Scheduling algorithm to avoid contention in meshed networks

Dominique Barth[1], **Maël Guiraud**[1,2], **Brice Leclerc**[2], **Olivier Marcé**[2], and **Yann Strozecki**[1]

[1]David Laboratory, UVSQ
[2]Nokia Bell Labs France

May 29, 2020

Introduction

TODO: doner le contexte CRAN, puis l'objectif d'optimisation de latence. Dire qu'on a déjà fait désynchronysé mais que dans les archtectures actuelles il faut plutot faire synchronisé. Comparer à nos travaux antérieurs et à d'autres qu'on a déjà cité (voir les papiers précédents, surtout les citations du dernier)

1 Modeling Periodic Scheduling of datagrams over a Network

Let $[n]$ denote the interval of n integers $\{0, \ldots, n-1\}$.

1.1 Modeling the Networks and its Contention Points

We study a communication network with pairs of source-destination nodes between which messages are sent periodically. The routing between each pair of such nodes is given. The network is represented by a directed acyclic multigraph $G = (V, A)$. The set of vertices is composed of three disjoint subsets: $\mathcal{S}$ the set of sources of the messages, $\mathcal{D}$ the set of destination of the messages, and $\mathcal{C}$ the set of contention points in the network. Indeed, some links of the network are shared between several pairs of source-destination nodes. A contention point represent the beginning of a shared link, i.e. the physical node of the network which sends the messages into the link. An arc in G can represent several physical links or nodes, which do not induce contention points. Each arc (u, v) in A is labeled by an integer weight $\omega(u, v)$ which represents the number of tics elapsed between the sending time of the message in u and the reception time of this message in v using this arc.

A **route** r in G is a directed path, that is, a sequence of adjacent vertices $u_1, \dots, u_l$, with $(u_i, u_{i+1}) \in A$. The **weight of a vertex** u_i in a route $r = (u_0, \dots, u_l)$ is defined by $\lambda(u_i, r) = \sum\limits_{1 \leq j < i} \omega(u_j, u_{j+1})$. It is the number of tics needed by a message to go from the first vertex of the route to u_i. The **length** of the route r is defined by $\lambda(r) = \lambda(u_l, r)$. We denote by $\mathcal{R}$ a set of routes of the graph G, the pair $(G, \mathcal{R})$ is called a **routed network** and represents our telecommunication network.
Let $r \in \mathcal{R}$, with $r = (u_0, u_1, \dots, u_l)$, then we say that u_i is of **contention level** i for the route r, and we denote it by $cl(u_i, r) = i$. The contention level of a node u is the maximum of its contention level over all routes going through itself: $cl(u) = \max\limits_{r \in \mathcal{R} \text{ and } u \in r} cl(u, r)$.

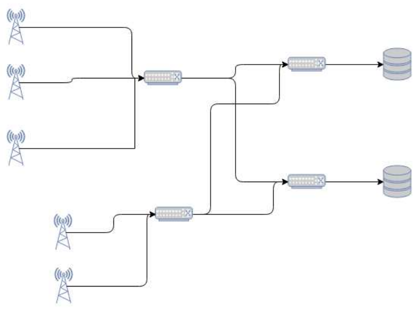

Figure 1: A C-RAN fronthaul network

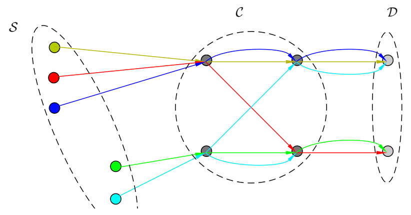

Figure 2: The corresponding routed network

The **contention depth** of a routed network $(G, \mathcal{R})$ is equal to the maximum of the contention level over all vertices. It is the number of contention points on the longest route of the network. In all the article, $(G, \mathcal{R})$ is the routed network, the number of routes, $|\mathcal{R}|$, is denoted by n and the contention depth is denoted by d. The set $\mathcal{C}$ denotes the set of contention vertices.

