Phase expérimentale pour le TLBP/B-P

Dans ce chapitre, nous rapportons les résultats des tests expérimentaux que nous avons effectués pour le problème TLBP/B-P. L'objectif de cette étude est triple, il s'agit d'abord de montrer l'apport des différentes améliorations en particulier celle de la réduction du nombre de variables. Ensuite, nous comparons les différents modèles, et enfin nous étudions l'influence des paramètres du problème. Tous les calculs ont été effectués sur une station de travail de type HP Xeon, avec un processeur de 3,6 GHz et une mémoire vive de 2 Go. Les programmes sont écrits en langage C++ et le compilateur utilisé est Visual Studio version 6.0.

6.1 Description de l'environnement des tests

6.1.1 Le logiciel de résolution

Nous utilisons les logiciels ILOG Cplex et ILOG Solver qui sont des librairies programmées en C++ proposées par ILOG pour la résolution de modèles linéaires en nombres entiers, en variables mixte ou encore de modèles quadratiques. Pour résoudre un PLNE, Cplex met en œuvre un algorithme de *Branch & Cut*, c'est-à-dire une procédure par séparation et évaluation intégrant plusieurs types de coupes telles que les coupes de cliques et les coupes de sac à dos. Quant à Solver c'est une PSE qui intègre des algorithmes de filtrage associés à certaine contraintes du type *allDiff* pour imposer aux variables de prendre des valeurs distinctes.

6.1.2 Jeux de données (instances)

Pour pouvoir faire des conclusions statistiques, nous générons aléatoirement un grand nombre des jeux de données. Pour générer ces instances, nous fixons les paramètres tels que les densités des graphes de contraintes et les cardinalités des ensembles **N** et **B**. Nous y intégrons l'ensemble des pré-traitements décrits précédemment (voir la sous-section 4.3.1). Nous vérifions également la cohérence des paramètres entre eux. Par exemple, si nous générons un

graphe de précédence dont le chemin le plus long est égale à une valeur $k+1$, il n'est pas possible d'imposer une borne supérieure sur le nombre de stations m_0 qui soit inférieure[1] à k . Chaque instance générée est donc réalisable. Nous indiquons dans le tableau 6.1 les paramètres les plus importants que nous avons utilisé pour construire les jeux de données.

Paramètres	descriptif
$OpMin$	le nombre minimum d'opérations par bloc, par défaut cette valeur est à 1
$OpMax$	le nombre maximum d'opérations par bloc, par défaut cette valeur est à 5
$Min\|\mathbf{B}\|$	la cardinalité minimale de l'ensemble $\mathbf{B}$
$Max\|\mathbf{B}\|$	la cardinalité maximale de l'ensemble $\mathbf{B}$
$Avg\|\mathbf{B}\|$	la cardinalité moyenne de l'ensemble $\mathbf{B}$
OS	la densité du graphe de précédence G^{or}
OS_{min}	la densité minimale du graphe G^{or}
OS_{max}	la densité maximale du graphe G^{or}
OS_{avg}	la densité moyenne du graphe G^{or}
OSI	l'intervalle défini par $[OS_{min}, OS_{max}]$

TAB. 6.1 – Descriptif des paramètres pour la génération d'instances d'une famille

Nous utilisons OS comme mesure de la densité du graphe de précédence [Sch99]. Pour la définir, nous introduisons la matrice P telle que tout élément p_{ij} de cette matrice est décrit par :

$$p_{ij} = \begin{cases} 1 & \text{si l'opération } i \text{ précède l'opération } j, \\ 0 & \text{sinon.} \end{cases}$$

La densité du graphe de précédence est alors donnée par :

$$OS = \frac{2z}{|\mathbf{N}|(|\mathbf{N}|-1)} \tag{6.1}$$

où z est le nombre d'éléments non nuls de P. Ce ratio est nul pour un jeu de données qui ne comporte aucune contrainte de précédence. Lorsque l'ordre entre les opérations est complet, c'est-à-dire qu'il y a plus qu'une seule séquence d'opérations possible, alors ce ratio est égal à un.

Il est à souligner que nous avons généré chacune des instances de manière à ce que les densité des différents graphes de contraintes ainsi que leur structure soit les plus proches de celles des instances industrielles dont nous disposons. Par exemple, les contraintes de précédence dans l'usinage (contrairement à l'assemblage) ont une forme de graphe série-parallèle, c'est-à-dire plusieurs séquences d'opérations sont en parallèle telles qu'une séquence possible peut être : perçage d'ébauche, filetage et finition. De façon générale, nous testons 7 familles d'instances que nous décrivons dans le tableau 6.2. Pour chaque famille, nous serons amenés à générer différentes sous-familles en fonction de l'étude effectuée que nous décrirons plus en détails aux sous-sections correspondantes.

[1]Ceci est dû à l'activation parallèle des blocs au sein d'une station, suite à quoi deux opérations liées par une contrainte de précédence ne peuvent pas être affectées simultanément à la même station.

Familles	$\|\mathbf{N}\|$	m_0	$[Min\|\mathbf{B}\|, Max\|\mathbf{B}\|]$	$Avg\|\mathbf{B}\|$
F_{10}	10	6	[13,17]	15
F_{15}	15	8	[20,27]	25
F_{20}	20	13	[29,31]	30
F_{30}	30	16	[45,50]	48
F_{40}	40	21	[57,70]	68
F_{50}	50	26	[67,94]	93
F_{60}	60	30	[117,119]	113
F_{70}	70	34	[125,135]	133
F_{80}	80	38	[144,154]	152

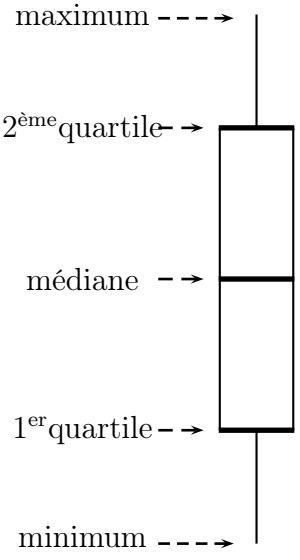
TAB. 6.2 – Les valeurs des paramètres pour la génération des familles instances

6.1.3 Représentation des résultats statistiques

Nous rapportons les résultats des expérimentations globalement sous deux formes. La première forme est classique, elle correspond à un nuage de points représentant chacun le temps de résolution d'une instance donnée appartenant à une des familles. Nous utilisons cette représentation lorsque le nombre d'instances testées le permet.

Pour les sections dont le nombre de tests est très important, nous employons une seconde forme nommée **candlesticks**. Celle-ci représente les temps d'exécution des instances appartenant à une sous-famille donnée. Sa représentation graphiquement nécessite de calculer les valeurs suivantes :

- **le maximum** : le point extrême le plus haut (limite de la droite),
- **le 3ème quartile** : pour un échantillon de valeurs triées, il correspond à la valeur délimitant deux tranches, à savoir : celle des 75% plus petites valeurs des 25% restantes. Cette valeur représente graphiquement le côté supérieur du rectangle,
- **le 1er quartile** : cette valeur délimite les 25% plus petites valeurs de l'échantillon des autres qui leurs sont supérieures, il est indiqué graphiquement par le côté inférieur du rectangle,
- **le minimum** : le point extrême le plus bas. Nous intégrons la médiane de l'échantillon représentée graphiquement par un trait horizontal.



6.2 Évaluation de l'approche PPC

Nous comparons dans cette section l'approche PPC avec le modèle linéaire orienté blocs. Pour ce faire, nous testons les trois premières familles décrites précédemment et indiquons les temps de calcul obtenus dans les figures 6.1, 6.2 et 6.3.

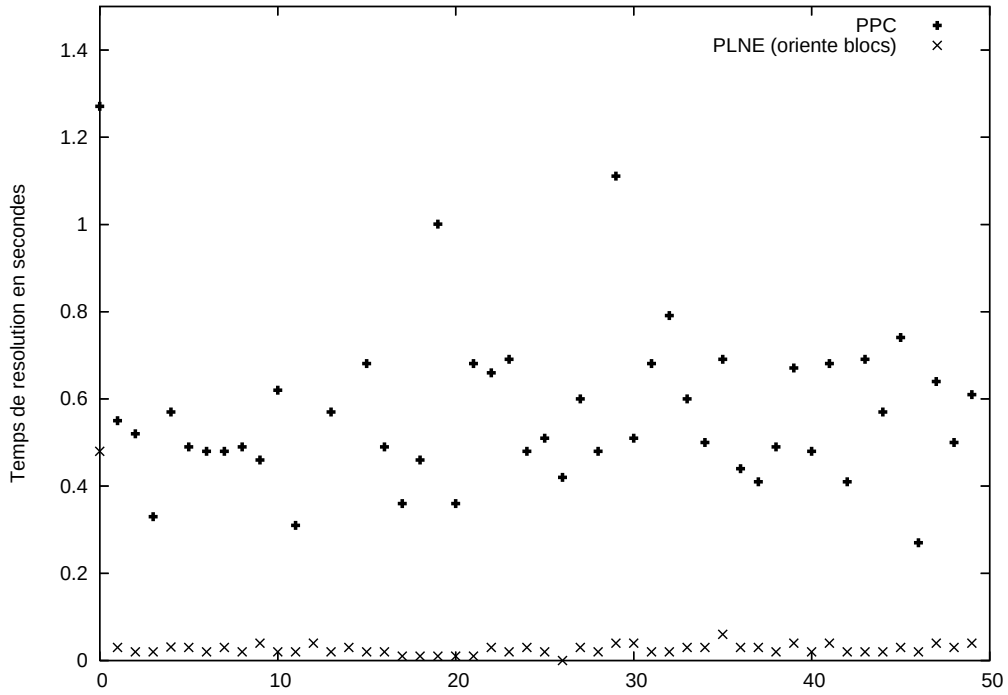


FIG. 6.1 – Temps de calcul des modèles PPC et PLNE pour les instances de F_{10}

Pour l'ensemble des 150 instances testées, le modèle linéaire orienté blocs (5.17)-(5.25) est plus performant. En effet, quand le modèle orienté blocs résout l'ensemble des instances en près d'une seconde, le modèle PPC peut aller jusqu'à 350 sec (voir figures 6.2 et 6.3). De plus, le modèle PPC n'a pas permis de résoudre certaines instances ayant plus de 20 opérations (sous la limite de 2h de temps). Nous signalons toutefois les difficultés rencontrées lors de l'implémentation du modèle PPC qui sont essentiellement dues au manque de documentation sur ILOG Solver d'une part et à l'instabilité du logiciel lui-même lors de la résolution de nombreuses instances d'autre part.

6.3 Modèle orienté blocs

6.3.1 Réduction du nombre de variables

L'objectif de cette sous-section est de faire une analyse comparative de l'efficacité des algorithmes de réduction du nombre de variables. Ces algorithmes calculent les intervalles

