first construction type, we will set aside Y_1 and maximize the use of X_i on the other vertices, where $i \neq 1$. Varying across i , this yields $t_s - 1$ EDSTs. Now we describe the construction:

Construction 1

1. Connect one supernode using Y_1 .
2. Connect all other supernodes by using $|V_n|$ copies of the tree X_i . Formally, if o is the root of the directed tree X_i , then draw in all paths of the form $\{(o, v), (x_1, f_{(o, x_1)}(v)), \dots\}$.

A visual representation is given in Figure 3.

Similarly, for our second construction, we set aside X_1 and maximize the use of Y_i on all supernodes, where $i \neq 1$, yielding $t_n - 1$ EDSTs:

Construction 2

1. Connect all supernodes using Y_i .
2. Fix $v \in V_n$. Connect the supernodes by using the directed tree X_i to connect all (x_2, v) with their ancestors, more formally, with the edges $((x_1, f_{(x_1, x_2)}^{-1}(v)), (x_2, v))$.

A visual representation of the above is given in Figure 3.

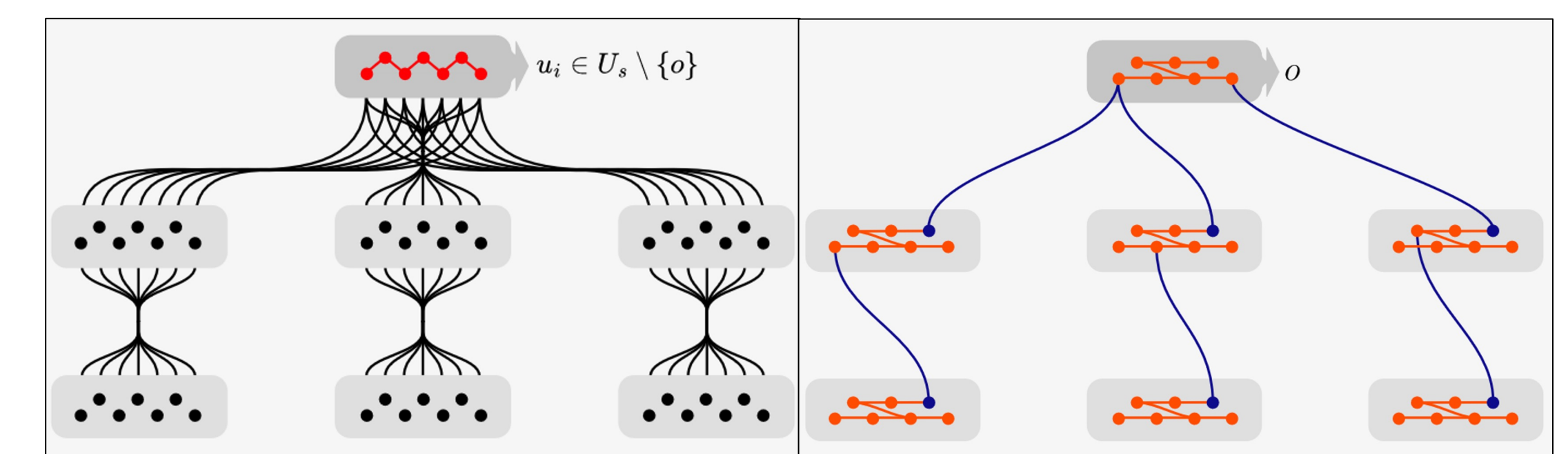


Figure 3. A visual representation of Constructions 1 (left) and 2 (right).

Additional Analysis

ADDITIONAL CONSTRUCTIONS

If the structure graph G_s has r_s spare edges (not used in the EDSTs) and $r_s \geq t_s$, then we can get one more spanning tree (and similarly for the supernode G_n). When $r_s = t_s$ and $r_n = t_n$, we get maximality.

