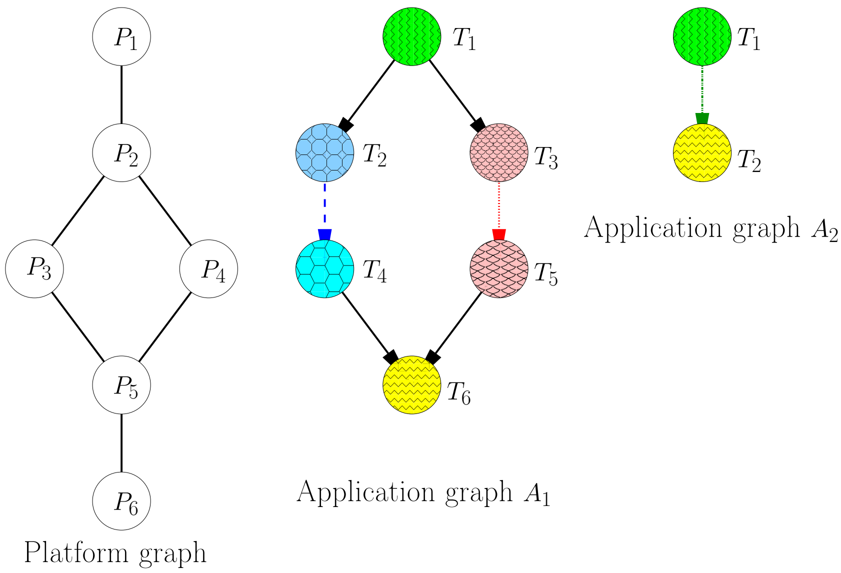

Figure 1: Examples of platform and applications

1 Notations

We have many instances of an application represented by a directed acyclic graph (DAG) to schedule over a computer platform represented by a general graph.

Let $G_A = (V_A, E_A)$ be the directed application graph, where $V_A = \{T_1, \dots, T_n\}$ and $|E_A| = m$. To simplify some equations, we assume, without any loss of generality, that tasks are numbered in a topological order. Especially, T_1 is the first task and T_n is the last one. The size of the file to transmit from T_k to T_l is given by $\text{data}_{k,l}$.

Let $G_P = (V_P, E_P)$ be the undirected platform graph, where $V_P = \{P_1, \dots, P_p\}$. The time to transmit a unitary file from P_q to P_r is given by $c_{q,r}$. Task T_k needs a time $w_{i,k}$ to be entirely processed by processor P_i.

We use a bidirectionnal, bounded multiport model for communications, and we allow the overlap of computations by communications.

We are looking for a steady-state schedule, which maximizes the throughput of the complete platfom. An allocation is a mapping of V_A over V_P and a mapping of E_A over the path of E_P, such that all dependence is correctly processed: any task must have all its files before being processed.

Figure 1 shows an example of platform graph, and two small examples of application graphs.

1.1 Glossary

application: An application is a directed acyclic graph, composed of a set of tasks, or jobs, called T_ks which are the vertices of the graph, and of dependences between tasks, or files, called $T_k \rightarrow T_l$s, which are the edges of the graph. The file $T_k l$ represents a result of T_k and is necessary to the computation of T_l.

platform: A platform is a directed graph, made of a set of processors, or workers, called P_is, which are the vertices of the graph, and of communication links, called $P_i \rightarrow P_j$s, which are the edges of the graph. We assume that this graph is a fully connected graph. TODO comment dit-on fortement connexe ? Any processor can be used for processing tasks and for transmitting a file from a neighbour to another. In other words, there is no distinction between routers and actual computational processors.

allocation: An allocation is a mapping of each task T_k on a set of processors P_i and a mapping of each communication $T_k \rightarrow T_l$ on a set of paths $P_i \rightsquigarrow P_j$. To be valid, any allocation has to satisfy several properties given in Subsection 1.2.

troughput: The throughput of an allocation scheme is the average number of processed allocations in one time unit.

transfer: A transfer is the sending of a file $T_k \rightarrow T_l$ from a source processor P_i, such that T_k is processed by P_i, to a destination processor P_j. The only use to such a transfer is the processing of T_l by P_j, even if we consider allocations as valid when we have useless transfers.

communication: A transfer is made of several communications between neighbours. The source processor sends the file to one of its neighbours, which forwards to one of its own neighbours, and so on, until the file reaches its destination processor. A transfer is made of at least one communication.