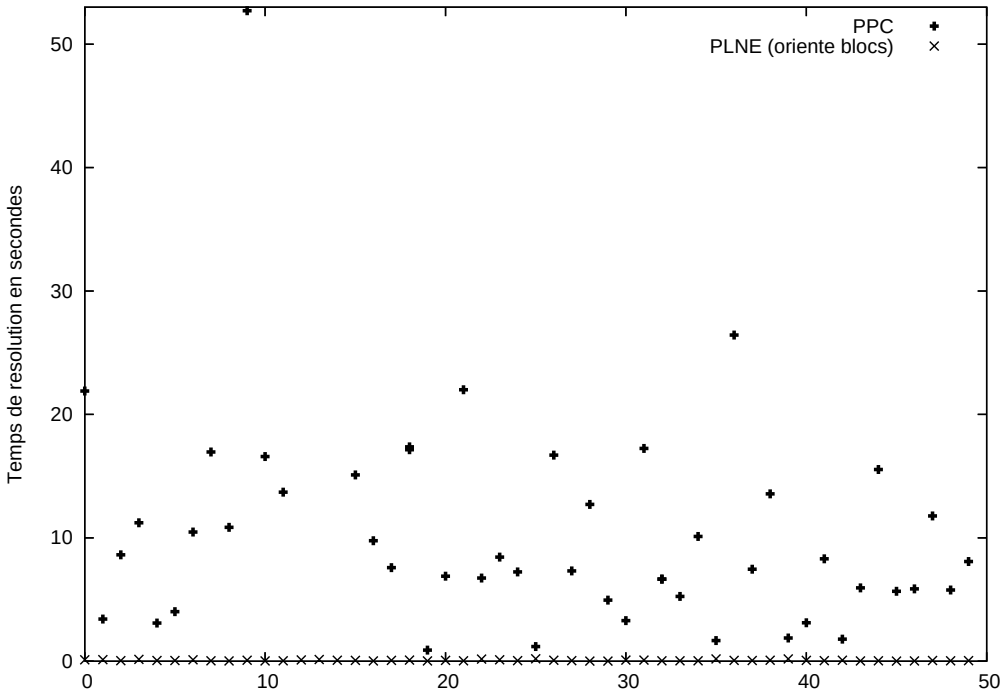


FIG. 6.2 – Temps de calcul des modèles PPC et PLNE pour les instances de F_{15}

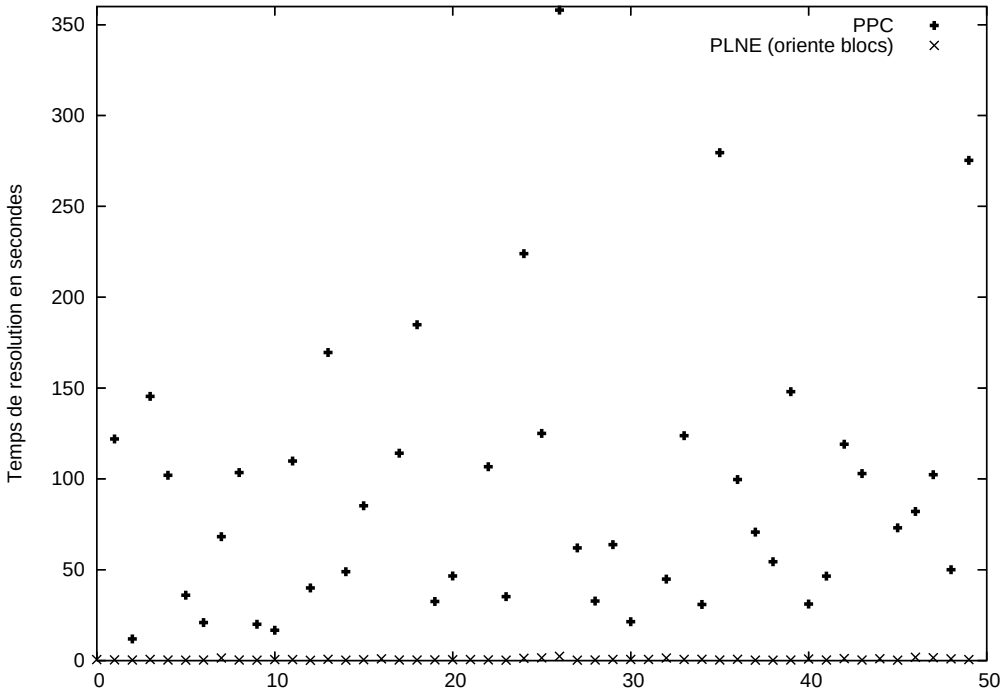


FIG. 6.3 – Temps de calcul des modèles PPC et PLNE pour les instances de F_{20}

des indices de stations pour chaque bloc $[head_b, tail_b]$. Nous avons implémenté l'Algorithme 1 et l'algorithme 2 (voir section 5.4).

À partir de cette section, nous étudions uniquement les familles dont le nombre d'opérations est supérieur à 20, car les autres familles ont des temps de calcul beaucoup trop faible. Pour chaque famille x, nous générons 3 sous-familles $F_x^y, y = 1, 2, 3$ de 50 instances chacune. Les sous-familles appartenant à une même famille se distinguent par la densité du graphe de précédence. Ceci est dû au fait que les algorithmes de réduction exploitent la structure du graphe de précédence, il est donc intéressant de voir l'évolution du temps de calcul en fonction de ce paramètre. Les autres paramètres des sous-familles appartenant à la même famille sont identiques. Nous ne fixons pas la densité exacte mais un intervalle dans lequel elle varie. Ceci est dû à la génération aléatoire des exemples. Nous avons défini trois intervalles (noté par l'indice $y = 1, 2, 3$) pour chaque famille F_x. Les sous-familles F_x^1 ont une faible densité (entre 8 % et 32 %), les F_x^2 ont une densité moyenne (entre 40 % et 65 %) et les F_x^3 une forte densité (entre 60 % et 89 %) du graphe de précédence.

Dans le tableau 6.3, nous montrons le gain apporté. Les colonnes $NbVars1$ et $NbVars2$ contiennent le nombre moyen de variables binaires de la sous-famille correspondante après avoir appliqué l'Algorithme 1 et Algorithme 2, respectivement. Nous indiquons également, dans les colonnes $gain1$ et $gain2$ les pourcentages de gain réalisé en appliquant l'Algorithme 1 et Algorithme 2, respectivement. Ce gain est calculé en utilisant (6.2), il montre l'écart par rapport au nombre initial de variables (soit $m_0 \times Avg|\mathbf{B}|$).

$$gain = \frac{(m_0 \times Avg|\mathbf{B}|) - NbVars}{m_0 \times Avg|\mathbf{B}|} \tag{6.2}$$

Le tableau 6.3 permet de constater que la réduction du nombre de variables binaires en utilisant le calcul des rangs (Algorithme 1) est significative. En observant les différentes sous-familles F_x^y, $x = |\mathbf{N}| = \{20, 30, 40, 50, 60, 70, 80\}$ et $y = 1, 2, 3$, nous apercevons que le gain en nombre de variables est proportionnel à la densité du graphe de précédence. En fait, plus les contraintes de précédence sont nombreuses, plus les rangs des blocs sont serrés et plus la réduction qui les exploitent est efficace. Ce gain peut aller jusqu'à 72 % pour la sous-famille F_{60}^3 dont la densité est dans l'intervalle $[80, 89]$.

L'Algorithme 1 donne déjà un gain intéressant. L'Algorithme 2 permet d'en réaliser davantage. Ce dernier apporte un gain supplémentaire d'au moins[2] 2% par rapport à l'Algorithme 1. Ce gain peut même atteindre 10 % dans certains cas, par exemple pour les familles F_{50}^1 et F_{70}^2.

6.3.2 Temps de calcul

La figure 6.4 représente les temps d'exécution des instances appartenant aux sous-familles de 20 et 30 opérations. Pour chaque sous-famille F_x^y nous donnons trois résultats, le premier, noté A_0, est le temps de résolution sans l'application d'algorithmes de réduction. Le second,

[2]Par rapport à l'échantillon de familles testées.

					Algo1		**Algo2**	
	$\|\mathbf{N}\|$	OSI	OS_{avg}	$m_0 \times Avg\|\mathbf{B}\|$	$NbVars1$	gain1	$NbVars2$	gain2
F_{20}^1		$[8, 15]$	12		330	**15** %	324	**17** %
F_{20}^2	20	$[40, 50]$	35	13×30 = 390	263	**32** %	253	**35** %
F_{20}^3		$[60, 81]$	63		183	**53** %	175	**55** %
F_{30}^1		$[15, 32]$	22		565	**26** %	547	**29** %
F_{30}^2	30	$[42, 65]$	52	16×48 = 768	416	**46** %	394	**49**%
F_{30}^3		$[62, 80]$	70		287	**63** %	262	**66**%
F_{40}^1		$[15, 29]$	20		1114	**22** %	1084	**24**%
F_{40}^2	40	$[40, 60]$	52	21×68 = 1428	817	**42** %	788	**45**%
F_{40}^3		$[68, 88]$	79		440	**69** %	400	**72**%
F_{50}^1		$[17, 31]$	24		1885	**22** %	1826	**32**%
F_{50}^2	50	$[47, 64]$	54	26×93 = 2418	1442	**40** %	1385	**43**%
F_{50}^3		$[78, 87]$	83		773	**68** %	646	**73**%
F_{60}^1		$[9, 18]$	13		3019	**11** %	2944	**13**%
F_{60}^2	60	$[47, 64]$	58	30×113 = 3390	2133	**37** %	2008	**41**%
F_{60}^3		$[80, 89]$	84		951	**72** %	854	**75**%
F_{70}^1		$[12, 19]$	15		3817	**16** %	3708	**18,5**%
F_{70}^2	70	$[44, 62]$	51	34×133 = 4522	3000	**34** %	2854	**44**%
F_{70}^3		$[79, 88]$	84		1428	**68** %	1255	**75**%
F_{80}^1		$[12, 28]$	18		4879	**16** %	4751	**19**%
F_{80}^2	80	$[43, 56]$	49	38×152 = 5776	4066	**30** %	3920	**33**%
F_{80}^3		$[80, 89]$	84		1918	**66** %	1712	**70**%

TAB. 6.3 – Analyse comparative des algorithmes de réduction du nombre de variables

libellé A_1, indique les temps de résolution des instances après le pré-traitement avec l'Algorithme 1. Enfin, le troisième résultat, nommé A_2, correspond à l'utilisation de l'Algorithme 2. Ainsi, pour comparer les performances des algorithmes pour une sous-famille donnée F_x^y, $x = 20, 30$ et $y = 1, 2, 3$, il faut observer les trois abscisses F_x^y : A_0, A_1 et A_2. Il est important de préciser que nous employons une échelle logarithmique, car la variation des résultats est importante.

Sur l'ensemble de ces sous-familles, nous constatons un gain considérable du temps de résolution suite à l'application de l'Algorithme 1. En particulier, pour les sous-familles de 30 opérations, l'ensemble des indicateurs notamment la médiane est divisée par 10 grâce à l'Algorithme 1. Ce qui signifie que plus de 50 % des instances sont résolues 10 fois plus rapidement. L'application de l'Algorithme 2 permet davantage d'accélération dans les calculs. Pour la famille F_{30}, le gain obtenu avec l'Algorithme 2 par rapport à celui de l'Algorithme 1 n'est toutefois pas aussi important que le gain réalisé par l'Algorithme 1 par rapport à A_0 (sans l'utilisation d'algorithmes de réduction) pour la même famille.

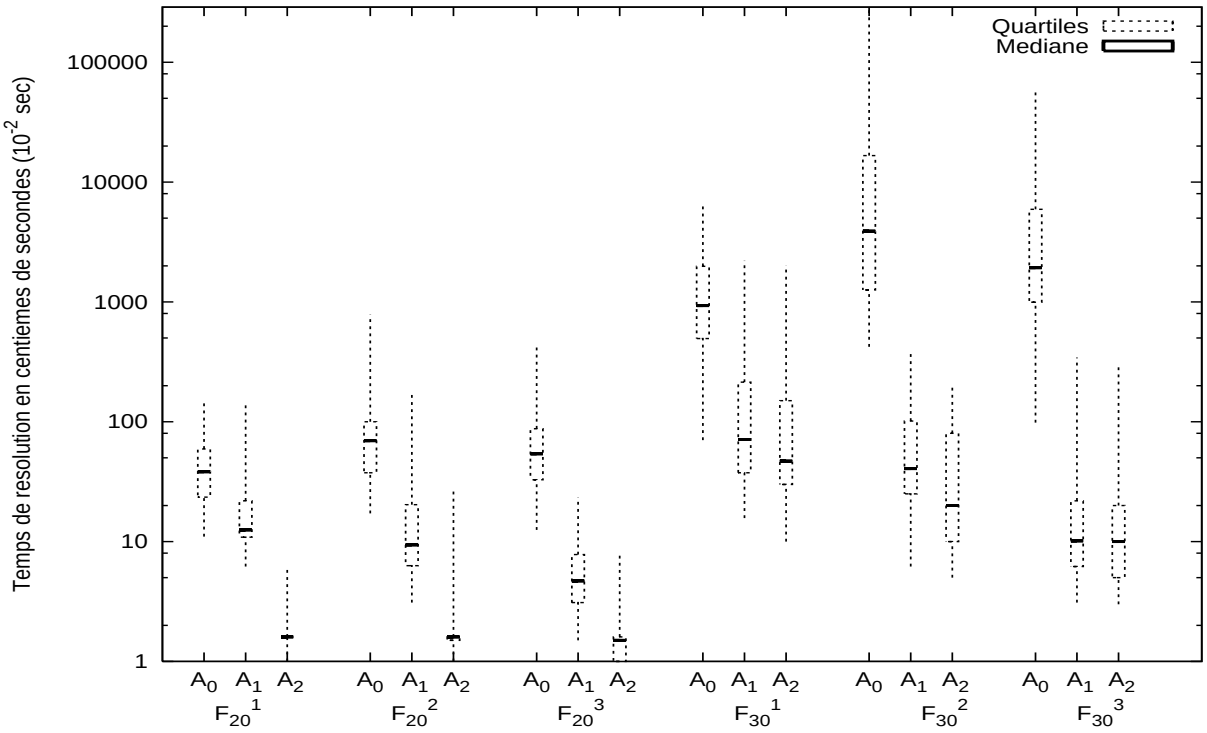


FIG. 6.4 – Le gain obtenu avec les Algorithmes 1 et 2 pour les instances de F_{20}^x et F_{30}^x

6.3.3 Apport de la modification de l'objectif (epsilon)

Pour cette partie des expérimentations, nous appliquons l'Algorithme 1 et testons l'impact de la modification de l'objectif en introduisant une constante ε dans la formulation de l'objectif. Nous résolvons les familles de 20, 30, 40, 50 et 60 opérations avec les deux formulations de l'objectif, à savoir : avant et après l'introduction de l'*epsilon* (voir respectivement, (5.17) et (5.35)).

Les abscisses dans la figure 6.5 sont à observer par paires successives. Nous utilisons un asterisk pour désigner les temps d'exécution après modification de l'objectif en introduisant l'*epsilon*.

Pour les familles de 20 et 30 opérations, soit la figure 6.5, la modification suite à l'ajout d'un epsilon n'améliore pas systématiquement les temps de calcul. En effet, nous remarquons une amélioration des temps pour trois sous-familles : F_{20}^1, F_{20}^2 et F_{30}^1, tandis que nous notons une détérioration des 3 autres sous-familles. Nous pensons que l'emploi de l'epsilon pour ces instances de petites taille a dû perturber l'ordre de branchements ce qui a eu pour effet de ralentir le processus de résolution.

