Chapitre 3

Conception d'un algorithme génétique parallèle pour le problème d'ordonnancement sur machine unique avec temps de réglage dépendant de la séquence

3.1 Description du problème d'ordonnancement sur machine unique avec temps de réglage dépendant de la séquence

Dans certains contextes industriels, les unités de production sont bâties autour d'une seule ressource avec des traitements successifs. On peut citer, par exemple, des domaines comme la métallurgie, la coloration du plastique ou encore la fabrication de papier. Ce cas de figure implique qu'une seule commande (job) peut être traitée à la fois et, qu'une fois lancée, la commande ne peut être interrompue. Une autre particularité réside dans les temps de réglages entre deux commandes pendant lesquels la ressource est inutilisable.

L'utilisation de la ressource est régie par un carnet de commandes. Chaque commande est identifiée par un numéro (de 1 à N), un temps de traitement, une date de livraison, un temps de réglage initial et des temps de réglage relatifs aux autres commandes. L'ordonnancement vise à déterminer la séquence de production permettant d'optimiser un objectif particulier. Dans ce mémoire, nous nous intéressons à la minimisation du retard total qui s'exprime par la somme des retards de chaque commande de la manière suivante :

$$T_Q = \sum_{k=1}^{N} t_{Q(j)} \tag{3.1}$$

où $t_{Q(j)}$ est le maximum entre 0 et la différence entre la date de fin de traitement ($c_{Q(j)}$) et la date de livraison ($d_{Q(j)}$) de la commande j selon l'équation suivante :

$$t_{Q(j)} = max\{0, c_{Q(j)} - d_{Q(j)}\} \tag{3.2}$$

Parmi les principaux travaux ayant traités de cette problématique, on cite ceux de Rubin et Ragatz [134], Gagné *et al.* [59] [60], Koulamas [90], Ragatz [128], Tan *et al.* [146], Lee *et al.* [93] [147], França *et al.* [58], Luo *et al.* [97] [96] ainsi que Gupta *et al.* [78]. Une classification de tous les problèmes d'ordonnancement ayant ou non une dépendance avec la séquence a été proposée par Allahverdi *et al.* [11]. Dans la prochaine section, nous nous intéressons plus particulièrement aux travaux de Rubin et Ragatz [134] qui ont utilisé un algorithme génétique pour solutionner ce problème.

3.2 L'algorithme génétique de Rubin et Ragatz

Rubin et Ragatz [134] ont abordé le problème de l'ordonnancement sur machine unique avec temps de réglage dépendant de la séquence avec un algorithme génétique classique tel que décrit à la Figure 2.15. La fonction objectif adoptée est la minimiation du retard total. Les caractéristiques de cet algorithme sont décrites dans les paragraphes qui suivent.

Premièrement, la taille de la population a été fixée à 40. Selon les auteurs, une population de 20 donnait de mauvaises performances alors qu'une population de 60 ou encore 80 n'amenait aucune amélioration par rapport au taux de convergence. De plus, comme le temps de calcul augmente de façon linéaire avec la taille de population, il n'est pas bénéfique de trop augmenter la taille de la population.

La méthode de sélection adoptée par les auteurs est celle par roulette. La probabilité de sélection d'un individu est inversement proportionnelle à sa fonction objectif. Une fonction de transformation a aussi été implémentée pour éviter les divisions par 0.

L'opérateur de croisement implémenté par Rubin et Ragatz est particulier et ne correspond à aucun de ceux décrits à la Section 2.2.2.1. Aucune explication n'a été fournie par les auteurs sur le choix de cet opérateur par rapport à un autre de la littérature. Toutefois, ils

ont mentionné avoir préféré cette codification par rapport à la codification binaire pour éviter d'avoir des vecteurs trop volumineux. Un exemple de cet opérateur de croisement est présenté à la Figure 3.1. Tout d'abord, deux groupes de commandes sont formés. À la Figure 3.1 partie (a), le groupe g1 est celui des commandes impaires et le groupe g2 est celui des commandes paires. Le choix des groupes est aléatoire mais il est préférable que les deux groupes soient de même taille. L'ordre selon lequel chaque groupe apparaît dans chacun des parents est noté dans la partie (b). Toutes les combinaisons sont rangées en colonnes pour une meilleure visibilité. Ensuite, les positions auxquelles apparaissent les éléments des deux groupes sont notées dans la partie (c) du schéma. Ces positions sont représentées par des lettres pour éviter la confusion avec les numéros des commandes. Enfin, pour constituer les enfants, on effectue la combinaison des différents ordres et positions. Par exemple, l'enfant E1 est constitué par les éléments de g1 dans l'ordre qu'ils apparaissent dans P1 aux positions issues de P1. Il est aussi constitué du groupe g2 dans l'ordre de P1 aux positions de P1. Par conséquent, l'enfant E1 va être un clone de P1. Les différentes combinaisons donnent au total huit enfants dont deux clones des parents.

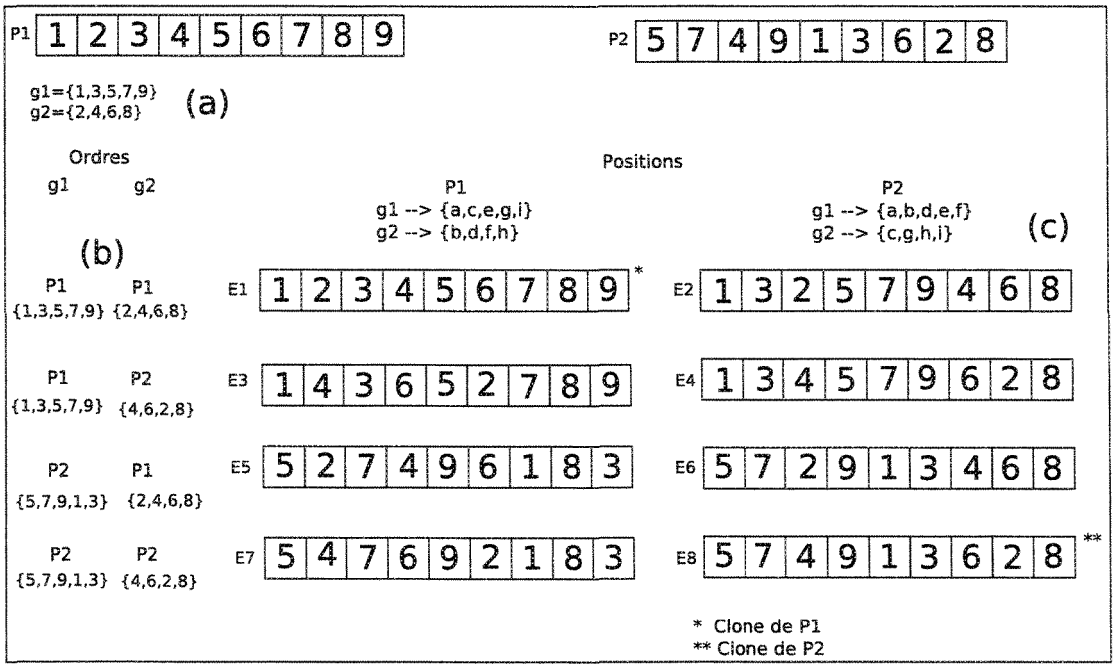

Figure 3.1 – Opérateur de croisement Rubin & Ragatz [134]