1.2 Datagram Transmission over the Network

In this article, we consider a discretized time. The unit of time is called a **tic**. This is the time needed to send an atomic data in a link of the network. We assume that the speed of the links is the same over all the network. In practice, the size of an atomic data is usually 64B, the speed of the links is considered to be 10Gbps, and the length of a tic is thus TODO: rajouter durée d'un tic.

In the process we study, a message called a **datagram** is sent on each route from the source node. The size of a datagram is an integer, denoted by τ, it is the number of tics needed by a node to emit the full datagram through a link. In this paper, we assume that τ is the same for all routes. It is justified by our application to C-RAN, where all source nodes are RRHs sending the same type of message. Once a datagram has been emitted, it cannot be fragmented during its travel in the network. We consider that the first datagram sent of each route is ready to be sent at time 0.

Let $r = (u_0, \dots, u_l)$ be a route. In order to avoid contention, it is possible to buffer datagrams in contention points. The function A, called an **assignment**, associates an integer value, greater or equal to 0, to each couple (route,vertex) of the routed network $(G, \mathcal{R})$. Those integers represent the buffering time of the datagrams in the nodes of the network: the time a datagram on the route r waits in the buffer of a vertex u_i is $A(r, u_i)$.

The **arrival time** of a datagram in vertex u_i of r, is the first time at which the datagram sent on r reaches u_i, and is defined by $t(r, u_i) = \lambda(u_i, r) + \sum_{k=0}^{i-1} A(r, u_k)$. The date at which a datagram reaches a vertex u_i is decomposed into a *physical delay* due to the time to go through the links before u_i and a *logical* delay caused by the use of buffers as determined by assignment A. The **sending time** of a datagram in vertex u_i in r, is the first time at which the datagram is sent by u_i. It is defined by $s(r, u_i) = t(r, u_i) + A(r, u_i)$. This is the arrival time of the datagram plus the buffer time fixed by A.

If u_l is the last vertex of the route r, the transmission time of the datagram on r is denoted by $TR(A, r)$ and is equal to $t(r, u_l)$. We define the **full transmission time** of an assignment A as $TR(A) = \max\limits_{r \in \mathcal{R}} TR(A, r)$. This is the time elapsed before the reception of the beginning of the last datagram.

1.3 C-RAN Network Modeling

TODO: completement ré ecrire cette section en disant qu'on peut ou non décomposer la BBU, et que le graph est symetrique car c'est full duplex mais que ce n'est pas obligé. Cette section doit peut être venir juste après network modeling

1.4 Periodic Datagrams

The process we modelize in this article is **periodic**: during each period of P tics, a datagram is ready to be sent from each source node in the network at time zero. The process is assumed to be infinite, that is it should work for an arbitrary number of periods. We chose, because it is simplier to implement in real networks and also more tractable from a theoritical perspective, to always use the same buffering in all periods. In other words, at the same time of two different periods, all messages are at the same position in the network: the assignments we build are themselves periodic of period P. Thus, we only need to consider the behavior of the datagrams on each node of the network during a single period, and to apply the same pattern to every subsequent period. Let us call $[r, u]_{P,\tau}$ the set of tics used by a datagram on r at vertex u in a period P, that is $[r, u]_{P,\tau} = \{s(r, u) + i \mod P \mid 0 \leq i < \tau\}$.

Let us consider two routes numbered r_1 and r_2, they have a **collision** at the contention point u iff $[r_1, u]_{P,\tau} \cap [r_2, u]_{P,\tau} \neq \emptyset$.

An assignment A of a routed network $(G, \mathcal{R})$ is said to be **valid** if *no pair of routes has a collision*. The validity of an assignment depends on the period P and the size of the messages τ, but most of the time these values are clear from the context and we do not recall them (contrarily to previous work [1]). Note that the period P, as well as the size of a message τ is fixed in our $C-RAN$ application, but not the buffering policy. Hence, the aim of our work is to find a valid assignment which minimizes the latency of transmissions over the network, that is $TR(A)$.

Synchronized Periodic Assignment for Low Latency (SPALL)
Input: A symmetric routed network $(G, \mathcal{R})$, period P, datagram size τ.
Question: find A which minimizes $TR(A)$.