Pour les instances de taille supérieure, c'est-à-dire celles appartenant aux familles de 40, 50 et 60 opérations, nous constatons une nette amélioration (voir les figures 6.6 et 6.7). Le gain apporté est significatif car il permet une réduction de 50% du temps de résolution par rapport au modèle initial utilisant la formulation (5.17).

Dans la figure 6.8, nous rapportons les temps d'execution de toutes les instances des

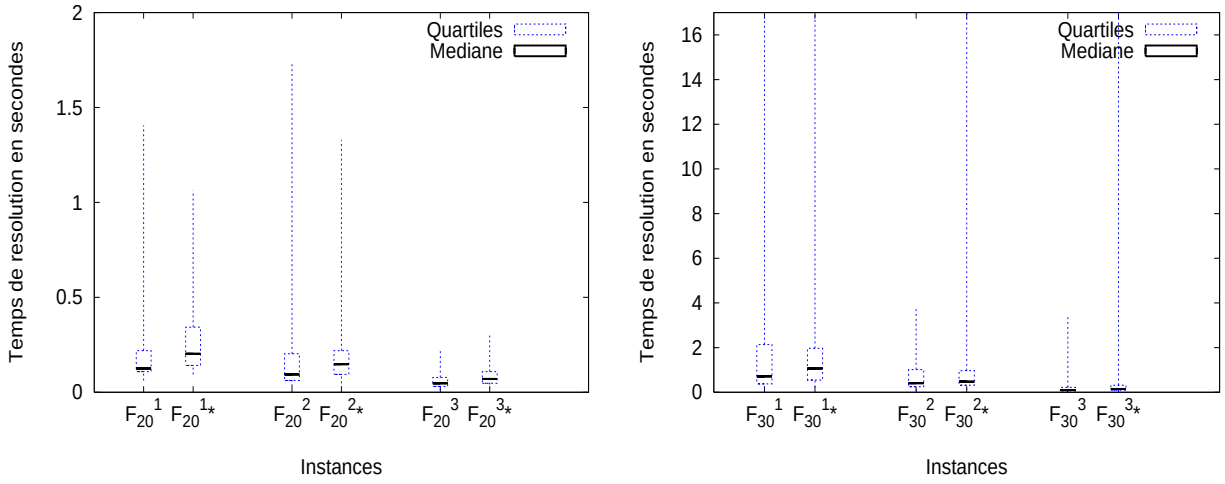


FIG. 6.5 – Influence de la modification de l'objectif pour les instances de 20 et 30 opérations

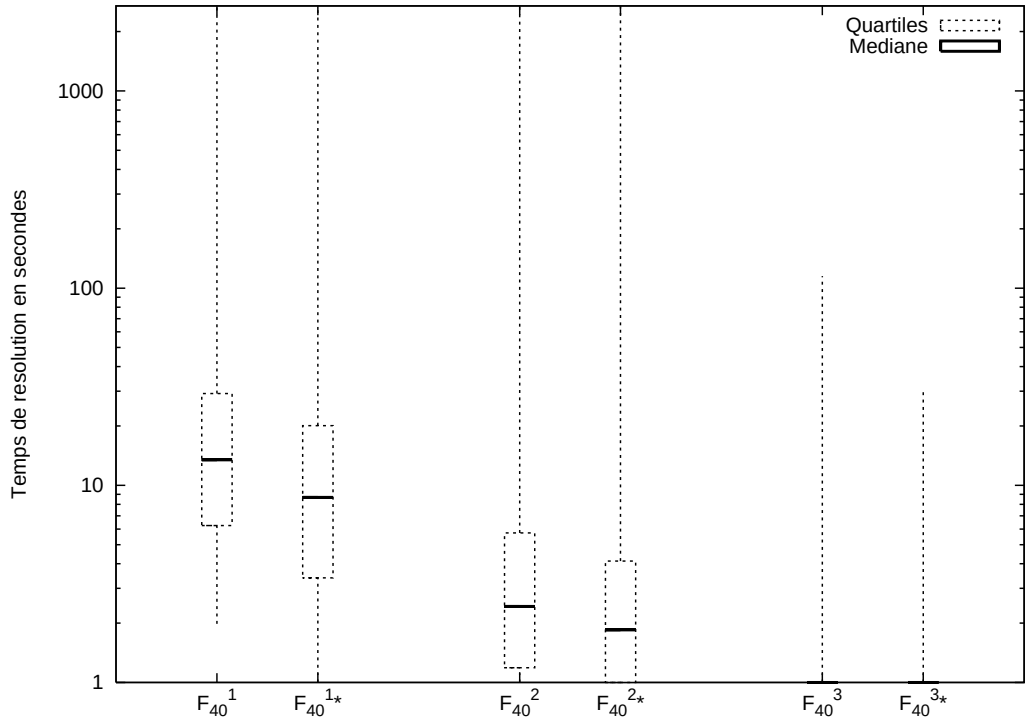


FIG. 6.6 – Influence de la modification de l'objectif pour les familles de 40 opérations

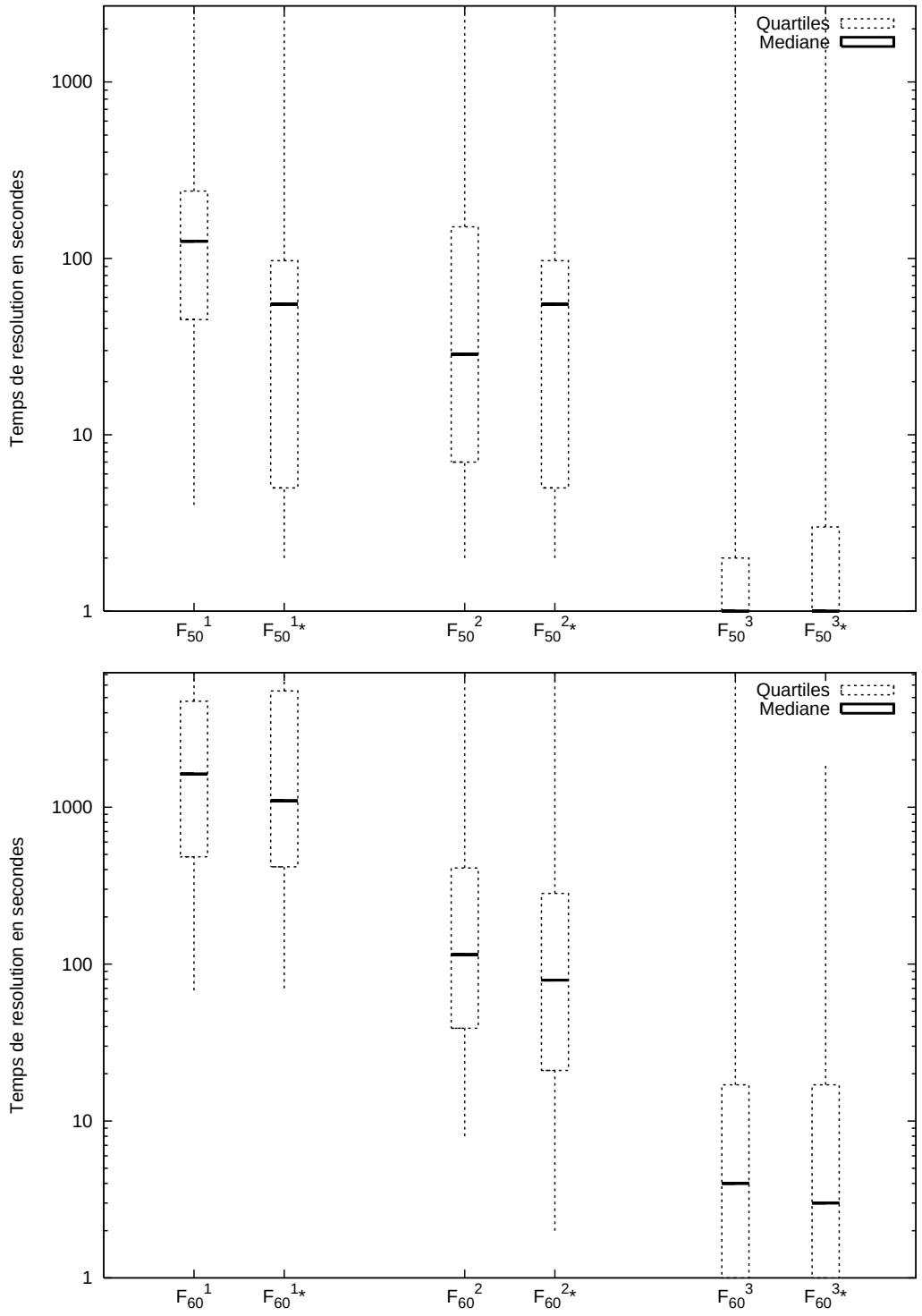


FIG. 6.7 – Influence de la modification de l'objectif pour les familles de 50 et 60 opérations

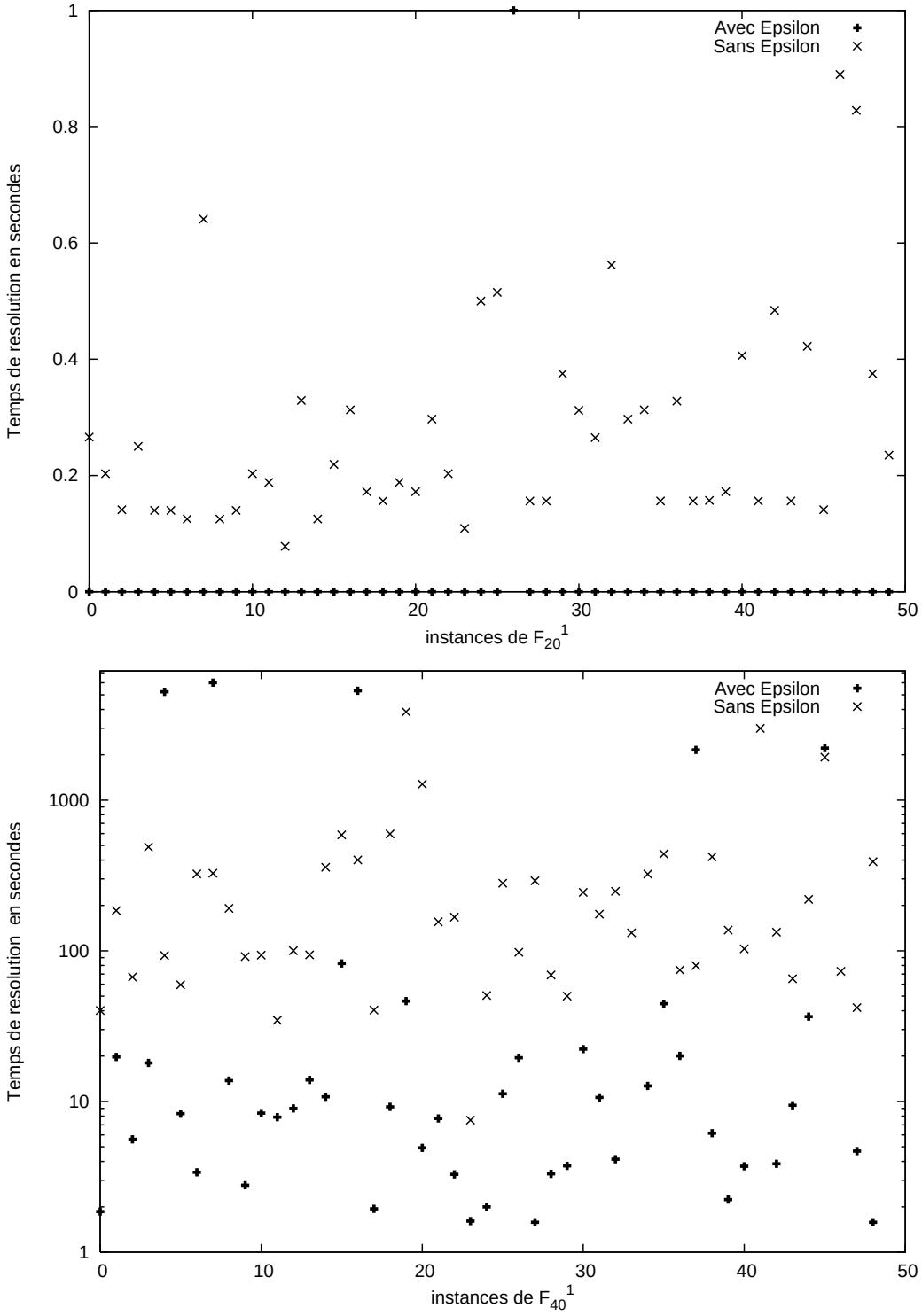


FIG. 6.8 – Influence de la modification de l'objectif pour F_{20}^1 et F_{40}^1

sous-familles F_{20}^1 et F_{40}^1 telle que chaque abscisse représente une instance (nuage de points). Ces résultats permettent de voir l'amélioration apportée par la modification de l'objectif pour chacune des instances.

