ILP Model Definitions

Victor Sberse Guerra and Gabriel Luca Nazar

May 2025

In modern networking, providing customized services often involves routing network traffic through a specific sequence of functions, known as a Service Function Chain (SFC). These functions can include firewalls, load balancers, or data processors. To efficiently deploy these SFCs, network operators must make complex decisions about where to place each function and how to route the traffic, all while respecting the limitations of the network's hardware and infrastructure.

A powerful tool for solving such complex optimization problems is Integer Linear Programming (ILP). An ILP model uses mathematical equations to represent the problem's objective (e.g., maximizing accepted services), its decision variables (e.g., where to place a function), and its constraints (e.g., resource capacity). By solving this model, we can find the optimal way to allocate resources to meet service demands.

This document outlines three ILP models for allocating SFCs on a network where some nodes are equipped with Field-Programmable Gate Arrays (FPGAs), which are specialized, reconfigurable hardware accelerators. We begin with the foundational model, named Flex.

The Flex ILP Model

The Flex model is the foundational framework for optimizing the placement of service functions on an FPGA-accelerated network. Its primary goal is to maximize the total "value" of the accepted service requests, where value is tied to the resources required. The "Flex" in its name comes from its flexibility; it simultaneously decides:

- Which service requests to accept.
- Where to place the FPGAs in the network.
- How to partition the resources on each FPGA.
- Which network path to use for each accepted request.
- Where to allocate each function of a service chain along the chosen path.

By solving for all these variables at once, the Flex model provides a comprehensive and optimal solution for a given set of service demands. The model is defined by its sets, parameters, decision variables, objective function, and a series of constraints that ensure the solution is both valid and practical.

Sets

- $\mathcal{R}$: Set of requisitions. Each requisition represents a combination of information about a network service.
- $\mathcal{F}_r$: Set of functions of a requisition $r \in \mathcal{R}$. Each requisition may have multiple functions, Service Function Chain (SFC).
- $\mathcal{P}$: Set of partitions. Each node can have multiple FPGA partitions with specific computational resources.
- $\mathcal{K}$: Set of paths.
- $\mathcal{N}$: Set of network nodes. These are the locations in the network topology.
- $\mathcal{L}$: Set of network links. Links that connect nodes in the network topology.

Parameters

- value_r: Value associated with requisition r. This represents a monetary arbitrary value that is tied to the amount of resources needed by the requisition.
- thr_r: Minimum required throughput for requisition r. It indicates the minimum data rate the path for this requisition must support.
- lat_r: Maximum allowed latency for requisition r. This defines the maximum time the requisition can take to travel from its origin to its destination. Here it is included the execution latency and the travel latency.
- clb_f, bram_f, dsp_f: Resource requirements for function f. These are the FPGA resources (CLB, BRAM, DSP) needed by each function.
- $\text{clb}_{n,p}$, $\text{bram}_{n,p}$, $\text{dsp}_{n,p}$: Available resources in partition p at node n. These indicate how much of each resource is available in each partition.
- t_{ij}: Throughput of link (i, j). The data transfer capacity between nodes i and j.
- l_{ij}: Latency of link (i, j). The time delay associated with the link between nodes i and j.

Decision Variables

- $x_{r,f,n,p,k} \in \{0,1\}$: 1 if function f of requisition r is allocated at node n, partition p, along path k. Otherwise, it is 0.
- $y_r \in \{0,1\}$: 1 if requisition r is accepted, meaning all its required functions are allocated on the network.
- $z_{r,k} \in \{0,1\}$: 1 if path k is used for requisition r. This variable helps select the path for the requisition.
- $w_{n,p} \in \{0,1\}$: 1 if partition p is active at node n. This indicates whether the partition is being used for function allocation.
- $\texttt{node\_fpga}_n \in \{0,1\}$: Indicates whether node n has an FPGA installed.

Objective Function

The objective of the optimization model is to maximize the total value of the accepted requisitions. The decision variable y_r determines whether a requisition is accepted, and the value of the requisition is multiplied by this decision variable:

$$\max \sum_{r\in\mathcal{R}} \text{value}_r \cdot y_r$$

Constraints

1. requisition acceptance A requisition can only be accepted if all its functions are allocated in the network. The sum of the allocation decisions across all possible paths and nodes must be at least as large as y_r for each requisition:

$$y_r \leq \sum_{k\in\mathcal{K}} \sum_{f\in\mathcal{F}_r} \sum_{n\in k} \sum_{p\in\mathcal{P}} x_{r,f,n,p,k} \quad \forall r \in \mathcal{R} \tag{1}$$

2. Function allocation Each function of a requisition can be allocated at most once in the network. This constraint ensures that no function is duplicated in the allocation:

$$\sum_{k\in\mathcal{K}} \sum_{n\in k} \sum_{p\in\mathcal{P}} x_{r,f,n,p,k} \leq 1 \quad \forall r \in \mathcal{R}, f \in \mathcal{F}_r \tag{2}$$

3. One path per requisition A requisition can only use one path at a time. This constraint ensures that the path selection for each requisition is binary and that the requisition can only be routed along one path:

$$\sum_{k\in\mathcal{K}} z_{r,k} = y_r \quad \forall r \in \mathcal{R} \tag{3}$$

4. **Path implies function allocation** If a path is selected for a requisition, then the functions of the requisition must be allocated along that path in the network:

$$x_{r,f,n,p,k} \leq z_{r,k} \quad \forall r, f \in \mathcal{F}_r, n \in k, p \in \mathcal{P}, k \in \mathcal{K} \tag{4}$$

5. **Partition usage** Each partition can only be used by one function at a time. This constraint ensures that no partition is over-allocated:

$$\sum_{r \in \mathcal{R}} \sum_{f \in \mathcal{F}_r} \sum_{k \in \mathcal{K}} x_{r,f,n,p,k} \leq 1 \quad \forall n \in \mathcal{N}, p \in \mathcal{P} \tag{5}$$