1.5 Networks of Contention Depth 1

C'est exactement le problème BRA du papier précédent (enfin chaque composante connexe du graphe). Si en plus on met 0 comme poids à tous les arcs, on doit résoudre avec longest shortest.

1.6 Networks of Contention Depth 2 and more

Parler d'abord de l'étoile et de la propriété de routage cohérent. Dire que ça n'est pas le cas général d'un contention depth 2. Introduire les réseaux plus généraux de contention depth 3. Si on a que deux sommets, ça correspond à l'étoile et donc au cas étudié dans nos deux autres papiers à citer ici (mais en version synchronisée).

TODO: np completude, meme sur l'etoile, parler du papier avec PALL

2 Compact Representation of an Assignment

We define $\prec$, the pointwise order on assignments: $A_1 \preceq A_2$ if for all $r \in \mathcal{R}$, $TR(A_1, r) \leq TR(A_2, r)$. Moreover, we say that $A_1 \prec A_2$ if $A_1 \preceq A_2$ and there is an $r \in \mathcal{R}$ such that $TR(A_1, r) < TR(A_2, r)$. Remark that assignments which minimize $TR(A)$ are also minimal for $\prec$. Hence, it is enough to consider minimal assignments (for $\prec$) to solve SPALL.

We explain in this section how to represent most assignment in a compact way, forgetting about the precise buffering time by only considering informations about the order of the datagrams in each contention point. All miniminal assignments have a compact representation, which implies that we do not need to consider assignment without a compact representation when solving SPALL. It allows to design an FPT algorithm for SPALL by going through all compact representations, but also to design good polynomial time heuristics using taboo search or simulated annealing, since one can easily define the neighborood of a compact representation.

Let us denote by $\mathcal{R}_u$ the subset of routes of $\mathcal{R}$ which contains u.

Definition 1 (Compact assignment)**.** *Let $(G, \mathcal{R})$ be a routed network. A compact assignment CA is, for each contention point u in G, a pair (O_u, S_u) where O_u is an order on R_u and S_u is a subset of R_u.*

2.1 From a valid assignment to its compact representation

We first define a method to obtain a compact assignment from some valid assignment A, that we call the compact representation of A and that we denote by $CR(A)$. Assume all routes are indexed by an integer in $[n]$ and let u be a contention point of G. We also assume that for all vertices u, there is a route $r \in \mathcal{R}_u$ such that $A(r_i, u) = 0$.

Let r_0 be route of smallest index such that $A(r_0, u) = 0$. The datagram of r_0 arrives and goes to the next contention point at time $t(r_0, u)$. Let us define the *normalized arrival time* of r at u: for all $r \in \mathcal{R}_u$, $nt(r_0, r, u) = (t(r, u) - t(r_0, u)) \mod P$. It is the time at which the datagram of r arrives at u, in a period normalized so that the datagram of r_0 goes through u at time 0. Similarly, we define the *normalized sending time* as $ns(r_0, r, u) = (s(r, u) - t(r_0, u)) \mod P$.

We define O_u as the order on the routes of R_u induced by the values $ns(r, u)$. The set S_u is defined as the set of routes going through u such that $ns(r_0, r, u) < nt(r_0, r, u)$. Imagine the time is cut into periods $[t(r_0, u) + iP, t(r_0, u) + iP[$, with $i \in \mathbb{N}$. Intuitively, S_u represents the set of routes with a datagram going through u in the period *after* the one they have been available in.

Figure 4 illustrate how a compact representation is computed from an assignment on a single node u. On the top, the datagrams are represented by sending time $s(r_i, u)$ while the bottom shows the datagrams in a single period, normalized by arrival time $ns(r_0, r_i, u)$.

Note that for $CR(A)$ to be defined, we need that on each contention point, a datagram is not buffered. We call such an assignment a **canonical** assignment. It turns out that any assignment A can be made canonical without increasing $TR(A)$.