Pour conclure, nous pouvons dire que l'amélioration apportée par la reformulation de l'objectif en introduisant une constante ε est significative pour les cas les plus difficiles à résoudre, toutefois cette réduction n'est pas garantie pour toute instance. En effet, concernant les instances de petite taille cette modification peut engendrer une perturbation dans le schéma de résolution de Cplex et donc le ralentir, aussi il n'est pas forcément judicieux de l'utiliser, pour de telles instances, d'autant plus que le temps de calcul n'est pas très important dans ces cas (n'excède pas les 20 secondes).

6.4 Modèle orienté opérations

Dans cette partie des expérimentations, l'Algorithme 2 est employé dans la phase de réduction préliminaire. Afin que la comparaison avec le modèle orienté blocs soit valide, nous avons également appliqué l'Algorithme 2 pour le modèle orienté blocs.

6.4.1 Comparaison avec le modèle orienté blocs

Nous rapportons les résultats des tests pour les familles de 20, 30, 40, 50 et 60 opérations que nous avons résolues avec les deux modèles linéaires. Les abscisses libellées MB indiquent les résultats obtenus en utilisant le modèle orienté blocs alors que celles libellées MO indiquent ceux du modèle orienté opérations.

Dans la figure 6.9, nous montrons les temps de résolution de la famille F_{20} alors que dans la figure 6.10 nous rapportons les temps de chacune des instances appartenant à la sous-famille F_{20}^1. Les figures 6.11 et 6.12 montrent les résultats obtenus pour les famille de 30 à 60 opérations. Pour les familles 40, 50 et 60 opérations, l'échelle des graphiques est logarithmique. Par exemple, le temps maximum pour la sous-famille F_{40}^1 est égale à 7200 sec pour le modèle orienté blocs alors qu'il est ramené à 11 sec dans le modèle orienté opérations.

En observant l'ensemble des figures, nous constatons une forte réduction des temps d'exécution en faveur du modèle orienté opérations. En effet, le modèle orienté opérations permet une résolution en un dixième de seconde pour les instances appartenant aux familles de 20 et de 30 opérations (à l'exception d'une instance de la sous-famille F_{30}^1 qui est résolue en 2.5 sec, voir le maximum $F_{30}^1 : MO$).

Concernant les familles de 40, 50 et 60 opérations, la limite de temps de 7200 sec accordée aux deux modèles est systématiquement dépassée pour le modèle orienté blocs (sauf pour F_{60}^3). Ainsi, pour les instances difficiles soit celles de F_x^1, où $x \in \{40, 50, 60\}$), le modèle orienté blocs est incapable d'apporter les preuves de l'optimalité des solutions trouvées tandis que le modèle orienté opérations résout à l'optimum l'ensemble des instances en réduisant le temps de calcul de façon considérable (pour F_{60}^1 le temps de calcul maximum dépasse la limite 7200 avec le modèle orienté blocs il est réduit à 32 sec à l'aide du modèle orienté opérations).

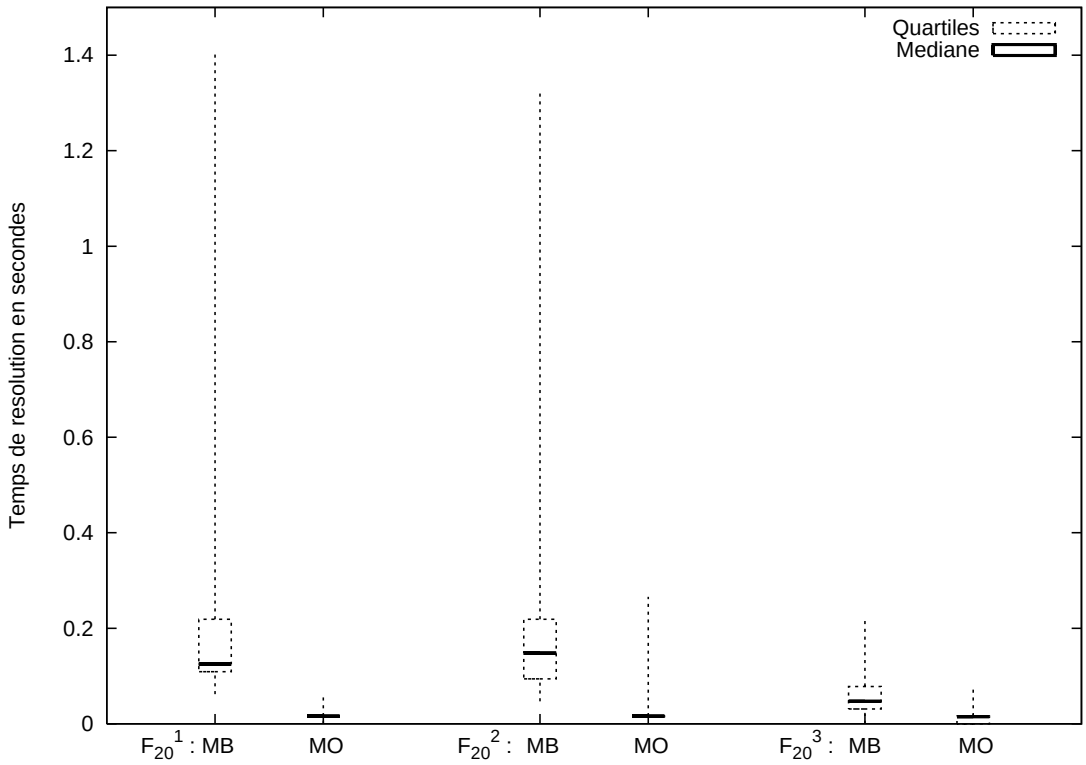


FIG. 6.9 – Résolution des familles de 20 opérations avec les deux modèles linéaires

Pour conclure, nous dirons que le modèle orienté opérations est plus performant tant sur le plan du temps de calcul des instances difficiles que pour sa stabilité (résolvant des instances jusqu'à 80 opérations).

6.4.2 Étude de l'influence des précédences (OS)

La figure 6.13 permet de voir l'évolution des temps de résolution pour les familles de 20, 30, 40 et 50 opérations. Pour une même famille, il faut observer les trois abscisses libellées F_x^1, F_x^2 et F_x^3, elles correspondent aux sous-familles avec faible, moyenne et forte densité, respectivement. Nous constatons une réduction significative des temps de calcul pour les sous-familles ayant une plus grande densité du graphe de précédence. La même conclusion peut être faite pour les familles de 60, 70 et 80 opérations dont les résultats sont dans la figure 6.14. Les médianes des sous-familles F_x^3 par rapport à celles des F_x^1, sont divisées par 10 ou 100 selon les familles (100 fois pour les 60 et 70 et plus pour 80 opérations).

De plus, une influence directe du nombre d'opérations sur le temps d'execution est à noter. Les sous-familles de même densité semblent requérir un temps de calcul croissant en fonction du nombre d'opérations. La sous-famille de F_{80}^1 est la plus difficile à résoudre par rapport aux autres sous-familles de type F_x^1. En effet, pour cette sous-famille, plus de 25% des instances (voir le 3$^{\text{ème}}$ quartile de F_{80}^1 dans la figure 6.14) nécessitent un temps de résolution

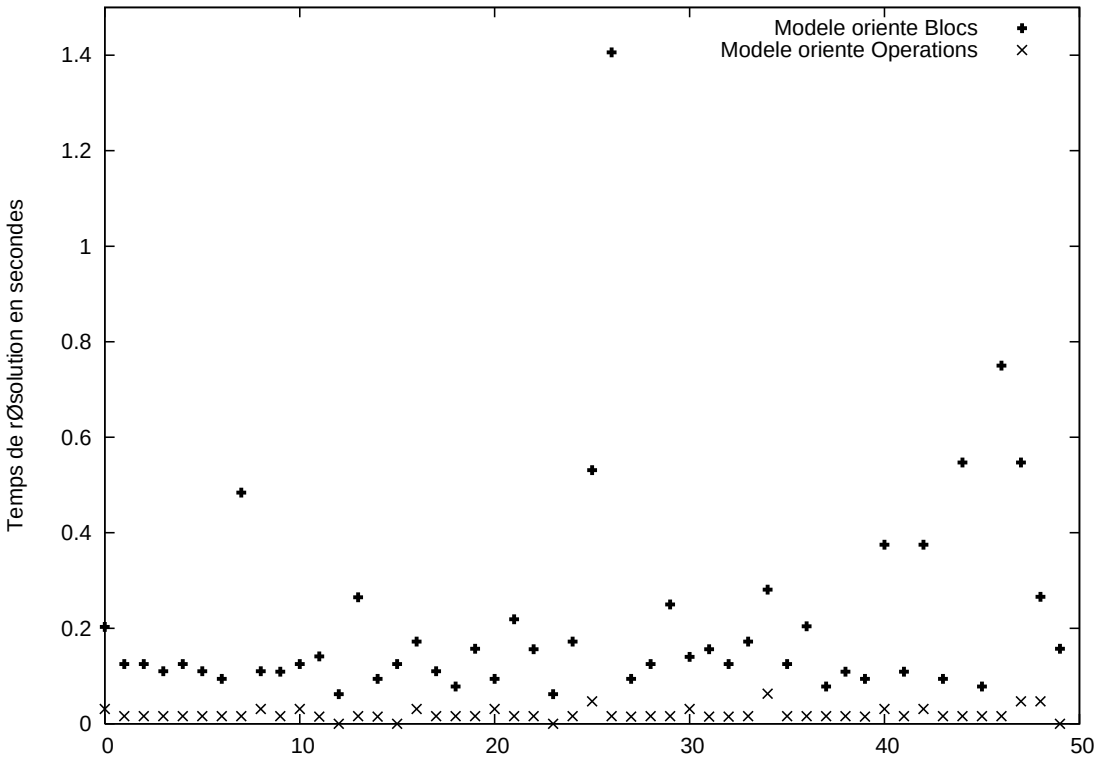


FIG. 6.10 – Temps de calcul de F_{20}^1 avec les deux modèles linéaires

supérieur à la limite que nous avons accordé au solver, soit 7200 sec. De la même manière, les sous-familles F_{80}^2 et F_{80}^3 sont plus coûteuse en temps de calcul que F_x^2, F_x^3, respectivement tel que $x < 80$.

Sur la base des expérimentations comprenant 1050 tests, nous constatons également un lien direct entre la densité du graphe de précédence et le temps de calcul des instances. En observant chacune des familles séparément, nous constatons une nette diminution du temps d'exécution lorsque la densité OS augmente. Ainsi, pour chaque famille, la sous-famille à faible densité est toujours plus difficile à résoudre que celle ayant une densité moyenne. De même, les sous-familles ayant une moyenne densité sont plus coûteuses en temps de calcul que celles ayant une forte densité.

La difficulté des instances à faible densité peut s'expliquer par le grand nombre d'affectations possibles des blocs aux stations. Ceci a une répercussion directe sur le nombre de branchements à générer lors de la construction de l'arbre de séparation et d'évolution par Cplex. Toutefois, ce paramètre n'est pas le seul à influencer le temps de calcul d'autant plus que la difficulté de certaines instances peut être due à une combinaison particulière de données.

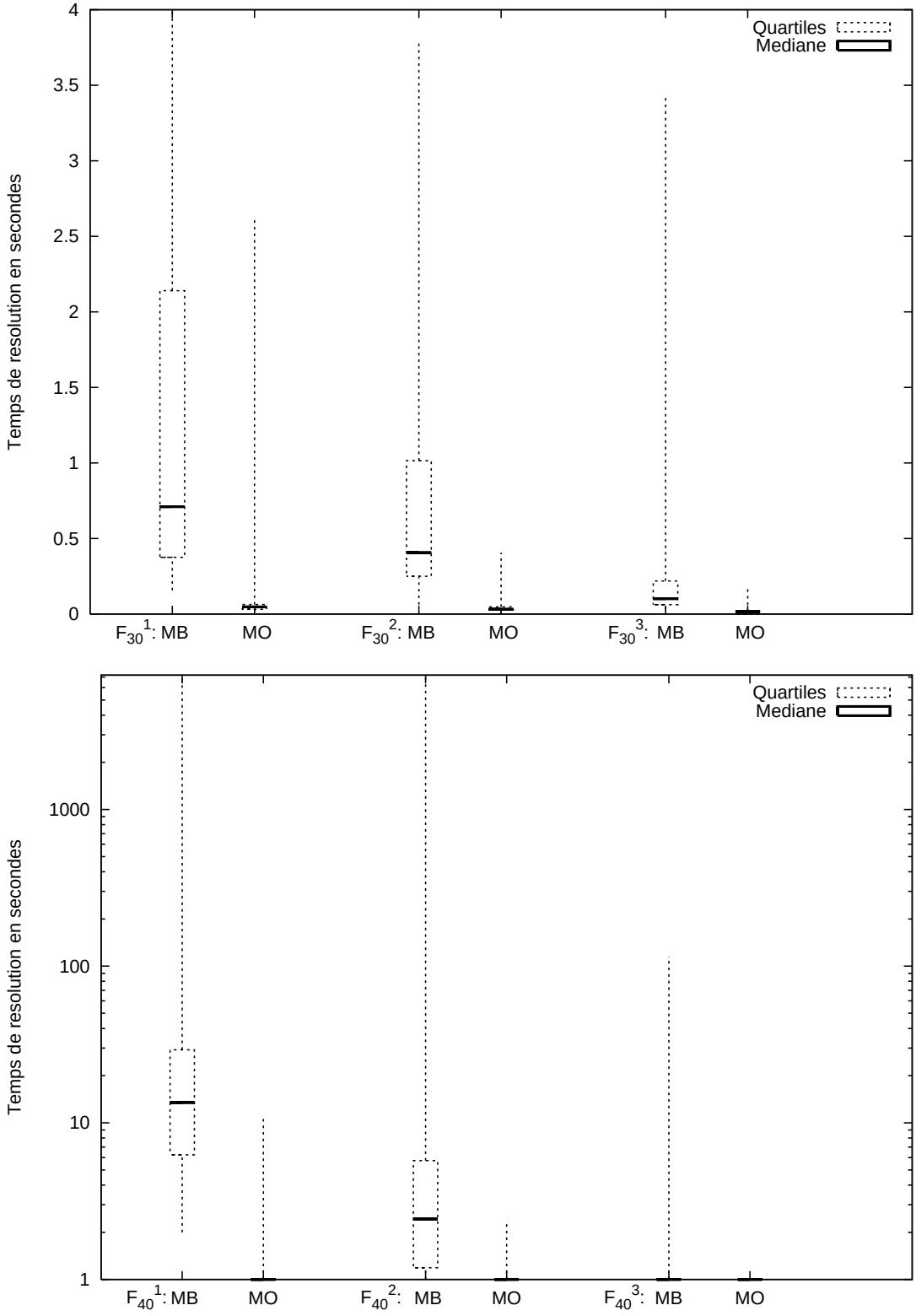


FIG. 6.11 – Résolution des familles de 30 et 40 opérations avec les deux modèles linéaires