La mutation a aussi été abordée par Rubin et Ragatz de manière particulière. L'opérateur de mutation ne correspond à aucun de ceux vus à la Section 2.2.2.1. En effet, c'est une recherche locale qui consiste à permuter toutes les positions adjacentes et conserver une permutation seulement si elle améliore la fonction fitness de l'individu. Ce genre de mutation est beaucoup plus lent à l'exécution que la version traditionnelle mais, selon les auteurs, assure une amélioration de la majorité des individus mutés. Le taux de mutation a été fixé à 10% à chaque génération.

Enfin, la phase de remplacement a été abordée de manière élitiste et la gestion de la diversité s'est faite par un procédé nommé *immigration* et qui consiste à la régénération aléatoire des 10% pires individus à chaque génération.

Nous avons, dans un premier temps, tenté de reproduire l'algorithme génétique proposé par ces auteurs. Cet algorithme, noté AG0 dans la suite du document, a été évalué sur un ensemble de problèmes tests proposés par Ragatz [127] pour valider son fonctionnement. Ces problèmes ont spécialement été générés pour l'ordonnancement sur machine unique avec temps de réglage dépendant de la séquence avec l'objectif de minimiser le retard total. Les problèmes tests se composent de quatre groupes selon le nombre de commandes qu'ils contiennent. Ces groupes sont respectivement de 15, 25, 35 et 45 commandes. Pour chaque taille de problème, huit problèmes sont générés en faisant varier trois paramètres. Le premier paramètre, "Processing Time Variance" (PTV), définit la variabilité du temps d'exécution et peut prendre deux valeurs "low" et "high". Le deuxième paramètre, "Tardiness Factor" (TF), correspond au degré de retard dans les problèmes générés et peut prendre deux valeurs "low" et "medium". Enfin, le troisième paramètre, "Due Date Range" (DDR), correspond à la variabilité dans les dates d'échéance et peut être "narrow" ou "wide".

Rubin et Ragatz ont fixé le nombre total de générations à 300 ou jusqu'à ce que les résultats du Branch & Bound soient atteints ou améliorés. En effet, les auteurs ont conçu une procédure de type Branch & Bound limitée à l'exploration de deux millions de noeuds. Ces solutions servent de base de comparaison pour la performance de l'algorithme génétique.

Le Tableau 3.1 présente les écarts absolus avec la soluion de référence issus de l'article

de Rubin et Ragatz [134] et ceux obtenus par l'AG0 pour 300 et 1000 générations. Tout comme Rubin et Ragatz, les résultats de l'AG0 représentent une moyenne de 20 essais. En terme de qualité, trois mesures sont prises en compte : le meilleur résultat obtenu (Meilleur), le résultat médian (Médian) et le pire résultat obtenu (Pire). Pour la comparaison de performance entre l'AG0 (300 générations) et l'algorithme génétique de Rubin & Ragatz, on observe, en utilisant un écart maximal de 5% entre les résultats médians, une performance identique sur 25 des 32 problèmes. Pour les 7 autres problèmes dont les résultats sont présentés en caractères gras, l'AG0 est moins performant, ce qui laisse croire que certains éléments diffèrent entre les deux algorithmes. L'augmentation du nombre de générations à 1000 corrige cette situation et fait même en sorte que les résultats sont améliorés pour 5 problèmes. Ces problèmes sont indiqués en caractères gras dans le Tableau 3.1. On trouve, en moyenne, la meilleure solution, après 830 générations, ce qui explique le nombre maximal de générations fixé à 1000.

À la section suivante, nous décrivons le fonctionnement d'un autre algorithme génétique ayant l'algorithme AG0 comme point de départ, mais utilisant des opérateurs de croisement classiques ainsi que des paramètres différents. Une analyse comparative de la performance sera réalisée avec l'AG0.

3.3 Conception d'un algorithme génétique séquentiel

L'algorithme génétique présenté dans cette section utilise des opérateurs génétiques classiques et a été conçu en suivant des indications retrouvées dans la littérature pour la fixation des différents paramètres. Cet algorithme est noté AG1.

La taille de la population est fixée égale au nombre de commandes du problème et le nombre de générations est fixé à 1000 selon l'étude de convergence réalisée. Des essais numériques, faisant varier le taux de mutation entre 10 et 20 %, ont également permis de fixer le taux de mutation à 19%. Cette mutation est identique à celle de l'AG0 et utilise le principe de la recherche locale introduit par Rubin et Ragatz [134].

L'opérateur de croisement a été défini après l'expérimentation des opérateurs suivants : MPX, OX, LOX et CX. Comme aucun de ces opérateurs ne s'est particulièrement distingué,