Lemma 1. *Let A be a valid non canonical assignment, then there is a valid canonical assignment A' such that $A' \preceq A$.*

Proof. Consider a vertex u of contention level 1, such that for all $r \in \mathcal{R}_u$, $A(r, u) > 0$. Let us define m as the minimimum of these values, we define $A'(r, u) = A(r, u) - m$. There are no collision on u, since all departure times have been translated by the same value. Moreover, if v is the vertex after u

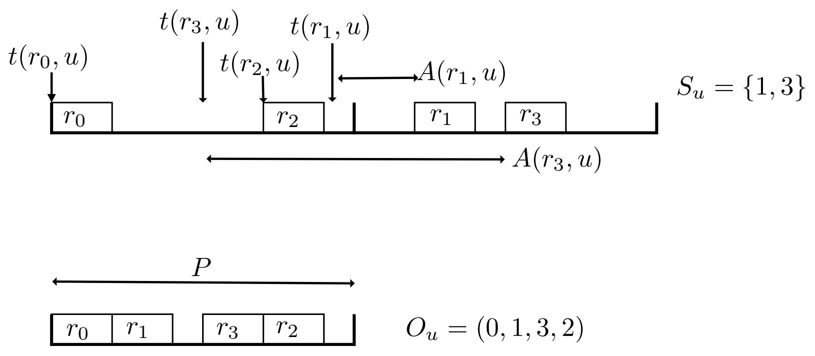

Figure 3: A compact representation of an assignment in which $O_u = (0,1,3,2)$ and $S_u = \{1,3\}$

in a route r, we define $A'(r,v) = A(r,v) + m$. Hence, all departure times for vertices of contention levels larger than one are the same in A and A', which implies that there are no collisions in these vertices. We have proven that A' is still valid. Since all departure times of A' are less or equal to those induced by A, we have $A' \preceq A$. Moreover, if r_0 is the route with $A(r_0,u) = m$, then $A'(r_0,u) = 0$.

We apply this transformation by increasing contention level. Since, the transformation applied at some contention level do not change A' for smaller contention levels, it proves that A' is valid, canonical and that $A' \preceq A$. □

2.2 From a compact assignment to it realization

We now explain how to transform a compact representation into a canonical assignment. Moreover, we will show that the obtained assignment is the smallest among all assignments of same representation. We first explain how to do the transformation on a routed network with a single contention point u.

Recall that the datagram of a route r is available at time $t(r,u)$ in the vertex u. Let us consider a compact assignment CA, which associates the pair (O_u, S_u) to u. We now define inductively how to build an assignment $Real(CA)$ from CA, that we call the realization of CA. Note that we can fail to build a realization from a compact representation, then $Real(CA)$ is undefined and we say that CA is not realizable. To lighten the notation in the next aragraphs, we will denote $Real(CA)$ by A.

TODO: expliquer la construction avant de la faire, dans lordre croissant en terme de valeur Let say that the order O_u is $(r_1,\dots,r_l)$. We fix $A(r_1,u)$ to zero, that is the first datagram in the period has no buffering time. Then, in each period beginning by the first datagram, the datagrams will be in order O_u. Since the first datagram of the period is fixed, we use it to consider normalized arrival times and normalized sending times. Assume that $A(r_i,u)$ have been set for $i \leq l$, we explain how to set $A(r_{i+1},u)$. If $r_{i+1} \notin S_u$, then we fix $A(r_{i+1}, u)$ so that $ns(r_{i+1}, u)$ is the maximum of $ns(r_1, r_i, u) + \tau$ and $nt(r_1, r_{i+1}, u)$. If $ns(r_1, r_{i+1}, u) > P - \tau$, then CA is not realizable. If $r_{i+1} \in S_u$, then we fix $A(r_{i+1}, u)$ so that $ns(r_1, r_{i+1}, u) = ns(r_1, r_i, u) + \tau$. In both cases, if $ns(r_1, r_{i+1}, u) \geq nt(r_1, r_{i+1}, u)$, then CA is not realizable (the sending time is in the wrong period with regard to S_u).