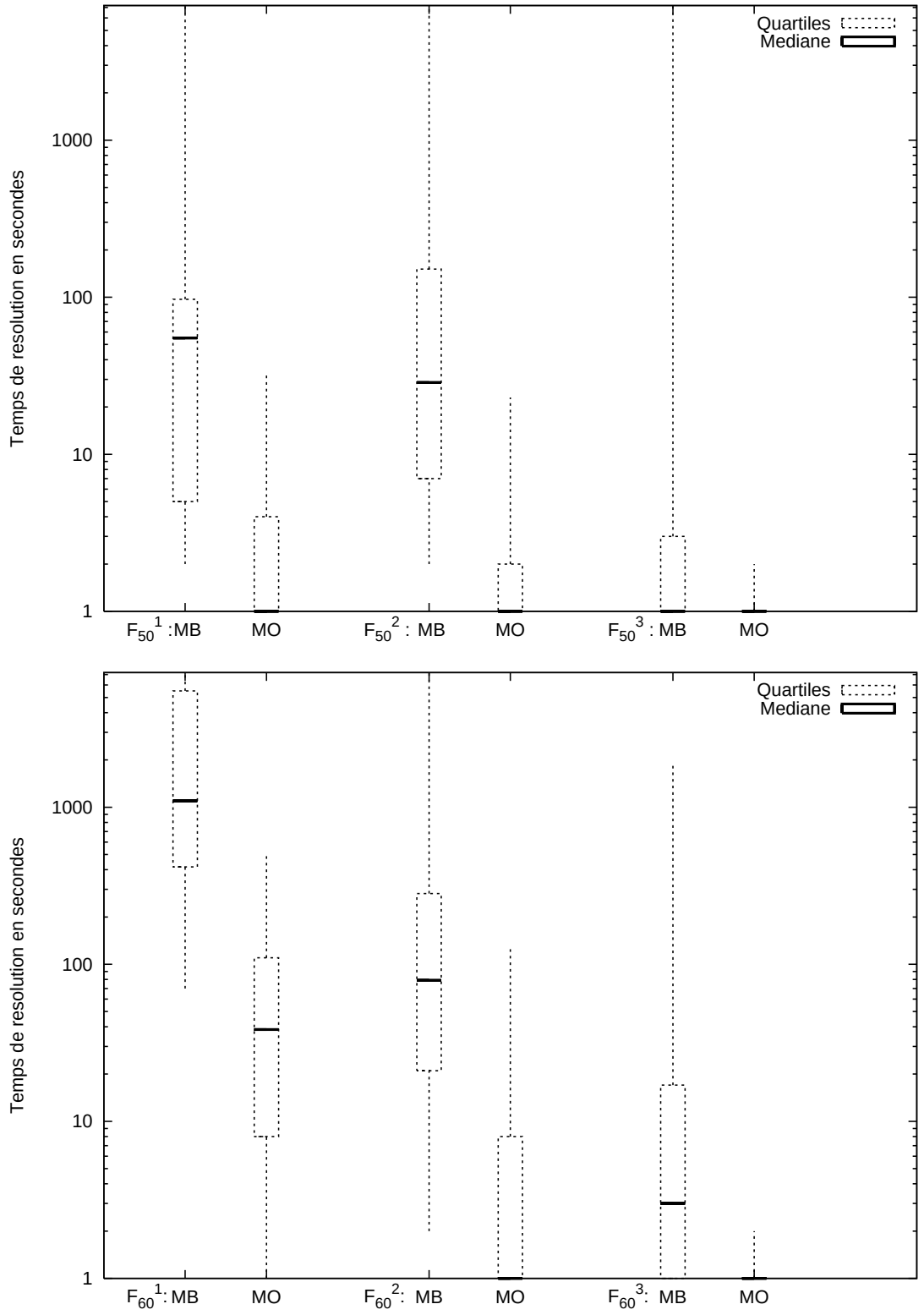


FIG. 6.12 – Résolution des familles de 50 et 60 opérations avec les deux modèles linéaires

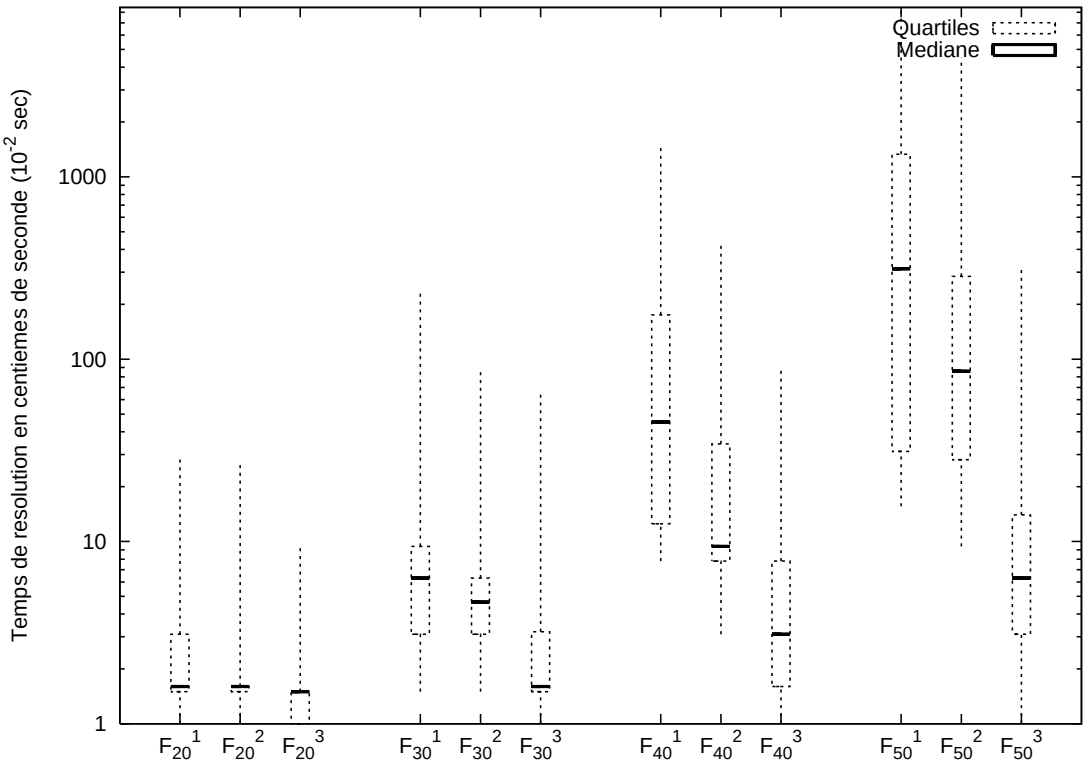


FIG. 6.13 – Temps de calcul en fonction de OS pour les familles de 20, 30, 40 et 50 opérations

6.4.3 Étude de l'influence du nombre de blocs (|B|)

Nous consacrons la présente sous-section pour l'étude de l'influence de la cardinalité de l'ensemble **B** sur le temps de résolution. Le tableau 6.4 décrit les familles d'instances qui ont été employées dans cette partie des expérimentations. Pour chaque famille, nous avons généré trois sous-familles comportant 50 instances chacune. Les sous-familles ont les mêmes caractéristiques : $|\mathbf{N}|$, m_0 et OSI.

Les sous-familles notées $F_x^{B_1}$ ont un nombre moyen de blocs égale à une fois et demi le nombre d'opérations. Par exemple, la sous-famille $F_{20}^{B_1}$ à un nombre moyen de blocs $Avg|\mathbf{B}|$= 30 = 1,5×20. Les sous-familles $F_x^{B_2}$ ont un nombre moyen de blocs qui est égal au double du nombre d'opérations. Quant aux dernières sous-familles : $F_x^{B_3}$, elles ont un nombre de blocs, égal au triple du nombre d'opérations.

La figure 6.15 montre le temps d'exécution pour les familles de 20, 30, 40, 50, 60 et 70 opérations. Nous constatons que les sous-familles $F_x^{B_3}$ sont plus coûteuses en temps de calcul par rapport aux $F_x^{B_1}$ et $F_x^{B_2}$. Leur temps de résolution augmente de façon significative : la médiane de $F_{20}^{B_3}$ est deux fois plus élevée que celle de $F_{20}^{B_1}$. Cette différence est encore plus visible pour la famille de 30 et 40 opérations, où la médiane de $F_{30}^{B_3}$ est 7 fois plus élevée que celle de $F_{30}^{B_2}$ et celle de $F_{40}^{B_3}$ est 15 fois plus importante que la médiane de $F_{40}^{B_2}$.

Le même constat peut être fait en comparant $F_x^{B_2}$ avec $F_x^{B_1}$ (à l'exception de $F_{20}^{B_2}$ et $F_{20}^{B_1}$

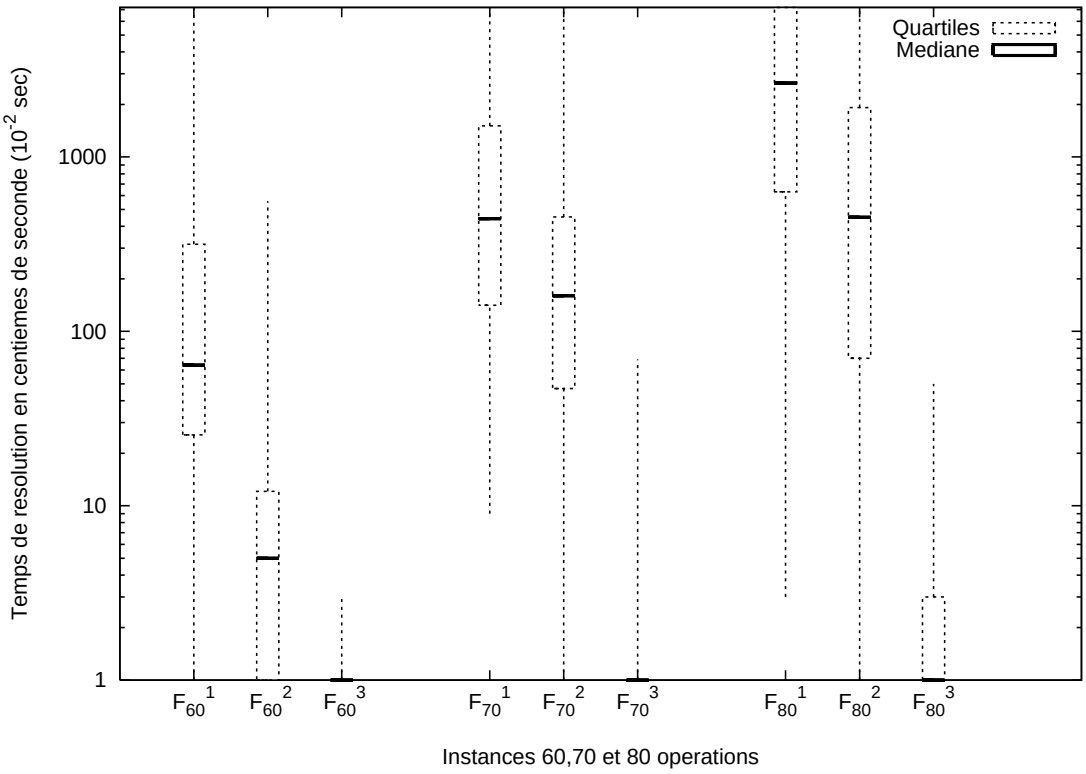


FIG. 6.14 – Temps de calcul en fonction OS pour les familles de 60, 70 et 80 opérations

car dans les deux cas la résolution des instances ne dépassent pas une seconde).