6. **Throughput constraint** The selected path for each requisition must provide enough throughput to meet the requisition's requirements. The minimum throughput along the path must satisfy the required throughput:

$$\sum_{k \in \mathcal{K}} \min_{(i,j) \in k} t_{ij} \cdot z_{r,k} \geq \text{thr}_r \cdot y_r \quad \forall r \in \mathcal{R} \tag{6}$$

7. **Latency constraint** The total latency of the selected path must not exceed the maximum allowed latency for the requisition:

$$\sum_{k \in \mathcal{K}} \sum_{(i,j) \in k} l_{ij} \cdot z_{r,k} \leq \text{lat}_r \cdot y_r \quad \forall r \in \mathcal{R} \tag{7}$$

8. **CLB capacity** The total amount of CLB resources used by the allocated functions in each partition must not exceed the available CLB resources in the partition:

$$\sum_{r \in \mathcal{R}} \sum_{f \in \mathcal{F}_r} \sum_{k \in \mathcal{K}} x_{r,f,n,p,k} \cdot \text{clb}_f \leq \text{clb}_{n,p} \quad \forall n, p \tag{8}$$

9. **BRAM capacity** The total amount of BRAM resources used by the allocated functions in each partition must not exceed the available BRAM resources in the partition:

$$\sum_{r \in \mathcal{R}} \sum_{f \in \mathcal{F}_r} \sum_{k \in \mathcal{K}} x_{r,f,n,p,k} \cdot \text{bram}_f \leq \text{bram}_{n,p} \quad \forall n, p \tag{9}$$

10. **DSP capacity** The total amount of DSP resources used by the allocated functions in each partition must not exceed the available DSP resources in the partition:

$$\sum_{r \in \mathcal{R}} \sum_{f \in \mathcal{F}_r} \sum_{k \in \mathcal{K}} x_{r,f,n,p,k} \cdot \text{dsp}_f \leq \text{dsp}_{n,p} \quad \forall n, p \tag{10}$$

11. **One FPGA per node** Each node can have at most one FPGA. This constraint limits the number of FPGAs that can be placed at each node:

$$\sum_{p \in \mathcal{P}} w_{n,p} \leq 1 \quad \forall n \in \mathcal{N} \tag{11}$$

12. **Partition implies FPGA usage** A partition can only be used if the corresponding FPGA at that node is active. This constraint ensures that a partition cannot be used unless an FPGA is assigned to the node:

$$x_{r,f,n,p,k} \leq w_{n,p} \quad \forall r, f \in \mathcal{F}_r, n, p, k \in \mathcal{K} \tag{12}$$

13. **Fix a single partitioning** Ensures that all partitions belonging to the same FPGA on a given node are either all active or all inactive. This maintains consistency in FPGA usage by treating the group of partitions as a unit.

$$w_{n,p} \leq \text{node\_fpga}_n \quad \forall n \in N,\ \forall p \notin \text{last} \tag{13}$$

14. **At Most One FPGA per Node** Enforces a hardware limitation: each node can have at most one FPGA installed.

$$\text{node\_fpga}_n \leq 1 \quad \forall n \in N : \tag{14}$$

15. **Function Allocation Order Along Paths** Ensures that functions within the same request are allocated in the correct order along a chosen path. This preserves the logical sequence of function execution as data flows through the network.

$$\sum_{n \in k} \sum_{p \in P} \text{idx}_k(n) \cdot x_{r,f_i,n,p,k} \cdot z_{r,k} \leq \sum_{n \in k} \sum_{p \in P} \text{idx}_k(n) \cdot x_{r,f_j,n,p,k} \cdot z_{r,k}$$
$$\forall r \in R,\ \forall (f_i, f_j) \in \mathcal{F}_r,\ \forall k \in K \tag{15}$$

Fixed Positioning Model (FixPos)

The second model, FixPos (Fixed Positioning), builds directly upon the Flex model by introducing a significant limitation to reduce the problem's complexity. While it uses all the same constraints as Flex, its defining feature is that the locations of the FPGAs are no longer a decision variable. Instead, the model enforces a specific placement of FPGAs across the network based on a predetermined configuration. This configuration is typically derived from the optimal FPGA locations found by an initial run of the more comprehensive Flex model or a random positioning for other research purposes. The goal of FixPos is to optimize the allocation of services and partition resources within a network where the hardware infrastructure is already established and cannot be changed.

Positioning Enforcement Through the *node_fpga* variable, enforce a certain positioning of the FPGAs into the network topology, given the initial result of Flex, where *Nodes_with_fpga* stores this data.

$$\text{node\_fpga}_n = 1 \quad \forall n \in \text{Nodes\_with\_fpga} \tag{16}$$

Fixed Partitioning Model (FixPart)

The third model, FixPart (Fixed Partitioning), offers a different approach that is conceptually the opposite of the FixPos model. In this scenario, the specific partitioning scheme is fixed, but the location of the FPGAs is not. The model is constrained to use a predetermined set of partition configurations—for instance, a specific number of partitions of a certain size and resource mix—but it retains the flexibility to decide on which network nodes to place these FPGAs to best serve the incoming requests. This model addresses the question: "Given that we have a specific inventory of pre-configured FPGA partition types, where is the most valuable place to deploy them across the network?" It optimizes the positioning of a standardized hardware set, rather than optimizing the hardware configuration for a fixed set of locations.

Partitioning Enforcement Through the partition component p of w, enforce a partitioning in a certain node, given the initial result of the ILP $free$.

$$\sum_{n \in N} w_{n,p(t)} \geq d_t \quad \forall t \in T \tag{17}$$