Prb	# comm.	B&B	*	GA (Rubin & Ragatz) Meilleur	Médian	Pire	AG0(300) Meilleur	Médian	Pire	AG0(1000) Meilleur	Médian	Pire
PROB401	15	90	*	0,0	4,4	4,4	0,0	6,7	27,8	4,4	4,4	21,1
PROB402	15	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB403	15	3418	*	0,0	0,0	0,0	0,0	1,1	2,3	0,0	2,1	3,7
PROB404	15	1067	*	0,0	0,0	0,0	0,0	0,0	6,8	0,0	0,0	0,0
PROB405	15	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB406	15	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB407	15	1861	*	0,0	0,0	0,0	0,0	0,7	2,1	0,0	0,7	2,7
PROB408	15	5660	*	0,0	0,0	0,9	0,0	0,0	1,5	0,0	1,3	2,5
PROB501	25	264		0,0	1,5	3,8	1,1	7,2	12,1	2,3	4,2	8,3
PROB502	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB503	25	3511		-0,4	0,2	0,9	-0,1	1,9	3,5	-0,1	0,4	4,9
PROB504	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB505	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB506	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB507	25	7225	*	2,1	6,1	9,6	0,4	4,9	9,3	1,2	3,0	5,3
PROB508	25	2067		-5,9	-5,9	-1,5	-7,4	-1,9	15,6	-7,4	-1,7	5,2
PROB601	35	30		76,7	150,0	193,3	73,3	**243,3**	343,3	53,3	**140,0**	236,7
PROB602	35	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB603	35	17774		-0,7	0,4	2,2	1,8	**5,8**	7,9	0,6	2,9	5,6
PROB604	35	19277		0,2	1,0	2,6	1,4	5,8	11,4	0,3	2,7	4,5
PROB605	35	291		13,7	37,3	56,7	11,7	**54,0**	97,3	5,5	**25,8**	45,4
PROB606	35	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB607	35	13274		5,0	6,6	7,6	0,9	6,4	8,8	1,1	2,5	3,7
PROB608	35	6704		-29,0	-28,6	-26,7	-28,2	**-11,6**	-4,0	-28,0	**-23,3**	-14,4
PROB701	45	116		57,8	82,8	118,1	85,3	**131,9**	175,9	59,5	80,2	136,2
PROB702	45	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB703	45	27097		-0,1	1,5	2,5	3,3	5,9	9,1	1,0	3,2	4,6
PROB704	45	15941		-2,4	-1,6	1,0	3,9	**12,1**	22,3	-0,8	3,8	9,1
PROB705	45	234		53,4	89,3	114,5	69,7	**95,7**	145,7	48,3	**69,2**	91,5
PROB706	45	0	*	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
PROB707	45	25070		-1,1	1,8	6,3	0,3	5,4	8,8	-2,3	1,3	5,0
PROB708	45	24123		2,8	7,2	10,1	-1,4	6,6	11,8	-2,2	**0,5**	3,4

* Solution Optimale

Tableau 3.1 – Résultats AG0

l'AG1 utilise aléatoirement, à chaque génération, l'un de ces 4 opérateurs pour garantir un maximum de diversité dans le processus de reproduction. Cette combinaison d'opérateurs a produit les meilleurs résultats.

Pour les essais numériques avec l'AG1, quatre autres groupes de problèmes de tailles respectives 55, 65, 75 et 85 commandes ont été ajoutés. Ces problèmes tests ont été proposés par Gagné *et al.* [59] et vont permettre d'analyser le comportement de l'algorithme sur des problèmes de plus grande taille et ainsi réaliser une meilleure comparaison entre l'AG0 et

l'AG1. Le Tableau 3.2 présente les résultats obtenus sur les 64 problèmes.

On observe, en utilisant un écart maximal de 5% entre les résultats médians, une meilleure performance de l'AG0 par rapport à l'AG1 sur 4 problèmes pour les groupes de taille 15 à 45 commandes. On constate toutefois que AG1 s'avère plus performant sur 6 problèmes lorsque l'on considère les groupes de taille 55 à 85 commandes tout en donnant un résultat identique pour les 26 autres problèmes. On note plus particulièrement que les gains de performance sont très importants sur le problème PROB655. Il y a également eu des améliorations sur les meilleurs résultats obtenus pour 4 problèmes (PROB651, PROB655, PROB658 et PROB751) et une amélioration des pires résultats pour 7 problèmes (PROB551, PROB651, PROB655, PROB658, PROB751. PROB754 et PROB855).

Si on considère l'ensemble des 64 problèmes, on constate, sur la base des résultats médians, que l'AG1 donne les mêmes performances que l'AG0 sur 54 d'entre eux, améliore les résultats médians de 6 problèmes (en gras dans la colonne AG1). Seulement 4 problèmes obtiennent de meilleurs résultats avec l'AG0 (en gras dans la colonne AG0). Les résultats de la colonne "Meilleur" et "Pire" sont également intéressants pour l'AG1. Dans la colonne "Meilleur", 6 problèmes ont été améliorés, 56 ont été atteints et 2 sont pires que l'AG0. Enfin, la colonne "Pire" présente, pour l'AG1, 9 problèmes améliorés, 51 atteints et 4 pires que l'AG0. L'algorithme AG1 est ainsi retenu dans la conception de l'approche de parallélisation décrite à la section suivante.

3.4 Parallélisation de l'algorithme génétique

L'un des objectifs spécifiques de ce mémoire est d'introduire le parallélisme dans les algorithmes génétiques. Nous utilisons un modèle en îlots avec une topologie en anneau unidirectionnel comme forme de parallélisation de l'algorithme génétique AG1. C'est un modèle qui a fait ses preuves dans la littérature et présente un moyen intéressant d'améliorer les performances. Plusieurs travaux tels ceux de Grosso [76], Pettey *et al.* [119], Tanese [148], Beldin [20] et Anderson *et al.* [14] ont adopté ce modèle.

Le fonctionnement de l'algorithme parallèle est le suivant : l'algorithme AG1 est