The function *Real* can easily be generalized to any routed network. Indeed, one can first consider all vertices of contention level 1, the routes going through them form disjoint sets. Hence, we can define *Real* independently on each vertex of contention level 1. Then using the buffering computed for this vertices, one can compute the arrival time of each route in vertices of contention level 2 and compute *Real* for these vertices in the exact same way, and so on for all contention levels. In the following lemmas and theorems, we always consider a single contention vertex, since it is trivial to extend any property for one contention vertex to the whole routed network as we just explained.

Lemma 2. *The funtion $Real(CA)$ can be computed in time $O(nd)$, where d is the contention depth of the network. If CA is realizable, then $Real(CA)$ is a valid canonical assignment.*

Proof. In the inductive construction of $Real(CA)$, only a constant number of comparisons and additions are needed to compute the the buffer time of a route from the previous one. Hence, the time spent in a vertex u is linear in $|\mathcal{R}_u|$. A route can go through only one vertex of a given contention level, hence the time spent computing buffers for all vertices of a contention level is in $O(n)$ and for the whole graph it is in $O(nd)$.

To prove that there are no collision between pair of routes for a given assignment, it is enough to prove it for any interval of time of size P. Hence, it is enough to consider the normalized sending time and to verify they do not induce a collision. By construction, $ns(r_1, r_{i+1}, u)$ is always larger than $ns(r_1, r_i, u) + \tau$ and less than $P - \tau$, which proves the absence of collision. Finally, $Real(CA)$ is canonical, since by definition $Real(CA)(r_1, u) = 0$, where r_1 is the first route in O_u. □

We can define the following equivalence relation over canonical assignments: A and B are equivalent if and only if $CR(A) = CR(B)$. We say that a compact assignment $CA = (O_u, S_u)_{u \in V(G)}$ is *canonical* if it is a realizable compact assignment, $CR(Real(CA)) = (O'_u, S'_u)_{u \in V(G)}$ and if for all vertices u, the first route of O_u and O'_u coincide. This notion of canonicity is useful because the function CR always sends a canonical assignment on a canonical compact assignment. It is just restrictive enough (by fixing the first element in each order), that the function CR is the inverse of *Real* over canonical compact assignments. It implies that $Real(CA)$ can be chosen as the representative of the equivalence class of the assignments having CA as a representation.

In fact, as implied by the following Lemma, we can be more precise on $Real(CA)$: it is minimal in its equivalence class for $\prec$.

Lemma 3. *Let A be a valid assignment, then $Real(CR(A)) \preceq A$.*

Proof. Given a vertex u and a route $r \in \mathcal{R}_u$, we prove by induction that $\mathit{Real}(CR(A))(r,u) \leq A(r,u)$. Let (O_u, S_u) be the pair associated to u by $CR(A)$, with $O_u = (r_1, \ldots, r_l)$. By definition of CR, r_1 the first route in O_u, is such that $A(r_1, u) = 0$. By definition of Real, we have that $\mathit{Real}(CR(A))(r_1, u) = 0 = A(r_1, u)$. Now assume that $\mathit{Real}(CR(A))(r_i, u) \leq A(r_i, u)$ for some i.

First, consider the case $r_{i+1} \notin S_u$. By definition of CR, $ns(r_1, r_{i+1}, u)$ must be larger than $ns(r_1, r_i, u) + \tau$ and because $r_{i+1} \notin S_u$ it must also be larger than $rs(r_1, r_{i+1}, u)$. Since $\mathit{Real}(CR(A))(r_{i+1}, u)$ is the minimum value so that both constraints are true for $\mathit{Real}(CR(A))$, using the induction hypothesis, we have $\mathit{Real}(CR(A))(r_i, u) \leq A(r_i, u)$. The case $r_{i+1} \in S_u$ is similar and left to the reader. □

$$O_u = (2, 1, 0, 3)$$
$$S_u = \{1\}$$

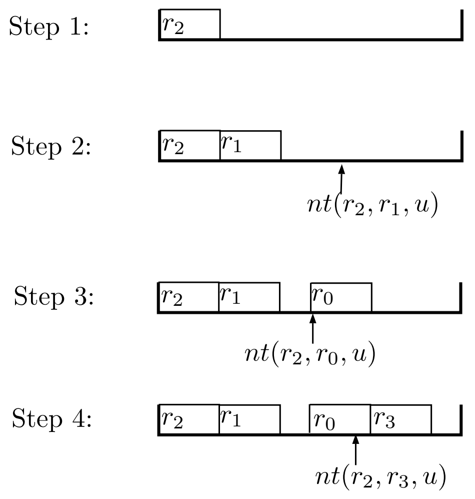

Figure 4: A canonical assignment built from a compact assignment.