path: A path $P_i \rightsquigarrow P_j$ is the sequence of processors $(P_{i_1}, \ldots, P_{i_k})$ traversed by a file during its transfer. A priori, it exists several different paths for a given couple (source, destination).

instance: As said before, we want to schedule multiple instances of the same application graph. Two different instances correspond to two different set of initial data. By example, when we want to apply sequentially several filters, say, k filters, to a set of n pictures, then we have k differents tasks, and n instances. Any of the k tasks has to be processed n times.

copy: Contrary to our definition of instance, two copies of a file or a task correspond to the same initial set of data.

duplication:

1.2 Definition of a valid allocation

A valid allocation is a mapping of each task T_k on a set of processors P_i and a mapping of each communication $T_k \rightarrow T_l$ on a set of paths $P_i \rightsquigarrow P_j$. The throughput ρ of an allocation is the number of final task, which can be processed during a time unit. We define $T = 1/\rho$. Moreover, the following properties have to be respected:

1. any task T_k cannot be partially computed by a processor P_i,
2. any file $T_k \rightarrow T_l$ cannot be partially sent from a processor P_i to a processor P_j,
3. at least the final task is processed,
4. for any task T_l, if T_l is processed by processor P_j, then all files $T_k \rightarrow T_l$ must be sent to P_j,
5. any resource is busy for at most T time units,
6. for any task T_k, if T_k is processed by processor P_i, then any file $T_k \rightarrow T_l$ can be sent by P_i,
7. any task is processed by exactly one processor if we do not allow task duplication.

1.3 From an allocation to a complete schedule

2 Different linear programs, for allocations without duplication

2.1 Compact linear program, for a linear chain of tasks

This is the simplest case. We want to maximize the number of processed DAGs in a single time unit.

2.1.1 Notations

- y_i^k is the average number of tasks T_k processed by P_i in one time unit,
- $f^{kl}(P_i \rightarrow P_j)$ is the average number of files $T_k \rightarrow T_l$ sent by $P_i \rightarrow P_j$ in one time unit,

2.1.2 Linear program

We want to minimize T under the following constraints:

$$
\begin{array}{llcl}
\forall T_k & \sum_{P_i} y_i^k & = & 1 \\
\forall P_i & \sum_{T_k} y_i^k \times w_{i,k} & \leq & T \\
\forall T_k, \forall P_i & y_i^k & \geq & 0 \\
\forall P_i, & \sum_{P_j \to P_i} \sum_{T_k \to T_l} f^{kl}(P_j \to P_i) \times c_{i,j} \times \text{data}_{k,l} & \leq & T \\
\forall P_i, & \sum_{P_i \to P_j} \sum_{T_k \to T_l} f^{kl}(P_i \to P_j) \times c_{i,j} \times \text{data}_{k,l} & \leq & T \\
\forall T_k \to T_l, \forall P_i \to P_j & f^{kl}(P_i \to P_j) & \geq & 0 \\
\forall P_i, \forall T_k \to T_l, & \sum_{P_j \to P_i} f^{kl}(P_j \to P_i) - \sum_{P_i \to P_j} f^{kl}(P_i \to P_j) & = & y_i^l - y_i^k
\end{array} \qquad (1)
$$

2.2 Compact linear program, for multiple tasks

2.3 Extensive linear program, for multiple tasks

2.4 Compact linear program, for a single allocation and multiple tasks

2.5 Linear Program $n^o 365$, for a single allocation for multiple tasks, fixed routing

In this subsection, we assume that the path between each couple of processors is fixed before the execution of the algorithm. Figure 2 illustrates the fixed routing of the platform: any communication from P_2 to P_5 is sent through P_3.

We define two new notations:

- y_i^k is a binary variable, $y_i^k = 1$ iff T_k is processed on P_i
- $x_{ij}^{kl} \in \{0,1\}$, $x_{ij}^{kl} = 1$ iff $T_k \to T_l$ is mapped on the path $P_i \rightsquigarrow P_j$, which is unique by assumption.