Nous observons le même phénomène pour les familles de 50, 60 et 70 opérations. Par ailleurs, nous constatons que les limites du modèle sont atteintes pour la famille $F_{70}^{B_3}$ dont le nombre moyen de blocs disponibles est de 210. La médiane de la sous-famille $F_{70}^{B_3}$ est égale au temps alloué à Cplex, ce qui signifie qu'il y a au moins 50 % des instances pour lesquelles il n'a pas été fourni une preuve de l'optimalité (les meilleures solutions trouvées peuvent même être juste réalisables). Nous observons un autre cas de dépassement du temps alloué lorsque le nombre de blocs est égal à 180 et le nombre d'opérations est à 60 ($F_{60}^{B_3}$). Dans ce cas, plus d'un quart des instances de la sous-famille $F_{60}^{B_3}$ n'ont pu être résolues à l'optimum, car le $3^{\text{ème}}$ quartile coincide avec le maximum de temps accordé (7200 sec).

6.4.4 Étude de l'influence du nombre d'opérations par bloc

Cette sous-section est réservée à l'étude de l'impact du nombre moyen d'opérations par bloc sur le temps de calcul. Afin de réaliser cette étude nous avons généré une autre série d'instances. Pour chaque famille F_x tel que $x \in \{20, 30, 40, 50, 60, 70\}$ nous avons généré trois sous-familles, notée comme suit :

$F_x^{O_1}$, $F_x^{O_2}$ et $F_x^{O_3}$. Les sous-familles de type $F_x^{O_1}$ ont l'intervalle $OpI = [OpMin, OpMax]$

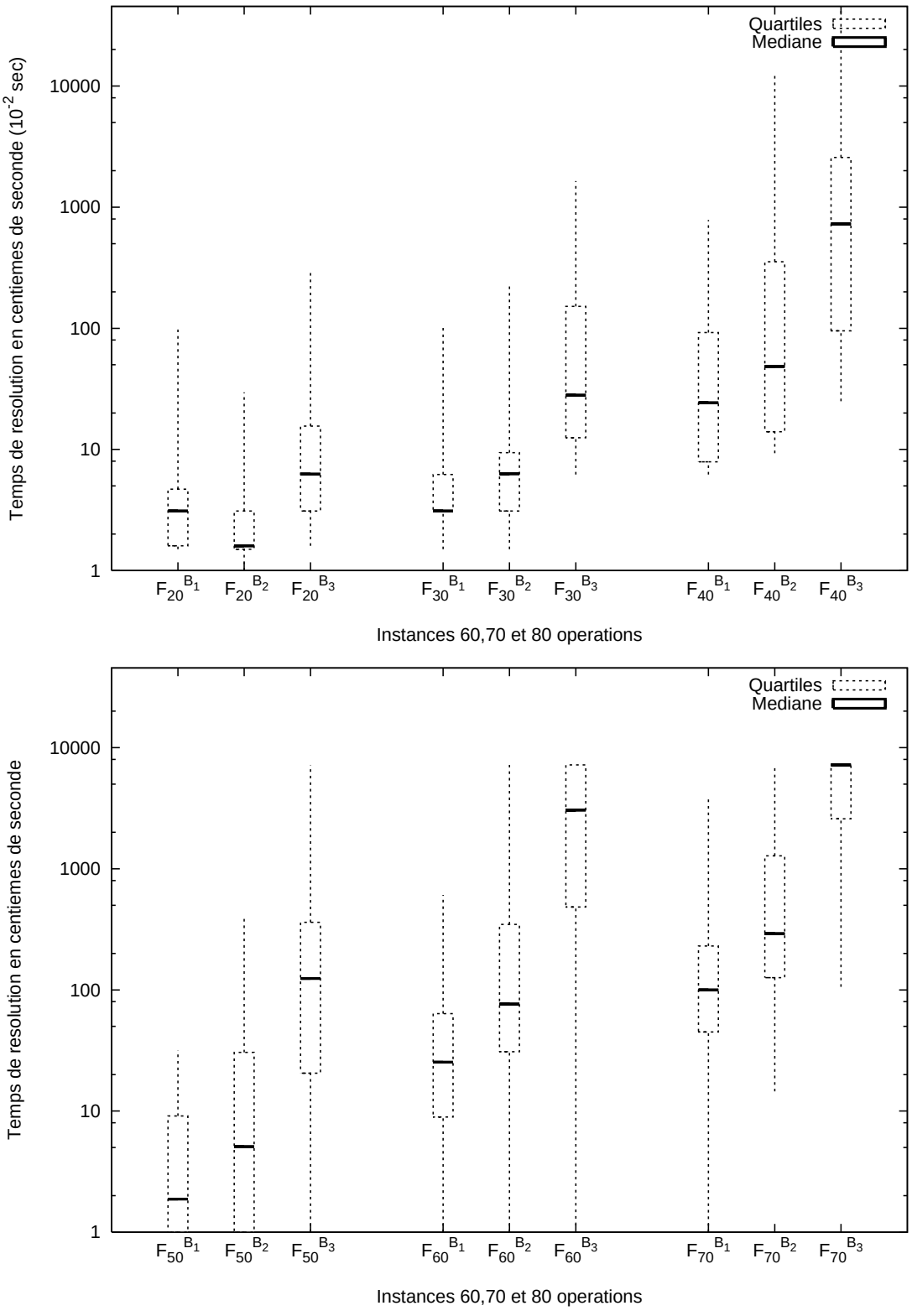


FIG. 6.15 – Temps de calcul pour l'étude de l'influence de $|\mathbf{B}|$

$F_x^{B_i}$	$\|\mathbf{N}\|$	$\|\mathbf{B}\|$	OSI	OS_{avg}	m_0	$F_x^{B_i}$	$\|\mathbf{N}\|$	$\|\mathbf{B}\|$	OSI	OS_{avg}	m_0
$F_{20}^{B_1}$		30				$F_{50}^{B_1}$		75			
$F_{20}^{B_2}$	20	40	$[8, 15]$	12	13	$F_{50}^{B_2}$	50	100	$[17, 31]$	24	26
$F_{20}^{B_3}$		60				$F_{50}^{B_3}$		150			
$F_{30}^{B_1}$		45				$F_{60}^{B_1}$		90			
$F_{30}^{B_2}$	30	60	$[15, 32]$	22	16	$F_{60}^{B_2}$	60	120	$[9, 18]$	13	30
$F_{30}^{B_3}$		90				$F_{60}^{B_3}$		180			
$F_{40}^{B_1}$		60				$F_{70}^{B_1}$		105			
$F_{40}^{B_2}$	40	80	$[15, 29]$	20	21	$F_{70}^{B_2}$	70	140	$[12, 19]$	15	34
$F_{40}^{B_3}$		120				$F_{70}^{B_3}$		210			

TAB. 6.4 – Description des instances pour étudier l'influence de $|\mathbf{B}|$

égal à [1,2], ce qui correspond aux instances dont les blocs contiennent une ou deux opérations. Les sous-familles de type $F_x^{O_2}$ correspondent à l'intervalle OpI =[2,5]. Les sous-familles $F_x^{O_3}$ sont les instances dans lesquelles les blocs peuvent contenir de 3 à 7 opérations, l'intervalle $OpI = [3, 7]$. Nous rapportons dans le tableau 6.5 l'ensemble des caractéristiques des instances qui ont été générées.

$F_x^{O_i}$	$\|\mathbf{N}\|$	OpI	$Avg\|\mathbf{B}\|$	m_0	$F_x^{O_i}$	$\|\mathbf{N}\|$	OpI	$Avg\|\mathbf{B}\|$	m_0
$F_{20}^{O_1}$		[1,2]			$F_{50}^{O_1}$		[1,2]		
$F_{20}^{O_2}$	20	[2,5]	32	13	$F_{50}^{O_2}$	50	[2,5]	95	26
$F_{20}^{O_3}$		[3,7]			$F_{50}^{O_3}$		[3,7]		
$F_{30}^{O_1}$		[1,2]			$F_{60}^{O_1}$		[1,2]		
$F_{30}^{O_2}$	30	[2,5]	45	16	$F_{60}^{O_2}$	60	[2,5]	110	30
$F_{30}^{O_3}$		[3,7]			$F_{60}^{O_3}$		[3,7]		
$F_{40}^{O_1}$		[1,2]			$F_{70}^{O_1}$		[1,2]		
$F_{40}^{O_2}$	40	[2,5]	60	21	$F_{70}^{O_2}$	70	[2,5]	130	34
$F_{40}^{O_3}$		[3,7]			$F_{70}^{O_3}$		[3,7]		

TAB. 6.5 – Description des instances pour l'étude de l'influence de OpI

La figure 6.16 montre les résultats de la résolution de ces instances. Les sous-familles de 20 opérations sont du même ordre de difficulté, la plupart des instances de cette famille sont solvables en moins de 3 centièmes de seconde. Par contre, une différence significative pour les familles de 30 et 40 opérations est observée. En effet, pour chacune de ces familles, nous observons une nette diminution de la médiane au fur et à mesure que le nombre d'opérations par bloc augmente. Ce résultat s'explique par le fait qu'un nombre élevé d'opérations par bloc engendre des solutions réalisables comportant moins de blocs. Ainsi, lors de la construction

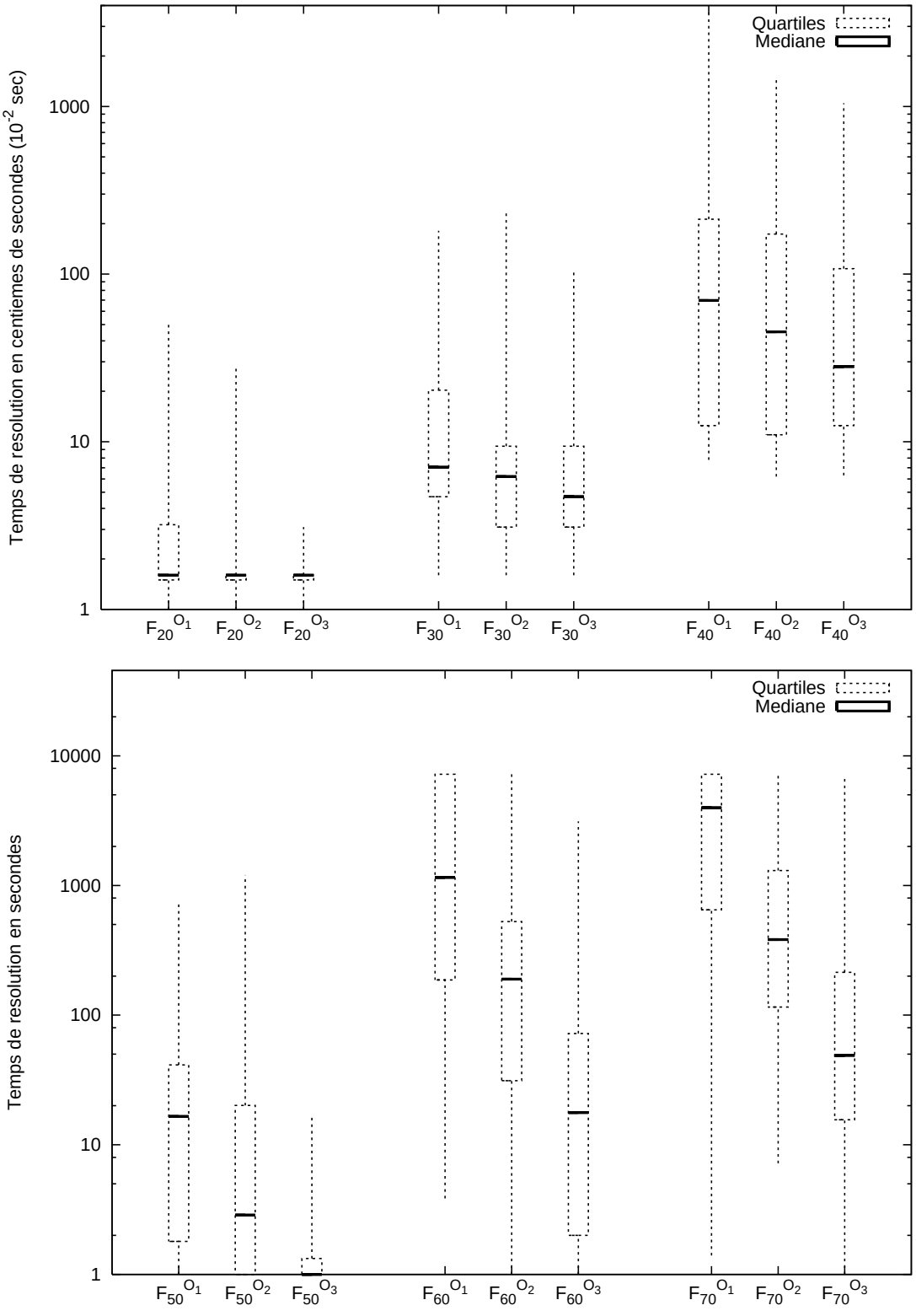


FIG. 6.16 – Temps de calcul en fonction de OpI

de l'arbre d'énumération dans la PSE utilisée Cplex les feuilles sont atteintes plus rapidement ce qui a pour effet de réduire le nombre total de branchements.