Prb	# comm.	B&B	*	AG0(1000) Meilleur	Médian	Pire	AG1 Meilleur	Médian	Pire
PROB401	15	90	*	4,4	**4,4**	21,1	4,4	11,1	37,8
PROB402	15	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB403	15	3418	*	0,0	2,1	3,7	0,0	1,9	3,4
PROB404	15	1067	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB405	15	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB406	15	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB407	15	1861	*	0,0	0,7	2,7	0,0	1,2	2,9
PROB408	15	5660	*	0,0	1,3	2,5	0,0	1,5	2,5
PROB501	25	264		2,3	4,2	8,3	3,0	4,9	12,1
PROB502	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB503	25	3511		-0,1	0,4	4,9	-0,1	0,6	2,2
PROB504	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB505	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB506	25	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB507	25	7225	*	1,2	3,0	5,3	0,5	2,8	5,2
PROB508	25	2067		-7,4	**-1,7**	5,2	0,0	5,3	11,7
PROB601	35	30		53,3	140,0	236,7	**43,3**	140,0	236,7
PROB602	35	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB603	35	17774		0,6	2,9	5,6	1,4	3,2	4,6
PROB604	35	19277		0,3	2,7	4,5	1,3	2,7	4,7
PROB605	35	291		5,5	25,8	45,4	6,9	23,7	**39,2**
PROB606	35	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB607	35	13274		1,1	2,5	3,7	-0,3	3,1	9,2
PROB608	35	6704		-28,0	-23,3	-14,4	-28,4	-23,0	-16,9
PROB701	45	116		59,5	**80,2**	136,2	**35,3**	85,3	**119,0**
PROB702	45	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB703	45	27097		1,0	3,2	4,6	1,4	2,8	4,8
PROB704	45	15941		-0,8	3,8	9,1	0,7	4,8	10,0
PROB705	45	234		48,3	**69,2**	91,5	45,3	77,8	106,0
PROB706	45	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB707	45	25070		-2,3	1,3	5,0	-1,2	0,5	3,1
PROB708	45	24123		-2,2	0,5	3,4	-3,4	0,2	5,5
PROB551	55	212		97,3	154,6	288,6	95,8	**128,8**	**162,3**
PROB552	55	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB553	55	40828		4,7	5,9	7,7	3,7	5,3	6,6
PROB554	55	15091		11,0	17,3	25,0	8,3	15,0	20,0
PROB555	55	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB556	55	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB557	55	36489		4,3	7,2	10,1	2,5	4,9	8,4
PROB558	55	20624		6,7	12,8	18,4	2,6	8,1	13,7
PROB651	65	295		96,6	136,3	178,2	**72,2**	**112,2**	**168,5**
PROB652	65	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB653	65	57779		4,8	6,8	8,9	5,3	6,3	10,0
PROB654	65	34468		8,9	14,6	18,8	7,6	11,3	15,7
PROB655	65	13		5300,0	7800,0	9500,0	**853,8**	**1107,7**	**1961,5**
PROB656	65	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB657	65	56246		5,7	7,5	10,6	4,0	5,4	6,8
PROB658	65	29308		9,1	15,1	21,6	**1,9**	**6,8**	**11,9**
PROB751	75	263		162,7	198,3	244,0	126,2	171,1	197,0
PROB752	75	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB753	75	78211		5,7	7,8	9,1	5,0	6,4	7,8
PROB754	75	35826		11,9	17,8	26,1	12,4	15,7	**19,8**
PROB755	75	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB756	75	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB757	75	61513		6,6	8,9	11,8	3,7	5,3	7,7
PROB758	75	40277		9,1	19,0	21,8	7,1	**11,4**	17,0
PROB851	85	453		126,9	165,6	194,3	123,4	164,2	191,2
PROB852	85	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB853	85	98540		5,0	6,8	8,2	4,9	7,0	9,0
PROB854	85	80693		7,7	9,8	12,4	7,4	9,3	12,1
PROB855	85	333		118,3	157,1	200,9	137,5	158,6	**191,3**
PROB856	85	0	*	0,0	0,0	0,0	0,0	0,0	0,0
PROB857	85	89654		4,2	6,1	9,3	4,5	6,4	8,9
PROB858	85	77919		5,4	8,9	11,5	5,1	8,2	11,1

* Solution Optimale

Tableau 3.2 – Résultats AG1

dupliqué sur un nombre de processeurs défini et chaque processeur dispose de sa population propre ou sous-population. Les sous-populations sont reliées entre elles en forme d'anneau. À un intervalle donné I, chaque processeur envoie une copie de ses R meilleurs individus à son voisin de droite, et va, par conséquent, recevoir R individus qu'il intègre dans sa sous-population en remplaçant ses pires individus. Lorsque le nombre total de générations est effectué, chaque sous-population est triée et les meilleurs individus issus de ces sous-populations sont eux-mêmes rassemblés et triés. Cette version parallèle est notée AGP0.

L'exécution de l'algorithme AGP0 s'est faite sur une machine parallèle possédant les caractéristiques suivantes : un noeud principal AMD Opteron(tm) Processeur 250 (64 bits) 2.4GHz avec 4Go de mémoire vive sur Fedora Core 3, deux noeuds quadruple processeurs Quad AMD Opteron(tm) Processeur 880 (64 bits) 2.4GHz avec 16Go de mémoire vive sur CentOS 4.3 et soixante six noeuds Dual AMD Opteron(tm) Processeur 250 (64 bits) 2.4GHz avec 4Go de mémoire vive. C'est le même environnement d'exécution utilisé pour l'AG0 et l'AG1 à l'exception du nombre de processeurs sélectionné via le mode de soumission. Cependant, pour garder les vitesses de communications homogènes, l'exécution s'est déroulée seulement sur les noeuds biprocesseurs en spécifiant l'utilisation d'un seul processeur par noeud.

L'implémentation de l'AGP0 s'est faite dans le langage C et à l'aide du standard MPI décrit à la Section 2.1.4.2. De plus, comme l'environnement matériel permet l'utilisation du réseau Infiniband [86], on a pu utiliser la dernière version mvapich 2 [114] qui est l'implémentation de MPI pour ce type de réseau. De même, la compilation s'est faite avec le compilateur mvapich2-0.9.8. Diverses fonctions MPI ont été utilisées pour gérer les communications entre les processeurs et sont citées durant les expérimentations.

L'exécution sur plusieurs processeurs amène la possibilité de parcourir l'espace de recherche de plusieurs manières différentes. Dans ce qui suit, deux manières vont être abordées : la première consiste à subdiviser la taille de la population sur le nombre de processeurs (stratégie POP), tandis que la deuxième réduit le nombre de générations (stratégie GEN). Les deux stratégies ont leurs avantages et leurs inconvénients comme on pourra le constater ultérieurement.

Dans les essais numériques réalisés, les problèmes dont les solutions sont déjà égales

à 0 par la version AG1 ne sont pas considérés puisque aucune amélioration ne peut y être apportée par la version parallèle. L'ensemble des problèmes tests est ainsi réduit à 43 problèmes. Le plan d'expérience fait varier le nombre de processeurs (1, 2 et 4) et ajuste, selon la stratégie, la taille de la sous-population sur chaque processeur ou le nombre de générations. La raison pour laquelle on ne dépasse pas 4 processeurs est la faible taille des problèmes. Le plan d'expérience comprend également la division des exécutions en deux grandes catégories : celles sans échanges d'information entre les processeurs et celles avec échanges. Pour les tests avec échanges, différentes fréquences (à toutes les 1, 5, 10, 20, 50,100 et 500 générations) et différents taux d'échanges (5%, 10% et 50% de la population échangée) sont évalués.

3.4.1 AG sans échange d'information

L'implémentation de l'AGP0 sans échange est un moyen de connaître le coût des communications. Cet ensemble de tests sert de plancher pour la qualité des solutions car on suppose, a priori, que les échanges sont bénéfiques pour la qualité. Il sert également de plafond pour les accélérations car, plus il y a d'échanges, plus il y a de ralentissements.