A compact representation of a solution for an instance of depth larger than k is a list of compact representations, one for each contention arc. The following theorem explains why it is enough to explore the compact representations to solve SPALL.

To solve SPALL, we need to find an assignment for which $TR(A)$ is minimal. Using the previous Lemmas, it is easy to see that it can be done by a buteforce enumeration of the compact representations,

To do so, because of Lemma 3, it is enough to enumerate all canonical compact representation, to compute their realization and the corresponding transmission time.

Theorem 4. *For routed networks of fixed contention depth d, the problem* SPALL *parametrized by n the number of routes is FPT: it can be solved in time $O(nd(n!2^n)^d)$.*

Proof. The algorithm to solve SPALL is the following: all compact assignments CA are generated, for each of them $TR(Real(CA))$ is computed in time $O(nd)$ by Lemma 2 and we keep the compact assignment for which this value is minimal. Because of Lemma 3, to compute the minimum of $TR(A)$, it is enough to compute the minimum of $TR(Real(CA))$.

Now, we need to evaluate the number of compact assignments. On a single contention point u with $s = |\mathcal{R}_u|$ routes going through, there are $s!2^s$ possible restrictions of a compact assignment by counting the number of pairs of set and order over $\mathcal{R}_u$. On a given contention level consisting in the vertices $\{u_1, \dots, u_l\}$, with $s_i = |\mathcal{R}_{u_i}|$, there are $\prod_{1 \leq i \leq l} s_i!2^{s_i}$. On a given contention level, all routes use at most 1 vertex, hence $\sum_{1 \leq i \leq l} s_i \leq n$. By convexity of the exponential and factorial functions, we have $\prod_{1 \leq i \leq l} s_i!2^{s_i} \leq n!2^n$ and since there are d contention levels, we have at most $(n!2^n)^d$ compact assignments which proves the theorem. □

3 Algorithms to solve SPALL

3.1 Greedy Algorithm

To initialize our loacal search heuristics, a compact assignment is needed. In order to obtain a realisable comapact assignment, we take the compact representation of a valid assignment given by the greedy algorithm described here.

The contention vertices are sorted by contention level. Several contention vertices of the same contention level are managed independantly. For a vertex u, we do the following routine:

1. The budget $2 \times \lambda(u, r) - t(r, u)$ is computed for each routes. Note that this value can be negative since a datagram can be buffered in contention vertices before u.

2. The datagram with the lowest $t(r, u)$ is sent without buffers.

3. If one or several datagrams are available at the end of the emmision of the last datagrams.

   - Chose the datagram with the lowest budget, compute the delay and send it

   - Go back to step 3.

4. If no datagrams are available, send the one with the lowest $t(r, u)$ which have not been sent yet.

5. Go back to step 3.

3.2 Local Search Heuristics

The number of compact representations grows extremely quickly with n. Hence, to find one which minimizes $TR(A)$, we propose several classical local search algorithm: hill climbing, tabu search and simulated annealing. This methods works as long as a relevant notion of neighborood of a solution is proposed. The neighborood relation must satisfy several properties: it must be quick to compute (hence not to large) and the implicit graph of solutions defined by the neighborood relation should be connected. We now propose a simple neighborood relation over compact assignments.