Il arrive toutefois qu'une instance ayant un nombre d'opérations par bloc élevé soit plus difficile à résoudre qu'une autre instance ayant les mêmes valeurs pour les autres paramètres mais avec un nombre d'opérations par blocs moins grand. Cette situation est constatée pour les instances les plus difficiles des sous-familles $F_{30}^{O_1}$ et $F_{30}^{O_2}$ (voir les points extrêmes hauts des candlesticks de $F_{30}^{O_1}$ et $F_{30}^{O_2}$). En effet, un jeu de données d'une instance peuvent la rendre difficile à résoudre et ce malgré son appartenance à une sous-famille, *a priori*, facile.

6.4.5 Étude de l'effet du nombre maximum de stations

Le nombre de stations maximum m_0 avec le nombre de blocs $|\mathbf{B}|$ déterminent le nombre de variables binaires x_{bk}. Ce paramètre a donc, *a priori*, une influence directe sur les performances des modèles. Afin de s'en assurer, nous avons généré une autre série de tests dans laquelle nous avons fait varier le nombre maximum de stations. Nous rapportons dans le tableau 6.6 les paramètres utilisés pour la génération de ces sous-familles $F_x^{m_i}$, $i = 1, 2, 3$ et $x \in \{20, 30, 40, 50, 60, 70\}$.

$F_x^{m_i}$	$\lvert\mathbf{N}\rvert$	m_0	$Avg\lvert\mathbf{B}\rvert$	OsI	$F_x^{m_i}$	$\lvert\mathbf{N}\rvert$	m_0	$Avg\lvert\mathbf{B}\rvert$	OsI
$F_{20}^{m_1}$		7			$F_{50}^{m_1}$		12		
$F_{20}^{m_2}$	20	10	32	$[8, 15]$	$F_{50}^{m_2}$	50	19	95	$[17, 31]$
$F_{20}^{m_3}$		13			$F_{50}^{m_3}$		26		
$F_{30}^{m_1}$		10			$F_{60}^{m_1}$		14		
$F_{30}^{m_2}$	30	13	45	$[15, 32]$	$F_{60}^{m_2}$	60	20	110	$[9, 18]$
$F_{30}^{m_3}$		16			$F_{60}^{m_3}$		30		
$F_{40}^{m_1}$		12			$F_{70}^{m_1}$		14		
$F_{40}^{m_2}$	40	17	60	$[15, 29]$	$F_{70}^{m_2}$	70	20	130	$[12, 19]$
$F_{40}^{m_3}$		21			$F_{70}^{m_3}$		34		

TAB. 6.6 – Description des instances générées pour l'étude de l'influence de m_0

Nous rapportons dans les figures 6.17 et 6.18 le temps de calcul pour les instances décrites dans le tableau 6.6. Les résultats confirment que ce modèle est plus performant pour les instances appartenant à des sous-familles dont la valeur de m_0 est faible. Les instances de 70 opérations marquent le plus grand écart. En effet, l'instance la plus difficile de $F_{70}^{m_1}$ nécessite à peine 24 sec contre plus de 7200 sec pour celle de $F_{70}^{m_3}$.

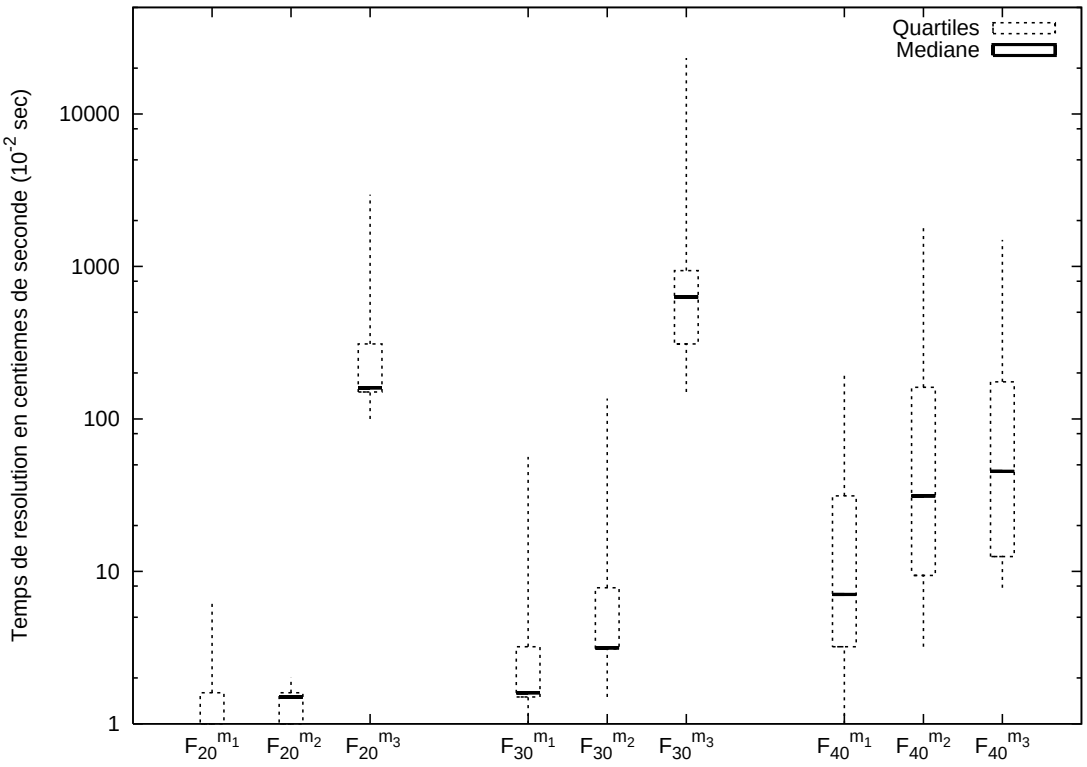


FIG. 6.17 – Influence de m_0 pour les familles de 20, 30 et 40 opérations

6.5 Comparaison avec une approche graphe

Dans cette sous-section, nous présentons l'étude comparative des performances entre le modèle orienté opérations et l'approche graphe qui a été proposée dans [DGL06] (voir section 3.6.1).

Pour cette étude, nous utilisons les familles d'instances décrites dans la sous-section 6.4.2. Nous rapportons les résultats obtenus dans les figures 6.19. Nous indiquons, pour chaque sous-famille F_x^y, le temps de calcul de la méthode graphe par *GRP* et celui du modèle linéaire orienté opérations par *MO*.

Ces figures permettent de constater une nette domination du modèle linéaire pour toutes les familles sauf pour les instances les plus difficiles des sous-familles F_x^3 (notons que dans ce cas le temps de résolution avec le modèle linéaire reste faible et proche de celui obtenu avec la méthode graphe). Le gain le plus intéressant est obtenu avec le modèle linéaire pour les familles les plus difficiles à résoudre c'est-à-dire les F_x^1.

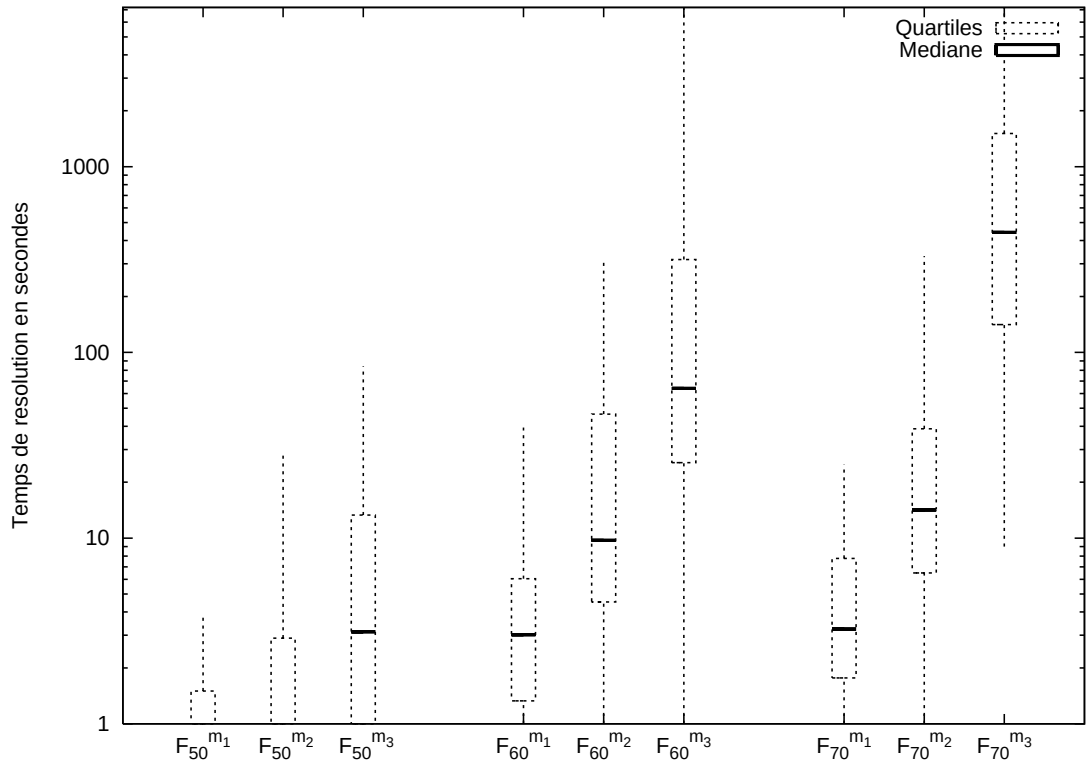


FIG. 6.18 – Influence de m_0 pour les familles de 50, 60 et 70 opérations

6.6 Comparaison avec une PSE

À présent, nous nous intéressons à la comparaison avec la PSE dédiée qui a été initialement proposée dans [DI05] pour la résolution du problème TLBP/B-M (voir section 3.6.2). Ce problème est une généralisation du TLBP/B-P, la procédure peut donc être appliquée dans le cadre du TLBP/B-P. Nous reprenons dans les figures 6.20, les résultats des expérimentations entreprises pour les instances décrites dans le tableau 6.3.

Dans les graphiques 6.20, les temps de calcul moyens obtenus avec la PSE sont notés *B&B* et ceux obtenus avec notre modèle linéaire orienté opérations sont marqués *CPLEX*. Rappelons que les familles de type F_x^1 ont une faible densité du graphe de précédence, il a été montré qu'elles sont les plus difficiles à résoudre (voir sous-section 6.4.2). Nous constatons pour ces familles que le modèle linéaire orienté opérations est souvent plus performant que la méthode dédiée, le gain le plus important est marqué pour certaines sous-familles de 60 et 70 opérations où Cplex permet de réduire le temps de calcul moyen par plus de la moitié. Concernant les sous-familles de type F_x^2 ayant une moyenne densité, le modèle linéaire présente des résultats équivalent à ceux de la PSE dédiée, néanmoins il peut être moins performant lorsque le nombre d'opérations devient trop important (70 et 80).

Afin de compléter l'analyse comparative des deux méthodes, nous présentons dans les figures 6.21 les temps minimum et maximum des instances. Nous constatons globalement

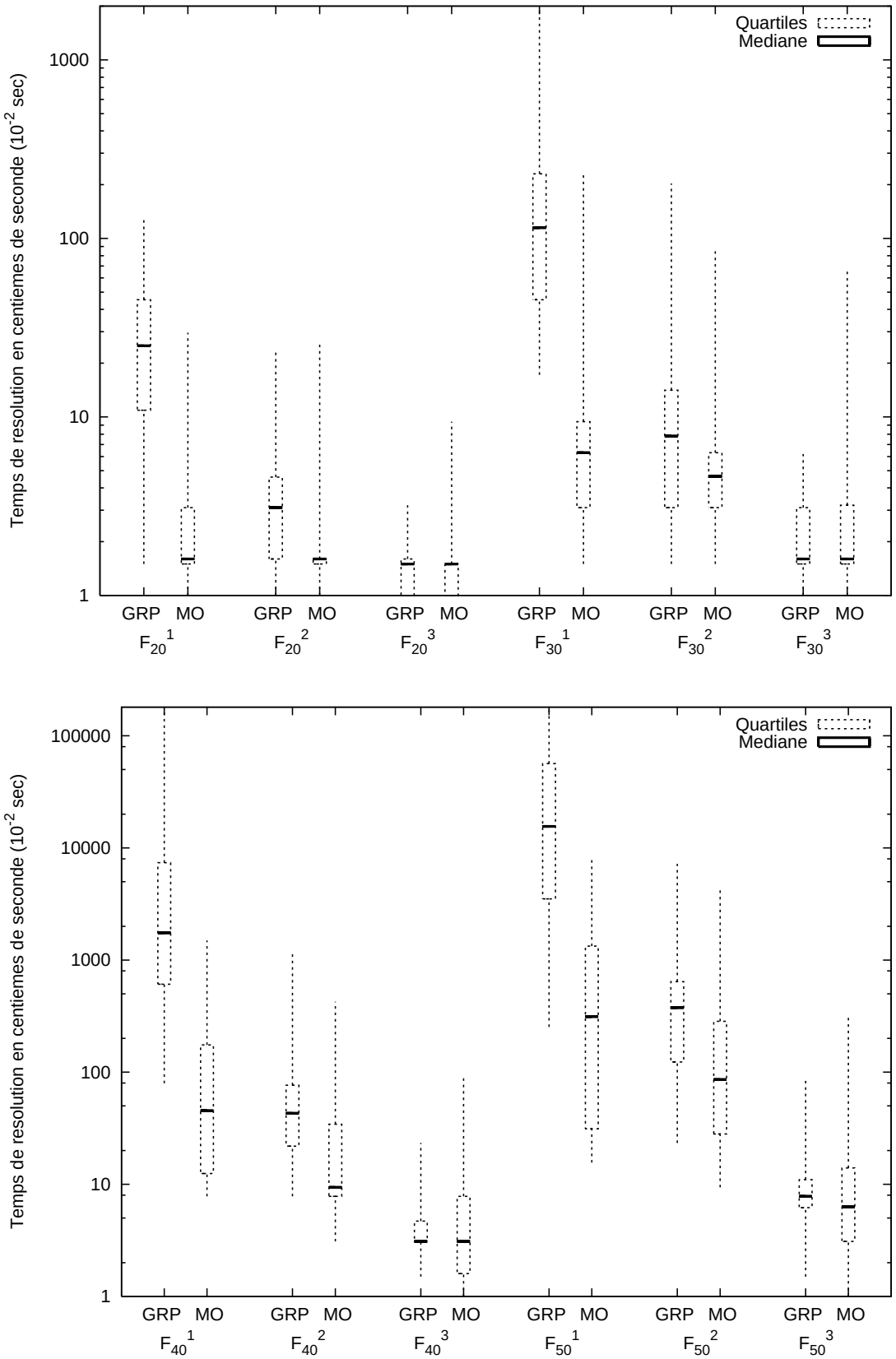


FIG. 6.19 – Temps de calcul de la méthode graphe et du modèle orienté opérations

le même comportement que pour les moyennes, c'est-à-dire que le modèle linéaire est plus performant sur les instances difficiles (F_x^1). Alors que la PSE dédiée est meilleure sur les instances moyennes F_x^2.