Pour la comparaison de la qualité des solutions, nous utilisons les mêmes critères que pour la comparaison de l'AG1 avec l'AG0, à savoir le résultat "Meilleur", "Médian" et "Pire". En premier lieu, on constate au Tableau 3.3 que les résultats de l'AGP0 utilisant 2 et 4 processeurs, selon la stratégie de subdivision de la population (POP) et de maintien du nombre de générations, ne permettent pas de conserver la qualité obtenue par l'AG1. En effet, un seul "Meilleur" et un seul "Pire" ont été améliorés (PROB655) alors qu'aucun résultat médian n'a été amélioré. Dans la majorité des cas, on observe une dégradation de la qualité des résultats par rapport à l'AG1. C'est la conséquence de la subdivision de la population et de l'isolement des sous-populations par l'absence d'échanges d'information.

En comparant les résultats de l'AGP0 exécuté sur 2 et 4 processeurs, on remarque que les résultats obtenus sont légèrement de moins bonne qualité sur 4 processeurs. Ainsi, on observe que le nombre de résultats médians atteints chute de 23 à 14 en considérant un écart maximal de 5% entre les résultats médians. Ce résultat peut être interprété de la manière

suivante : lorsqu'une population est isolée, elle a plus de chance de se diversifier si elle dispose d'un grand nombre d'individus. Cette diversité permet aux opérateurs de croisement d'être plus efficaces. Dans ce cas de figure, deux sous-populations de $N/2$ individus donnent de meilleurs résultats que quatre sous-populations de $N/4$ individus. C'est la raison pour laquelle la taille de la population est un paramètre capital pour la performance des algorithmes génétiques [81].

En deuxième lieu, le Tableau 3.4 présente les résultats de l'AGP0 sans échanges avec la stratégie de réduction du nombre de générations (GEN) et le maintien de la taille de la population pour chaque processeur. En comparant ces résultats avec l'AG1, on remarque que seulement deux "Meilleur" ont été améliorés (PROB605 et PROB655) et aucun résultat "Médian" ou "Pire" n'a été amélioré. Les résultats de l'AGP0 avec la stratégie de réduction du nombre de générations ne permettent pas de conserver la qualité de solution obtenue par l'AG1. Contrairement à la précédente stratégie, les sous-populations disposent de tous leurs individus mais l'algorithme s'exécute sur un plus petit nombre de générations. Comme les sous-populations sont isolées, l'AGP0 se comporte un algorithme séquentiel limité à 500 générations sur 2 processeurs et à 250 générations sur 4 processeurs. On a déjà constaté dans la Section 3.2 que l'AG0 performait mieux avec 1000 générations au lieu de 300 et que la convergence moyenne se situait autour de 850 générations. Les résultats de l'AGP0 confirment donc ce constat. Pour appuyer davantage cette idée, la comparaison des résultats sur 2 et 4 processeurs montre bien l'effet de la diminution de moitié du nombre de générations. En effet, aucun résultat médian sur 4 processeurs n'est meilleur que sur 2 processeurs. On observe également une chute dans le nombre de problèmes où le résultat médian de l'AG1 est atteint (26 avec 2 processeurs contre 17 avec 4 processeurs).

Le Tableau 3.5 résume le nombre de résultats médians de l'AG1 atteints par l'exécution de l'AGP0 sur 2 et 4 processeurs avec chacune des deux stratégies en considérant un écart maximal de 5%. On observe que les performances sur 2 processeurs sont meilleures que sur 4 processeurs et que la stratégie GEN s'avère un peu plus performante que POP (respectivement 26/43 et 23/43). Ce phénomène peut être expliqué par le fait qu'en absence de communications, l'algorithme est plus efficace lorsqu'il dispose d'une population complète même s'il a moins de générations pour tenter d'améliorer la solution. C'est ce qu'a notamment constaté Cantù-

Prb	# comm.	B&B *	AG1			Stratégie POP 2PROC			4PROC		
			Meilleur	Médian	Pire	Meilleur	Médian	Pire	Meilleur	Médian	Pire
PROB401	15	90 *	4,4	11,1	37,8	0,0	21,1	37,8	4,4	28,9	162,2
PROB403	15	3418 *	0,0	1,9	3,4	0,0	2,4	6,7	0,3	3,6	14,7
PROB404	15	1067 *	0,0	0,0	0,0	0,0	0,0	4,1	0,0	0,0	8,2
PROB407	15	1861 *	0,0	1,2	2,9	0,0	1,8	5,7	0,0	2,7	25,2
PROB408	15	5660 *	0,0	1,5	2,5	0,0	1,6	3,0	0,0	2,2	4,9
PROB501	25	264	3,0	4,9	12,1	1,9	6,1	31,1	3,4	10,6	129,5
PROB503	25	3511	-0,1	0,6	2,2	0,1	1,3	3,8	0,3	2,2	17,5
PROB507	25	7225 *	0,5	2,8	5,2	1,0	3,8	7,7	2,2	6,2	25,5
PROB508	25	2067	0,0	5,3	11,7	0,0	12,3	38,9	5,1	25,7	65,2
PROB601	35	30	43,3	140,0	236,7	93,3	193,3	346,7	120,0	266,7	716,7
PROB603	35	17774	1,4	3,2	4,6	2,8	4,5	7,7	3,0	6,1	11,8
PROB604	35	19277	1,3	2,7	4,7	2,2	4,1	7,0	3,4	6,3	12,0
PROB605	35	291	6,9	23,7	39,2	12,4	38,1	124,1	17,2	64,9	215,8
PROB607	35	13274	-0,3	3,1	9,2	1,6	4,3	10,0	3,8	8,4	18,4
PROB608	35	6704	-28,4	-23,0	-16,9	-27,6	-16,8	-4,3	-20,7	-10,2	6,4
PROB701	45	116	35,3	90,5	119,0	69,8	104,3	179,3	62,9	152,6	239,7
PROB703	45	27097	1,4	2,8	4,8	2,4	4,5	6,3	3,5	6,8	10,1
PROB704	45	15941	0,7	4,8	10,0	2,0	6,9	15,1	6,1	15,2	25,6
PROB705	45	234	45,3	77,8	106,0	49,6	89,7	227,8	66,7	103,0	215,0
PROB707	45	25070	-1,2	0,5	3,1	-0,9	3,4	6,7	1,4	6,2	10,4
PROB708	45	24123	-3,4	0,2	5,5	-0,7	3,4	9,0	3,7	8,9	15,5
PROB551	55	212	95,8	128,8	162,3	102,4	149,5	240,1	126,9	205,2	353,8
PROB553	55	40828	3,7	5,3	6,6	4,1	6,3	9,2	5,7	8,6	14,7
PROB554	55	15091	8,3	15,0	20,0	11,7	21,2	38,6	19,2	32,7	56,6
PROB557	55	36489	2,5	4,9	8,4	4,0	7,5	10,1	6,4	10,0	13,7
PROB558	55	20624	2,6	8,1	13,7	10,0	15,8	28,5	14,8	24,4	40,2
PROB651	65	295	72,2	112,2	168,5	90,8	128,8	171,9	103,1	161,7	228,8
PROB653	65	57779	5,3	6,3	10,0	5,7	8,3	10,2	6,8	9,9	13,2
PROB654	65	34468	7,6	11,3	15,7	11,9	18,0	24,1	15,4	25,2	32,5
PROB655	65	13	853,8	1107,7	1961,5	**838,5**	1369,2	**1892,3**	1115,4	1753,8	3192,3
PROB657	65	56246	4,0	5,4	6,8	5,5	7,9	10,2	7,0	10,1	13,8
PROB658	65	29308	1,9	6,8	11,9	7,2	12,9	20,3	11,6	25,1	35,3
PROB751	75	263	126,2	171,1	197,0	140,3	203,0	320,9	159,3	262,4	389,0
PROB753	75	78211	5,0	6,4	7,8	6,3	8,5	10,2	7,6	9,9	14,2
PROB754	75	35826	12,4	15,7	19,8	16,8	24,2	33,7	22,4	34,9	46,8
PROB757	75	61513	3,7	5,3	7,7	5,3	7,9	10,4	6,9	10,9	15,1
PROB758	75	40277	7,1	11,4	17,0	13,7	19,9	26,0	21,3	30,2	40,2
PROB851	85	453	123,4	164,2	191,2	150,3	177,3	233,3	148,8	237,1	341,9
PROB853	85	98540	4,9	7,0	9,0	6,4	7,8	10,4	7,9	10,7	13,7
PROB854	85	80693	7,4	9,3	12,1	8,9	13,4	16,2	12,0	17,9	23,2
PROB855	85	333	137,5	158,6	191,3	138,7	188,3	240,8	151,4	228,2	351,4
PROB857	85	89654	4,5	6,4	8,9	6,8	9,0	10,8	8,7	11,4	16,9
PROB858	85	77919	5,1	8,2	11,1	9,8	13,8	19,3	15,1	21,1	32,0