We want to minimize T under the following constraints:

$$
\left\{
\begin{array}{llcl}
\forall T_k, \forall P_i & y_i^k & \in & \{0,1\} \\
\forall T_k \to T_l, \forall P_i \rightsquigarrow P_j & x_{ij}^{kl} & \in & \{0,1\} \\
\forall T_k & \sum_{P_i} y_i^k & = & 1 \\
\forall T_k \to T_l, \forall P_i \rightsquigarrow P_j & x_{ij}^{kl} & \leq & y_i^k \\
\forall T_l, \forall T_k \to T_l, \forall P_j & y_j^k + \sum_{P_i \rightsquigarrow P_j} x_{ij}^{kl} & \geq & y_j^l \\
\forall P_i, & \sum_{T_k} y_i^k w_{i,k} & \leq & T \\
\forall P_i \to P_j, & \sum_{P_q \rightsquigarrow P_r, P_i \to P_j \in P_q \rightsquigarrow P_r} \left(\sum_{T_k \to T_l} \left(x_{qr}^{kl} c_{i,j} \text{data}_{k,l} \right) \right) & \leq & T
\end{array}
\right. \qquad (2)
$$

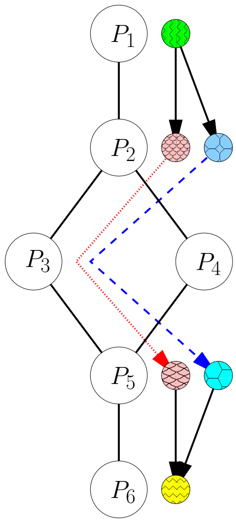

Figure 2: Example of fixed routing of Application A_1: both messages $T_2 \to T_4$ and $T_3 \to T_5$ are sent through P_3.

Proof. **all constraints are respected** We have to be sure that any solution returned by our linear program is valid, *i.e.* all properties given in Subsection 1.2.

- Property 1 is respected, since the y_i^k are binary variables (line 1).
- Property 2 is respected, since the x_{ij}^{kl} are binary variables (line 2),
- Property 3: all tasks are processed by exactly one processor (line 3), so the last one is processed by at least one.
- Property 4: let consider any task T_l and any of its input file $T_k \to T_l$. If T_l is processed by P_j, then we have $y_j^l = 1$ and the file is correctly sent to P_j by one of the P_is or processed by P_j itself (line 5).
- Property 5: lines 6 and 7 ensure that this constraint is respected.
- Property 6: let consider any file $T_k \to T_l$ sent by P_i to P_j. Then we have $x_{ij}^{kl} = 1$ and thus $y_i^k = 1$ (line 2): T_k is processed by P_i.
- Property 7: any task is processed by exactly one processor (line 1).

constraints are not too strong Let consider any valid allocation $\mathcal{A}$, which satisfies all properties given in Subsection 1.2. We show that $\mathcal{A}$ also satisfies the previous linear program.

1. The property 1 ensures that we can set y_i^k to 0 if P_i does not process T_k or to 1 if it does.

2. The property 2 ensures that we can set x_{ij}^{kl} to 1 if the file $T_k \to T_l$ is sent from P_i to P_j, or to 0 if it is not.
3. The property 7 ensures that each task is processed by exactly one processor. By definition of the y_i^ks, exactly one y_i^k is equal to 1 and the other ones are equal to 0. Thus, the third constraint is respected.
4. Let consider any task T_k, any file $T_k \to T_l$ and any processors P_i and P_j. From property 6, we know that if $T_k \to T_l$ is sent from P_i to P_j (thus, $x_{ij}^{kl} = 1$), then T_k is processed by P_i (thus, $y_i^k = 1$). In this case, the fourth constraint is respected. If there is no such communication, then we have x_{ij}^{kl} is equal to 0 and the constraint is always respected.
5. Let consider any task T_l, any file $T_k \to T_l$ and any processor P_j. From property 4, we know that if T_l is processed by P_j (thus, $y_j^l = 1$), then the file $T_k \to T_l$ is sent from at least one P_i to P_j (thus, $x_{ij}^{kl} = 1$) or T_k is processed by P_j (thus, $y_j^k = 1$). In this case, the fifth constraint is respected. If T_l is not processed by P_j, then we have $y_j^l = 0$ and the constraint is always respected.
6. Following property 5, all processors have a computation time smaller than T time units, and the computation time of processor P_i is equal to $\sum_{T_k} y_i^k w_i^k$. Then the sixth constraint is respected.
7. Following property 5, all communication links have a communication time smaller than T time units, and the communication time of link $P_i \to P_j$ is equal to $\sum_{T_k \to T_l} \left(x_{qr}^{kl} c_{i,j} \text{data}_{k,l} \right)$. Then the seventh constraint is respected.