Let u be a contention vertex of a network, and let CA be a compact assignment for this network, which associates the pair (O_u, S_u) to u. Let $r \in \mathcal{R}_u$, the *r-neighborood* of (O_u, S_u) is the set of pairs (O, S) such that:

1. $O_u = O$ or $O_u = (r_1, \ldots, r_l)$ with $r_i = r$ and $O = (r_1, \ldots, r_{i-2}, r_i, r_{i-1}, \ldots, r_l)$
2. $S_u = S \cup \{r\}$ or $S_u = S \setminus \{r\}$

Informally, a compact representation is in the r-neighborood of another one if it can be obtained by moving down r once (or not changing it) in the order and adding or removing r from the set. Remark that the r-neighborood of any pair (O_u, S_u) has at most 4 members (it can be 2 when the route r is in first position and cannot be exchanged with the previous one).

The *r-neighborood* of a compact assignment CA is the set of all compact assignments $CA' = (O'_u, S'_u)_{u \in V(G)}$, such that (O'_u, S'_u) is in the r-neighborood of (O_u, S_u). Finally, the *neighborood* of an assignment CA is the union for all $r \in \mathcal{R}$ of the r-neighboroods of CA.

Let us denote by $k_1, \ldots, k_n$ the number of contention vertices on the n routes of a routed network $(G, \mathcal{R})$. Then, a compact representation has at most $\sum_{i=1}^{n} 4^{k_i}$ neighbors. Since we always work with networks of bounded contention depth (2 or 3 in practice), the size of a neighborhood is linear in the number of routes. Remark that, with this definition, we allow to include unrealizable compact assignment in a neighborood. We can easily forbid them and we will, since it is much more practical for several of our algorithms. TODO: Expliquer que le graphe des solutions réalisable est connexe (il suffit d'utiliser des datagrams en position 2, les bouger ne peut pas rendre la soluce non réalisable)

3.3 Branch and Bound

We define a **partial compact representation** Pcr of a routed network (G, R) as a choice of pairs (O, S) on a subset of the arcs of the graph. The function $Bound(Pcr)$ gives a lower bound of $TR(G, \mathcal{R})$, considering Pcr.

As explained on theroem **??** there is a compact representation which minimizes $TR(G, \mathcal{R})$. We browse the entire set of compact representation to find the one which minimizes $TR(G, \mathcal{R})$.
To do so, we design the following branch and bound algorithm. First, we determine an upper bound for our branch and bound algorithm by computing

$TR(G,\mathcal{R})$ with a greed algorithm. Then, for all arcs (sorted by contention level), we enumerate all the possible compact representations. For each of those compact representation, we compute $Bound(Pcr)$. If the result given by this function is higher than the upper bound already found, we do no consider this compact representation. Otherwise, we go to the next arc. Once we have chose an pair (O,S) for all arcs, we compute $Sol(O,S)$ for all arcs, and if the given $TR(G,\mathcal{R})$ is lower than the previous upper bound we update this value for the rest of the search.

In order to speed up the search, we made the following cuts:
définition des différentes coupes.

4 Results

4.1 Random generation of routed network

Dans le cas d'application C-RAN, on prends un nombre de routes limitées ($<$20) et des tailles de routes petites. Néanmoins, on regarde des instances pour lesquelles le problème est difficile.

Définition du type de graph regardés, expliquer que les instances sont plus dures quand les routes sont de tailles similaires et quand on augmente le nombre de points de contentions.

4.2 Neighborood heuristics

Étudier empiriquement les différentes variantes, paramètres, solution de départ, cout en temps ...

Taboo : Avec une hash table pour la mémoire, ce qui rend long la recherche quand on fait beaucoup de pas

Recuit, expliquer comment j'ai réglé la température etc

4.3 Branch and bound

Montrer avec des courbes (nombre de représentation générées et pourcentage de minimales en fonction du nombre de routes) que ces coupes sont très efficace, elles permettent de générer uniquement des représentations compactes canoniques, et presque aucunes dont la réalisation n'est pas minimale. très peu de représentation compacte en

4.4 Results comparison

5 Conclusion

C'est vraiment génial comme travail, on va tuer le game du DETNET.

References

[1] D. Barth, M. Guiraud, B. Leclerc, O. Marce, and Y. Strozecki, "Deterministic scheduling of periodic messages for cloud RAN," in *2018 25th International Conference on Telecommunications (ICT) (ICT 2018)*, (Saint Malo, France), June 2018.