* Solution Optimale

Tableau 3.3 – Résultats AGP0 sans échanges avec stratégie POP

Prb	# comm.	B&B		AG1 Meilleur	AG1 Médian	AG1 Pire	Stratégie GEN 2PROC Meilleur	2PROC Médian	2PROC Pire	4PROC Meilleur	4PROC Médian	4PROC Pire
PROB401	15	90	*	4,4	11,1	37,8	4,4	16,7	40,0	4,4	24,4	50,0
PROB403	15	3418	*	0,0	1,9	3,4	0,0	2,3	4,6	0,0	3,2	7,2
PROB404	15	1067	*	0,0	0,0	0,0	0,0	0,0	6,7	0,0	0,0	9,2
PROB407	15	1861	*	0,0	1,2	2,9	0,0	1,8	3,0	0,0	2,4	9,0
PROB408	15	5660	*	0,0	1,5	2,5	0,0	1,7	3,1	0,0	2,3	4,2
PROB501	25	264		3,0	4,9	12,1	1,5	4,9	12,9	2,7	8,3	15,2
PROB503	25	3511		-0,1	0,6	2,2	0,0	1,2	3,8	0,4	1,9	5,0
PROB507	25	7225	*	0,5	2,8	5,2	1,5	4,4	7,7	1,9	5,3	11,1
PROB508	25	2067		0,0	5,3	11,7	0,0	6,7	24,4	4,2	20,2	49,6
PROB601	35	30		43,3	140,0	236,7	66,7	180,0	346,7	96,7	243,3	376,7
PROB603	35	17774		1,4	3,2	4,6	1,9	4,0	6,5	2,8	5,6	8,9
PROB604	35	19277		1,3	2,7	4,7	1,5	3,7	6,3	3,1	5,6	9,8
PROB605	35	291		6,9	23,7	39,2	**-0,7**	38,1	114,1	16,5	59,8	113,1
PROB607	35	13274		-0,3	3,1	9,2	2,5	4,6	9,1	3,2	8,0	17,6
PROB608	35	6704		-28,4	-23,0	-16,9	-25,6	-18,2	-5,3	-23,6	-11,0	6,2
PROB701	45	116		35,3	90,5	119,0	69,8	112,9	186,2	88,8	144,0	245,7
PROB703	45	27097		1,4	2,8	4,8	2,3	4,5	7,9	3,3	6,5	9,8
PROB704	45	15941		0,7	4,8	10,0	1,7	8,5	13,4	4,7	12,8	29,2
PROB705	45	234		45,3	77,8	106,0	60,7	94,0	120,1	70,9	106,0	161,5
PROB707	45	25070		-1,2	0,5	3,1	-0,5	3,5	8,6	1,7	5,7	10,3
PROB708	45	24123		-3,4	0,2	5,5	-0,2	3,2	9,2	2,2	6,6	11,8
PROB551	55	212		95,8	128,8	162,3	119,3	163,2	255,7	153,3	223,1	357,5
PROB553	55	40828		3,7	5,3	6,6	3,9	7,2	8,7	5,8	8,6	11,2
PROB554	55	15091		8,3	15,0	20,0	10,3	20,9	28,6	15,8	28,9	41,8
PROB557	55	36489		2,5	4,9	8,4	5,0	7,5	10,3	5,8	9,0	14,2
PROB558	55	20624		2,6	8,1	13,7	6,2	14,4	24,8	10,4	22,4	35,6
PROB651	65	295		72,2	112,2	168,5	96,3	132,2	172,5	116,9	173,6	272,5
PROB653	65	57779		5,3	6,3	10,0	6,1	8,4	10,0	6,5	10,7	13,2
PROB654	65	34468		7,6	11,3	15,7	12,2	16,1	22,2	15,6	22,8	30,8
PROB655	65	13		853,8	1107,7	1961,5	692,3	1469,2	2169,2	1169,2	1846,2	3507,7
PROB657	65	56246		4,0	5,4	6,8	4,9	7,2	10,5	6,3	10,1	14,4
PROB658	65	29308		1,9	6,8	11,9	6,7	12,3	23,8	9,3	20,9	34,6
PROB751	75	263		126,2	171,1	197,0	160,5	211,4	319,4	185,2	304,2	433,5
PROB753	75	78211		5,0	6,4	7,8	6,1	8,0	11,6	7,6	10,5	14,4
PROB754	75	35826		12,4	15,7	19,8	16,7	23,9	31,8	22,1	31,6	43,5
PROB757	75	61513		3,7	5,3	7,7	5,7	8,1	11,3	7,4	11,1	16,0
PROB758	75	40277		7,1	11,4	17,0	12,3	19,1	34,0	18,5	26,3	40,3
PROB851	85	453		123,4	164,2	191,2	139,7	192,9	287,2	193,8	271,3	429,8
PROB853	85	98540		4,9	7,0	9,0	6,0	8,7	11,0	8,4	11,2	14,6
PROB854	85	80693		7,4	9,3	12,1	9,5	12,7	16,6	11,2	17,5	24,1
PROB855	85	333		137,5	158,6	191,3	144,1	194,9	252,0	196,1	265,2	340,8
PROB857	85	89654		4,5	6,4	8,9	6,1	9,1	12,6	8,5	12,2	15,8
PROB858	85	77919		5,1	8,2	11,1	8,7	14,0	21,0	14,4	19,7	26,4