$\square$

2.6 Linear Program $n^o 365$, for a single allocation for multiple tasks, flexible routing

In the previous subsection, any communication from a processor P_i to another processor P_j follows a fixed path $P_i \rightsquigarrow P_j$. If there is no external constraint on the path followed by such a communication, we could allow the linear program to choose the best path for each communication. By example, a file $T_k \to T_l$ and a file $T_{k'} \to T_{l'}$ sent by the same processor P_i to the same processor P_j could follow different paths, as presented in Figure 3. This subsection addresses this new problem. Moreover, we could allow a given file $T_k \to T_l$ sent from P_i to P_j to be sent in several parts following different paths, like IP packets over the internet, as presented in Figure 4.

Before writing the complete linear program solving this problem, we explain the new variables we introduce here.

Let consider any file $T_k \to T_l$. We look for a function $f^{kl} : E_P \mapsto \mathbb{R}$, such that $f^{kl}(P_i \to P_j)$ is equal to the fraction of the file $T_k \to T_l$ sent through the link $P_i \to P_j$.

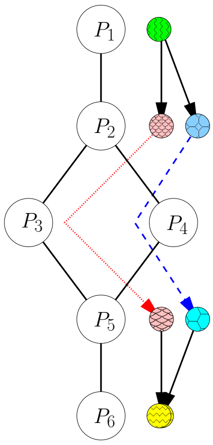

Figure 3: Messages $T_2 \rightarrow T_4$ and $T_3 \rightarrow T_5$ of Application A_1 are sent from P_2 to P_5 through different links.

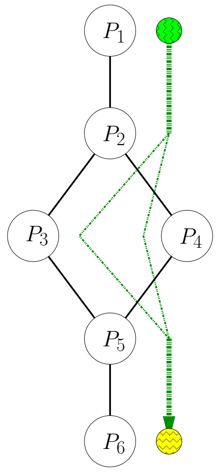

Figure 4: The file $T_1 \rightarrow T_2$ of Application A_2 is split in two parts sent from P_2 to P_5 through different links.

Let consider the same file $T_k \rightarrow T_l$ and any processor P_q. The total fraction of this file received by P_q from its neighbours is equal to $\sum_{P_r \rightarrow P_q} f^{kl}(P_r \rightarrow P_q)$. The total fraction of this file sent by P_q to its neighbours is equal to $\sum_{P_q \rightarrow P_r} f^{kl}(P_q \rightarrow P_r)$. If we use the x_{ij}^{kl}s, we have two more relations:

- By definition of the x_{ij}^{kl}s, $\sum_{P_s} x_{sq}^{kl}$ copies of the file $T_k \rightarrow T_l$ are sent from the whole set of processors to P_q and will not be forwarded by it since P_q will process T_l. In fact, we do not forbid a file to be sent to P_q while P_q do not process it, even if this communication is completely useless. Similarly, we do not forbid P_q to receive multiple copies of the same file. However, we consider that P_q consumes exactly $\sum_{P_s} x_{sq}^{kl}$ copies of $T_k \rightarrow T_l$.

- $\sum_{P_t} x_{qt}^{kl}$ copies of $T_k \rightarrow T_l$ are sent by P_q to the whole set of other processors. In this case, we recall that P_q needs to process T_k to be able to do such communications.

Using these relations, we can write the following equation:

$$\sum_{P_r \rightarrow P_q} f^{kl}(P_r \rightarrow P_q) - \sum_{P_q \rightarrow P_r} f^{kl}(P_q \rightarrow P_r) = \sum_{P_s} x_{sq}^{kl} - \sum_{P_t} x_{qt}^{kl} \qquad (3)$$

Figure 5 illustrates this idea for the file $T_2 \rightarrow T_4$.

Then the complete linear program we want to solve to obtain a complete and valid allocation is:

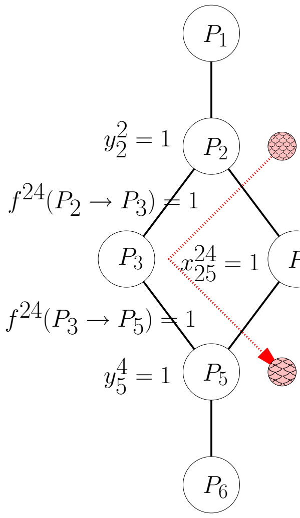

Figure 5: Flexible routing for file $T_2 \to T_4$ sent from P_2 to P_5; only positive values are displayed.