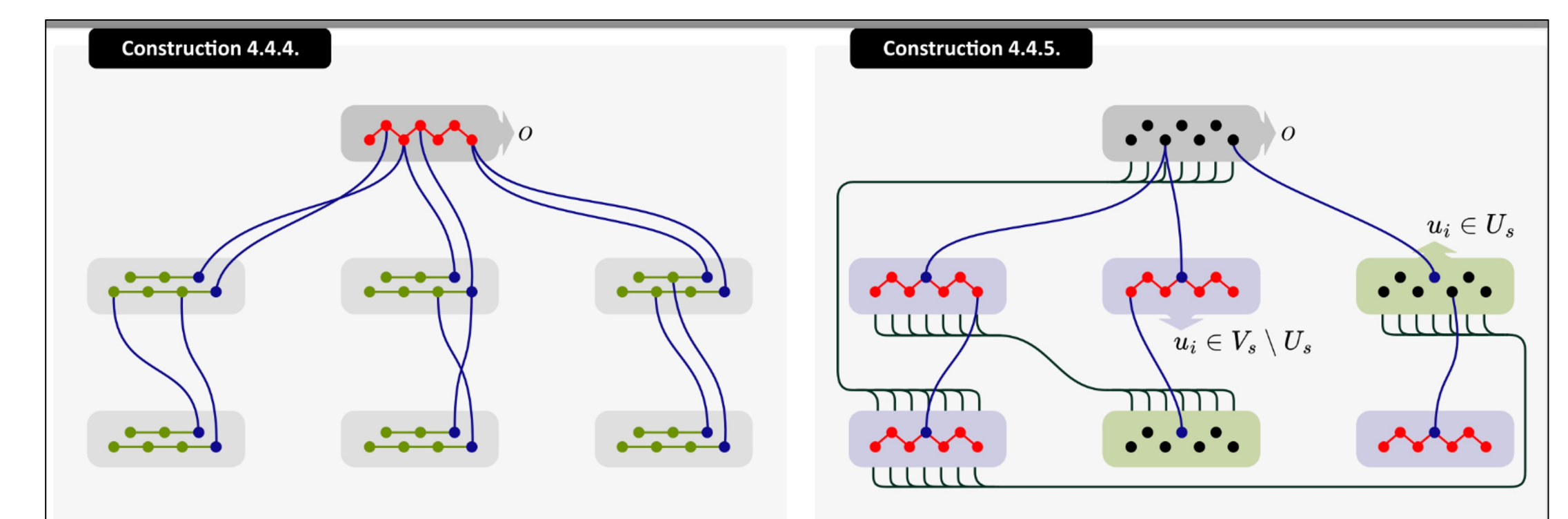


Figure 4. A visual representation of additional constructions possible.

DEPTH ANALYSIS

In addition to maximizing the number of EDSTs, we can lower their depth by carefully choosing our star product edges.

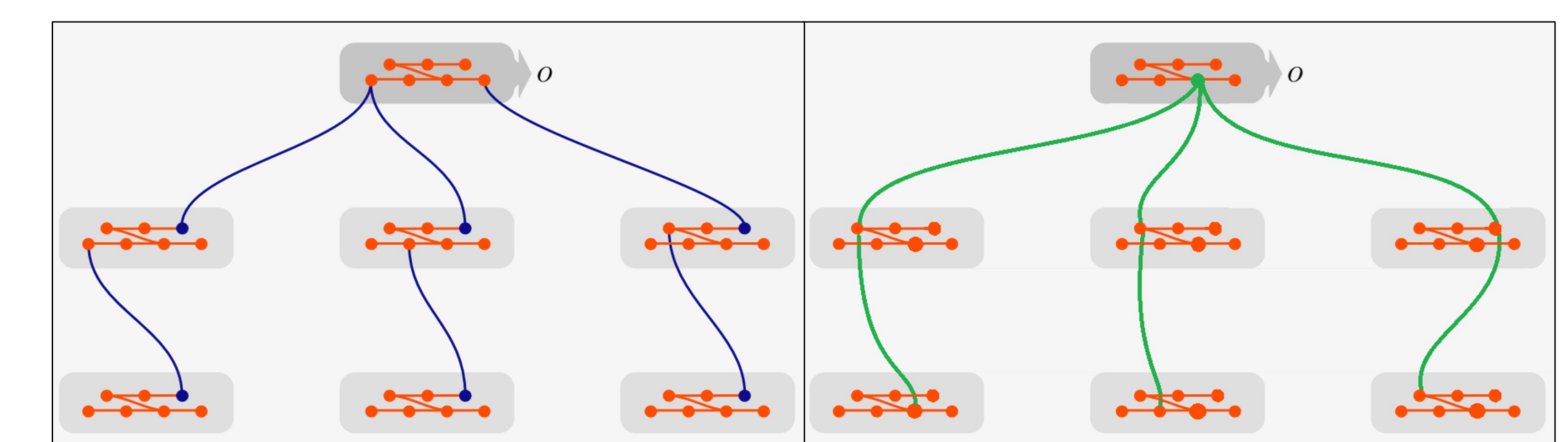


Figure 5. Construction 2 compared with a modified tree with lower depth (right).