* Solution Optimale

Tableau 3.4 – Résultats AGP0 sans échanges avec stratégie GEN

Paz [27] [28] en analysant le comportement des algorithmes génétiques parallèles possédant des sous-populations isolées.

Stratégie	2 Processeurs	4 Processeurs
POP	23	14
GEN	26	17

Tableau 3.5 – Nombre de résultats médians de l'AG1 atteints sans communication par l'AGP0

Pour les deux stratégies, les faibles performances de l'AGP0 sans communication s'expliquent par la diminution de l'espace de recherche et par l'absence de mécanismes de conservation de la diversité. Cette performance est prévisible puisque, privé de communication, l'AGP0 se comporte comme un algorithme séquentiel mais disposant d'un espace de recherche plus restreint que AG1.

Le Tableau 3.6 présente les temps d'exécutions *(T)* et les accélérations *(A)* relatifs aux essais numériques précédents. Les temps d'exécution représentent les moyennes des 20 exécutions. On constate que l'AGP0 sans communication produit de très bons temps d'exécution et des accélérations qui, pour la plupart des problèmes, atteignent le maximum théorique. C'est ce qu'on a appelé une accélération linéaire à la Section 2.1.5.1.

Deux problèmes tests de tailles respectives de 250 et 500 commandes ont été ajoutés afin d'observer le comportement de l'AGP0 lorsque la taille du problème augmente. De ce fait, le comportement de l'algorithme pourra également être analysé sur la base de l'utilisation d'un plus grand nombre de processeurs.

Les Tableaux 3.7 et 3.8 présentent les résultats pour ces deux problèmes en utilisant 2, 4, 8, 16 et 32 processeurs avec les deux stratégies POP et GEN sans échanges d'information. Comme ces problèmes ne sont pas des problèmes de la littérature et qu'on ne dispose pas de l'optimum trouvé par une méthode exacte, on suppose que l'optimum est le meilleur résultat trouvé par l'AG1 pour fins de comparaison. En premier lieu, le Tableau 3.7 présente des résultats similaires observés pour les problèmes de petite taille, à savoir que la qualité de solution se dégrade considérablement avec l'augmentation du nombre de processeurs avec la stratégie de subdivision de la population (POP).

Au Tableau 3.8, avec la stratégie de réduction du nombre de générations et de maintien

		AG1	AGP0 POP				AGP0 GEN			
			2PROC		4PROC		2PROC		4PROC	
Prb	#comm.	T	T	A	T	A	T	A	T	A
PROB401	15	0,318	0,172	1,8	0,106	3,0	0,158	2,0	0,080	4,0
PROB403	15	0,305	0,149	2,1	0,099	3,1	0,150	2,0	0,076	4,0
PROB404	15	0,347	0,190	1,8	0,125	2,8	0,173	2,0	0,087	4,0
PROB407	15	0,327	0,179	1,8	0,114	2,9	0,173	1,9	0,087	3,7
PROB408	15	0,350	0,189	1,9	0,124	2,8	0,175	2,0	0,087	4,0
PROB501	25	0,834	0,422	2,0	0,217	3,8	0,445	1,9	0,242	3,5
PROB503	25	0,889	0,471	1,9	0,220	4,0	0,461	1,9	0,237	3,8
PROB507	25	1,011	0,531	1,9	0,273	3,7	0,510	2,0	0,254	4,0
PROB508	25	1,074	0,554	1,9	0,292	3,7	0,541	2,0	0,270	4,0
PROB601	35	1,999	0,997	2,0	0,487	4,1	0,984	2,0	0,499	4,0
PROB603	35	1,930	0,981	2,0	0,483	4,0	0,966	2,0	0,482	4,0
PROB604	35	2,015	1,052	1,9	0,527	3,8	1,009	2,0	0,507	4,0
PROB605	35	1,964	0,998	2,0	0,485	4,0	1,005	2,0	0,501	3,9
PROB607	35	1,974	1,028	1,9	0,513	3,9	1,019	1,9	0,508	3,9
PROB608	35	2,214	1,153	1,9	0,571	3,9	1,108	2,0	0,557	4,0
PROB701	45	3,389	1,711	2,0	0,900	3,8	1,664	2,0	0,851	4,0
PROB703	45	3,294	1,665	2,0	0,872	3,8	1,660	2,0	0,824	4,0
PROB704	45	3,516	1,781	2,0	0,888	4,0	1,764	2,0	0,897	3,9
PROB705	45	3,435	1,712	2,0	0,916	3,7	1,720	2,0	0,861	4,0
PROB707	45	3,470	1,749	2,0	0,908	3,8	1,718	2,0	0,870	4,0
PROB708	45	3,674	1,843	2,0	0,921	4,0	1,845	2,0	0,922	4,0
PROB551	55	5,129	2,587	2,0	1,335	3,8	2,615	2,0	1,313	3,9
PROB553	55	5,135	2,625	2,0	1,333	3,9	2,620	2,0	1,304	3,9
PROB554	55	5,482	2,807	2,0	1,477	3,7	2,758	2,0	1,391	3,9
PROB557	55	5,441	2,786	2,0	1,470	3,7	2,743	2,0	1,379	3,9
PROB558	55	5,745	2,941	2,0	1,554	3,7	2,882	2,0	1,460	3,9
PROB651	65	6,950	3,796	1,8	2,189	3,2	3,769	1,8	1,892	3,7
PROB653	65	7,426	3,841	1,9	2,162	3,4	3,783	2,0	1,893	3,9
PROB654	65	7,974	3,983	2,0	2,222	3,6	4,023	2,0	2,032	3,9
PROB655	65	7,528	3,849	2,0	2,252	3,3	3,880	1,9	1,953	3,9
PROB657	65	7,895	4,009	2,0	2,246	3,5	3,946	2,0	1,979	4,0
PROB658	65	8,383	4,166	2,0	2,319	3,6	4,218	2,0	2,127	3,9
PROB751	75	10,321	5,206	2,0	2,811	3,7	5,203	2,0	2,631	3,9
PROB753	75	10,564	5,315	2,0	2,803	3,8	5,277	2,0	2,646	4,0
PROB754	75	11,090	5,664	2,0	2,994	3,7	5,601	2,0	2,844	3,9
PROB757	75	10,989	5,602	2,0	2,961	3,7	5,511	2,0	2,764	4,0
PROB758	75	11,438	5,817	2,0	3,083	3,7	5,758	2,0	2,908	3,9
PROB851	85	13,401	6,868	2,0	3,487	3,8	6,837	2,0	3,477	3,9
PROB853	85	13,932	6,951	2,0	3,483	4,0	6,976	2,0	3,483	4,0
PROB854	85	14,689	7,382	2,0	3,711	4,0	7,409	2,0	3,757	3,9
PROB855	85	13,853	7,091	2,0	3,583	3,9	7,056	2,0	3,591	3,9
PROB857	85	14,371	7,202	2,0	3,616	4,0	7,204	2,0	3,620	4,0
PROB858	85	15,241	7,632	2,0	3,834	4,0	7,899	1,9	3,905	3,9