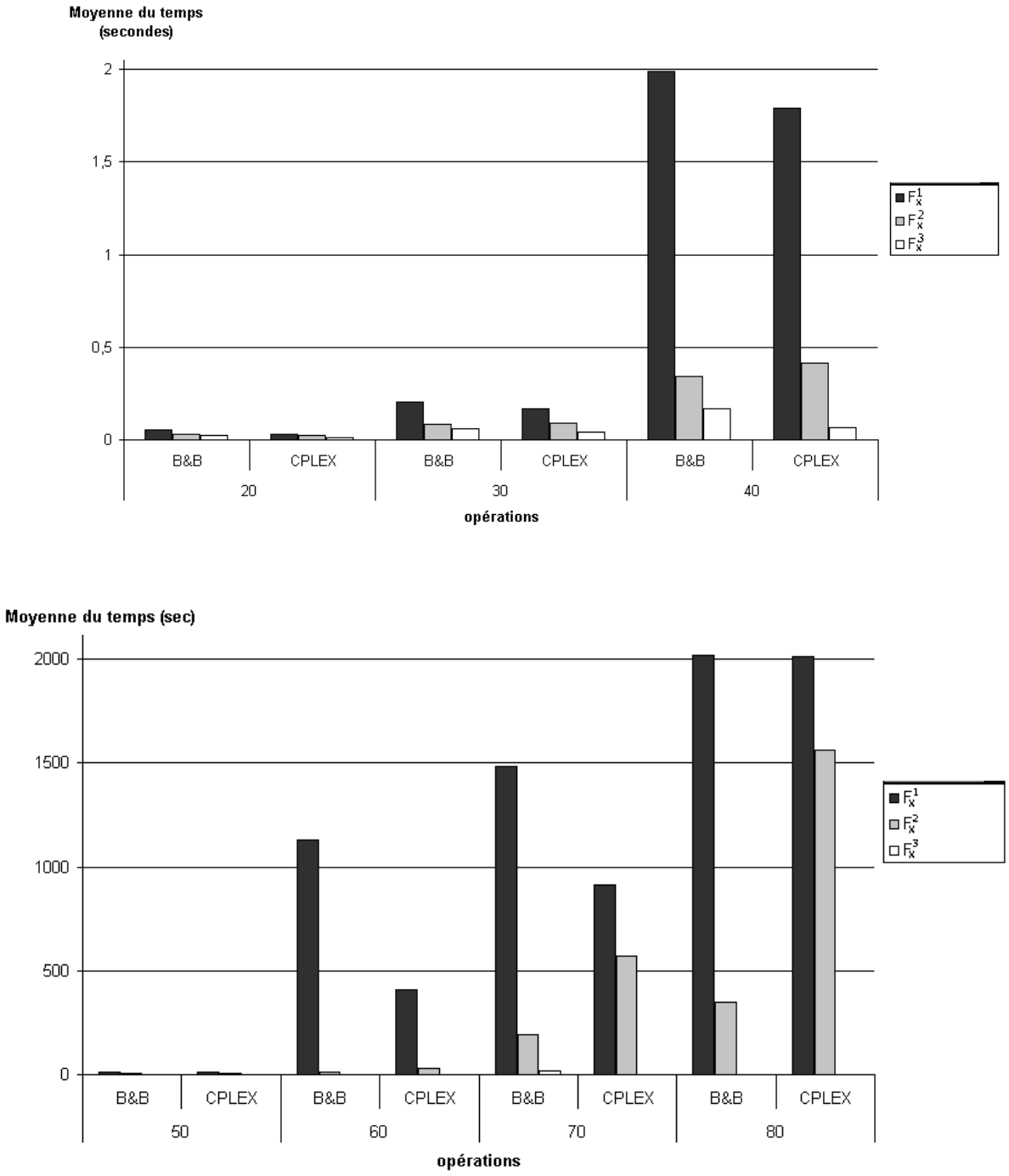


FIG. 6.20 – Temps moyens de la PSE et du modèle linéaire pour les familles de 20 à 80 opérations

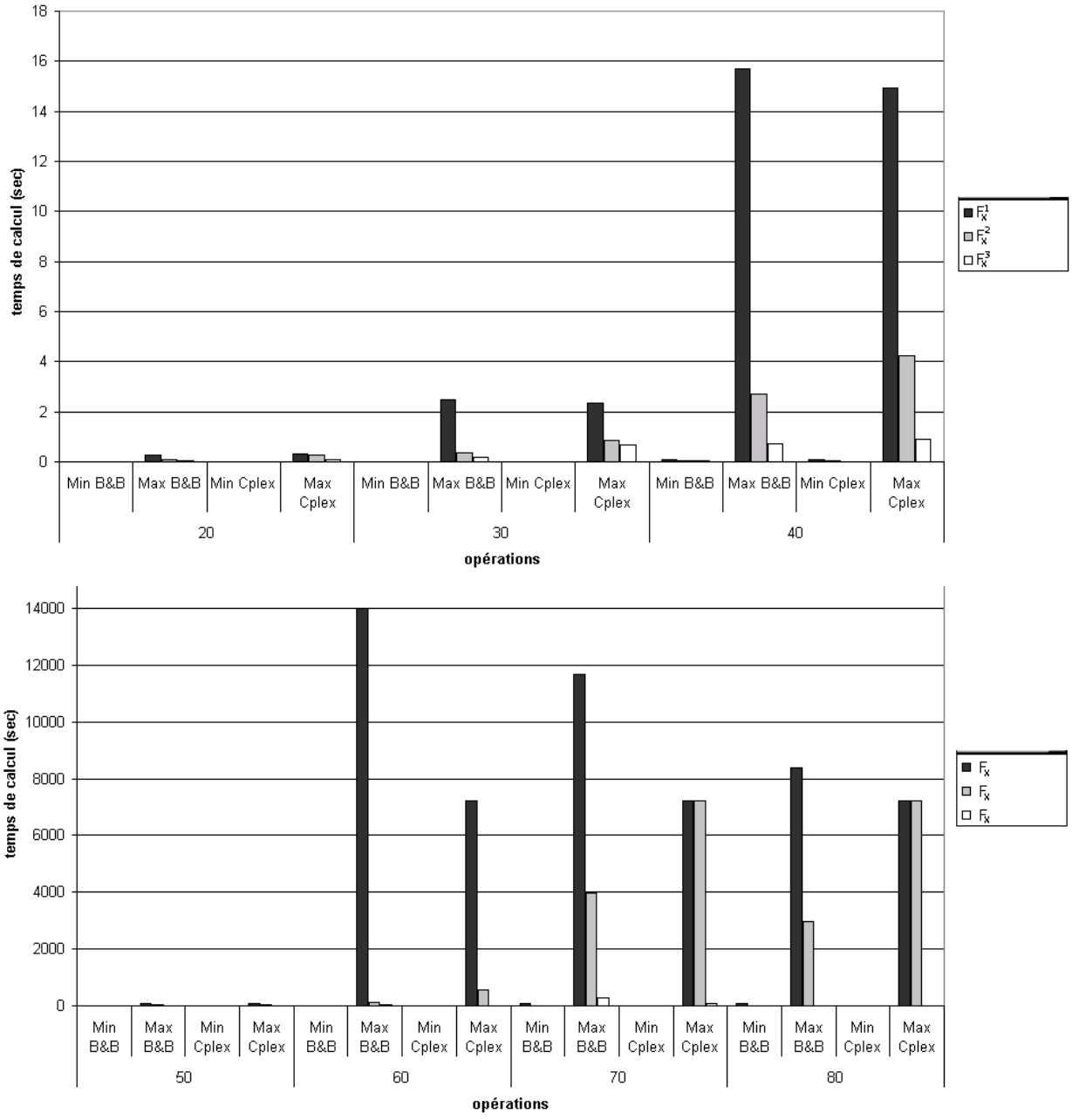


FIG. 6.21 – Temps de calcul de la PSE et du modèle orienté opérations

6.7 Conclusion

Dans ce chapitre, nous avons effectué de nombreux tests (près de 4000) pour évaluer les différents modèles proposés dans le chapitre précédent mais aussi pour mesurer l'apport des algorithmes de réduction du nombre de variables binaires. Les tests ont montré que les modèles de programmation linéaire en nombres entiers sont plus performants que les modèles de programmation par contraintes. Ils nous ont également permis de constater que les algorithmes de réduction du nombre de variables binaires permettaient de diminuer de façon significative le temps de calcul des instances.

Ensuite, nous avons comparé les performances des deux modèles linéaires en montrant que le modèle orienté opérations était plus efficace que le modèle orienté blocs. De plus, ces résultats confirment que l'utilisation d'outils tels que Cplex doit être accompagnée d'une bonne modélisation pour arriver à résoudre des instances difficiles. Ainsi, quand le modèle orienté blocs atteint ses limites pour des instances de 40 opérations, le modèle orienté opérations, permet de résoudre des problèmes ayant une taille de 80 opérations, doublant ainsi la taille des instances traitées. Ceci atteste de l'importance et du gain apporté par une modélisation exploitant la structure du problème. Cette phase d'expérimentation confirme ainsi qu'une coopération entre un modèle optimisé et un solver de qualité (utilisant ILOG Cplex) peut être aussi performante qu'une méthode dédiée telle que la méthode des graphes et la PSE.

Par ailleurs, nous avons également étudié l'influence de plusieurs paramètres tels que : le nombre de blocs, densité du graphe de précédence. L'étude a permis de déceler une corrélation entre le temps de calcul et les valeurs de ces paramètres.

Troisième partie

Quelques extensions du modèle TLBP/B-P

Chapitre 7

Mode d'activation mixte des unités d'usinage : TLBP/B-M

À présent, nous nous intéressons à la généralisation de notre approche au problème considérant un mode d'activation mixte dans le sens série-parallèle (voir formulation générique du problème dans [DIB05]). Dans ce type de lignes, une station correspond à plusieurs étages dont le fonctionnement est séquentiel de sorte que l'étage suivant ne peut être enclenché avant que le précédent n'ait terminé. Chaque étage peut contenir lui-même un sous-ensemble d'unités d'usinage qui sont activées en parallèle. Ainsi, lorsqu'il n'existe qu'un seul étage dans une station le problème se ramène au cas de l'activation parallèle, que nous avons étudié dans la seconde partie de ce mémoire. Par ailleurs, en présence d'une seule unité d'usinage par étage, le problème se ramène au cas séquentiel. Le cas mixte (TLBP/B-M) est donc une généralisation du cas parallèle (TLBP/B-P) et du cas séquentiel.

7.1 Description du TLBP/B-M

7.1.1 Les données

L'ensemble des opérations est également noté $\mathbf{N}$ pour désigner les opérations qui doivent être effectuées sur la ligne.

L'ensemble $\mathbf{B}$ est utilisé de la même manière que précédemment pour désigner les blocs disponibles. Ainsi, $\forall b \in \mathbf{B}, \mathcal{N}(b) \subseteq \mathbf{N}$ tel qu'il existe une unité d'usinage effectuant toutes les opérations de b. De même, chaque bloc b est caractérisé par son coût unitaire q_b et son temps d'exécution t_b.

Le coût moyen de l'ouverture d'une station est toujours noté C et chaque station peut contenir au maximum q_0 étages. Il est important de noter que le coût d'ouverture d'une station ne dépend pas du nombre d'étages qui seront effectivement utilisés.

La présence des étages activés de manière séquentielle nous amène à récrire les contraintes du problème en soulignant, à chaque fois, les différences avec le cas parallèle (TLBP/B-P).