$$\begin{cases}
\forall T_k, \forall P_i & y_i^k & \in & \{0,1\} \\
\forall T_k \to T_l, \forall P_i, \forall P_j & x_{ij}^{kl} & \in & \{0,1\} \\
\forall T_k & \sum_{P_i} y_i^k & = & 1 \\
\forall T_k \to T_l, \forall P_i, \forall P_j & x_{ij}^{kl} & \leq & y_i^k \\
\forall T_k \to T_l, \forall P_j & y_j^k + \sum_{P_i} x_{ij}^{kl} & \geq & y_j^l \\
\forall P_i, & \sum_{T_k} y_i^k w_{i,k} & \leq & T \\
\forall P_q, \forall T_k \to T_l & \sum_{P_r \to P_q} f^{kl}(P_r \to P_q) - \sum_{P_q \to P_r} f^{kl}(P_q \to P_r) & = & \sum_{P_s} x_{sq}^{kl} - \sum_{P_t} x_{qt}^{kl} \\
\forall P_i \to P_j, & \sum_{T_k \to T_l} \left(f^{kl}(P_i \to P_j) c_{i,j} \mathrm{data}_{k,l} \right) & \leq & T \\
\forall P_i \to P_j, \forall T_k \to T_l, & f^{kl}(P_i \to P_j) & \geq & 0
\end{cases} \tag{4}$$

In this linear program, the f^{kl} functions have their values in $\mathbb{R}$. Thus, any file can be split in several packets following different paths, like IP packets over the internet. We can force all packets to follow the same route, by forcing $f^{kl}(P_i \to P_j)$ to be in $\mathbb{Z}$ for any file $T_k \to T_l$ and for any link $P_i \to P_j$. If we use real number for the f^{kl}s instead of integer ones, another point of view is to consider that we have several allocations with the same task-to-processors mapping, but with different communication schemes. If we use again the example shown in Figure 4, we can consider than one file over two is sent through P_3 and the other is sent through P_4.

3 Looking for a small set of allocations: several heuristics

4 A single linear program, for allocations allowing duplication

In this section, we want to remove another constraint to the initial problem: any intermediate task hacwsve to be processed by exactly one processor. Thus, we look for a unique allocation allowing any task to be processed more than once. The duplication of a task on two processors can be interesting by avoiding some costly communications.

4.1 Linear Program $n^o 365$, for a single allocation for multiple tasks, fixed routing

We keep the notations y_i^k and y_{ij}^{kl} defined in Subsection 2.5. As in Section 2, we begin by searching an allocation, such that communications respect given communication paths between couples of processors.

We want to minimize T under the following constraints:

$$\left\{\begin{array}{llrcl}
\forall T_k, \forall P_i & & y_i^k & \in & \{0,1\} \\
\forall T_k \to T_l, \forall P_i \rightsquigarrow P_j & & x_{ij}^{kl} & \in & \{0,1\} \\
 & & \sum_{P_i} y_i^n & \geq & 1 \\
\forall T_k \to T_l, \forall P_i \rightsquigarrow P_j & & x_{ij}^{kl} & \leq & y_i^k \\
\forall T_l, \forall T_k \to T_l, \forall P_j & & \sum_{P_i \rightsquigarrow P_j} x_{ij}^{kl} & \geq & y_j^l \\
\forall P_i, & & \sum_{T_k} y_i^k w_{i,k} & \leq & T \\
\forall P_i \to P_j, & \multicolumn{2}{r}{\sum_{P_q \rightsquigarrow P_r, P_i \to P_j \in P_q \rightsquigarrow P_r} \left(\sum_{T_k \to T_l} \left(x_{qr}^{kl} c_{i,j} \mathrm{data}_{k,l} \right) \right)} & \leq & T
\end{array}\right. \tag{5}$$

Note that we allow any processor to send multiple copies of the same file. However, only the processor computing a given task can do the duplication of a file. Intermediate routers cannot duplicate them. Moreover, a given processor P_i can send to another processor P_j a given file $T_k \to T_l$ at most once (this constrainst comes from line 4). However, there is no interest in sending the same file twice to the same processor.

Proof. **all constraints are respected** Now, we show that all properties given in Subsection 1.2 are respected by a solution of our linear program.

- Property 1 is respected, since the y_i^k are binary variables (line 1).
- Property 2 is respected, since the x_{ij}^{kl} are binary variables (line 2).
- Property 3: since y_i^k are binary variables, line 3 expresses that the final task is processed by at least one processor,

- Property 4: we consider any task T_l and any of its input file $T_k \to T_l$. If T_l is processed by P_j, then we have $y_j^l = 1$ and the file is correctly sent to P_j by one of the P_is (line 5).
- Property 5: lines 6 and 7 ensure that this constraint is respected.
- Property 6: let consider any file $T_k \to T_l$ sent by P_i to P_j. Then we have $x_{ij}^{kl} = 1$ and thus $y_i^k = 1$ (line 2): T_k is processed by P_i.
- Property 7: in this section, this property is not required to consider an allocation as valid.

constraints are not too strong We can prove that the constraints given in the linear program above are not too strong in the same way as in 2.5, the only difference being the bound on the number of processor executing any task.

□

4.2 Linear Program $n^o 365$, for a single allocation for multiple tasks, flexible routing

$$\left\{\begin{array}{llcl}
\forall T_k, \forall P_i & y_i^k & \in & \{0,1\} \\
\forall T_k \to T_l, \forall P_i, \forall P_j & x_{ij}^{kl} & \in & \{0,1\} \\
 & \sum_{P_i} y_i^n & \geq & 1 \\
\forall T_k \to T_l, \forall P_i, \forall P_j & x_{ij}^{kl} & \leq & y_i^k \\
\forall T_k \to T_l, \forall P_j & y_j^k + \sum_{P_i} x_{ij}^{kl} & \geq & y_j^l \\
\forall P_i, & \sum_{T_k} y_i^k w_{i,k} & \leq & T \\
\forall P_q, \forall T_k \to T_l & \sum_{P_r \to P_q} f^{kl}(P_r \to P_q) - \sum_{P_q \to P_r} f^{kl}(P_q \to P_r) & = & \sum_{P_s} x_{sq}^{kl} - \sum_{P_t} x_{qt}^{kl} \\
\forall P_i \to P_j, & \sum_{T_k \to T_l} \left(f^{kl}(P_i \to P_j) c_{i,j} \mathrm{data}_{k,l}\right) & \leq & T \\
\forall P_i \to P_j, \forall T_k \to T_l, & f^{kl}(P_i \to P_j) & \geq & 0
\end{array}\right. \tag{6}$$

Note: in this model, if P_i computes the task T_k and has to send the resulting files to both P_j and $P_{j'}$ via a common router P_r, P_i sends the file twice, P_r receives it twice (although they are the same data in it) and sens them to P_j and $P_{j'}$. Of course, it should be more clever if P_i could send only one example of the file to P_r, P_r duplicating the file to P_j and $P_{j'}$.

To allow duplication of tasks among processors, we had to use integer variables instead of rational ones, even if mixed linear program are by far more difficult to solve. However, we could still use rational variables to find the best communication scheme. If we want to introduce the duplication of files by intermediate routers, we have to use integer variables for communications.

Below stands the complete linear program, allowing duplication of files by intermediate routers. For each file $T_k \to T_l$, we introduce a new function $g^{kl} : V_P \mapsto \mathbb{Z}$, which indicates whether the processor P_i has the file $T_k \to T_l$ (it can have received it, or producted it).

$$
\left\{
\begin{array}{lrcl}
\forall T_k, \forall P_i & y_i^k & \in & \{0,1\} \\
\forall T_k \to T_l, \forall P_i, \forall P_j & x_{ij}^{kl} & \in & \{0,1\} \\
 & \sum_{P_i} y_i^n & \geq & 1 \\
\forall T_k \to T_l, \forall P_i, \forall P_j & x_{ij}^{kl} & \leq & y_i^k \\
\forall T_l, \forall T_k \to T_l, \forall P_j & y_j^k + \sum_{P_i} x_{ij}^{kl} & \geq & y_j^l \\
\forall P_i, & \sum_{T_k} y_i^k w_{i,k} & \leq & T \\
\forall P_i \to P_j, \forall T_k \to T_l, & \sum_{T_k \to T_l} \left(f^{kl}(P_i \to P_j) c_{i,j} \mathrm{data}_{k,l} \right) & \leq & T \\
\forall P_i \to P_j, \forall T_k \to T_l, & f^{kl}(P_i \to P_j) & \geq & 0 \\
\forall P_i \to P_j, \forall T_k \to T_l, & g^{kl}(P_i) & \geq & f^{kl}(P_i \to P_j) \\
\forall P_i, \forall T_k \to T_l, & g^{kl}(P_i) & \geq & 0 \\
\forall P_j, \forall T_k \to T_l, & \sum_{P_i \to P_j} f^{kl}(P_i \to P_j) + y_j^k & \geq & g^{kl}(P_j)
\end{array}
\right.
\tag{7}
$$