Tableau 3.6 – Acccélerations AGP0 sans échanges

Prb	# comm.	Meilleur AG0	AG1			AGP0 POP					
						2 PROC			4 PROC		
			Meilleur	Médian	Pire	Meilleur	Médian	Pire	Meilleur	Médian	Pire
PREX250	250	504875	0,0	4,8	8,2	10,7	15,4	21,9	22,6	29,6	41,4
PREX500	500	38537	0,0	24,9	53,6	26,5	51,0	90,7	34,7	79,4	137,6
Prb	# comm.	Meilleur AG0	AGP0 POP								
			8 PROC			16 PROC			32 PROC		
			Meilleur	Médian	Pire	Meilleur	Médian	Pire	Meilleur	Médian	Pire
PREX250	250	504875	35,6	29,6	41,4	50,5	67,9	85,3	70,8	86,5	102,9
PREX500	500	38537	91,4	79,4	137,6	164,2	233,7	332,4	252,8	349,9	526,7

Tableau 3.7 – Résultats AGP0 sans échanges avec stratégie POP sur les grands problèmes

de la taille de la population pour chaque processeur (GEN), on observe, encore une fois, que la qualité de la solution se dégrade considérablement avec l'augmentation du nombre de processeurs. Cette dégradation de la qualité est toutefois plus rapide qu'avec la stratégie POP, ce qui est l'inverse de ce qui avait été observé pour les plus petits problèmes.

Prb	# comm.	Meilleur AG0	AG1			AGP0 GEN					
						2 PROC			4 PROC		
			Meilleur	Médian	Pire	Meilleur	Médian	Pire	Meilleur	Médian	Pire
PREX250	250	504875	0,0	4,8	8,2	12,5	18,0	26,2	31,1	39,3	50,2
PREX500	500	38537	0,0	24,9	53,6	97,9	135,0	160,4	197,3	260,1	306,0
Prb	# comm.	Meilleur AG0	AGP0 GEN								
			8 PROC			16 PROC			32 PROC		
			Meilleur	Médian	Pire	Meilleur	Médian	Pire	Meilleur	Médian	Pire
PREX250	250	504875	55,8	66,5	80,4	77,7	92,5	104,0	100,4	115,8	126,9
PREX500	500	38537	361,2	425,8	485,9	476,0	575,5	642,4	593,3	688,8	745,0

Tableau 3.8 – Résultats AGP0 sans échanges avec stratégie GEN sur les grands problèmes

Le Tableau 3.9 présente les accélérations obtenues sur ces deux problèmes. Comme c'était le cas précédemment, les accélérations sont très bonnes et l'augmentation du nombre de processeurs permet de mieux distinguer la performance des deux stratégies. En effet, les accélérations obtenues avec la stratégie GEN sont, dans tous les cas, inférieures à la stratégie POP. La non-linéarité des accélérations de la stratégie GEN s'explique par la conservation du nombre d'individus sur chaque processeur. En effet, il s'agit de la série de tests la plus exigeante en terme de calculs avec des sous-populations complètes (250 et 500 individus respectivement) à gérer. Cet écart n'était pas observable dans les problèmes de petite taille.

Cette première série de tests sur l'algorithme AGP0 sans échanges d'information permet de conclure que cette forme de parallélisation produit des résultats peu performants

		AG1	AGP0 POP									
			2PROC		4PROC		8PROC		16PROC		32PROC	
Prb	#comm.	T	T	A	T	A	T	A	T	A	T	A
PREX250	250	188,862	94,028	2,0	47,803	4,0	23,744	8,0	11,992	15,7	6,096	31,0
PREX500	500	1091,823	544,057	2,0	273,157	4,0	137,717	7,9	69,527	15,7	35,176	31,0
		AG1	AGP0 GEN									
			2PROC		4PROC		8PROC		16PROC		32PROC	
Prb	# comm.	T	T	A	T	A	T	A	T	A	T	A
PREX250	250	188,862	96,934	1,9	51,106	3,7	26,647	7,1	13,830	13,7	7,139	26,5
PREX500	500	1091,823	577,199	1,9	295,745	3,7	151,083	7,2	78,651	13,9	40,855	26,7

Tableau 3.9 – Accélérations sans échanges sur les grands problèmes

en ce qui concerne la qualité des solutions pour les deux stratégies testées en comparaison avec l'AG1. Cependant, pour les problèmes de 15 à 85 commandes, la stratégie GEN a donné de meilleurs résultats que la stratégie POP et cela s'est inversé dans le cas des problèmes de taille 250 et 500 commandes. Du point de vue des accélérations, on a globalement constaté des accélérations linéaires sauf dans le cas des gros problèmes avec la stratégie de réduction des générations. Ainsi, en poussant les tests sur les gros problèmes avec un plus grand nombre de processeurs, on a mesuré l'impact des deux stratégies sur les accélérations. D'après la littérature, les communications représentent le principal frein aux accélérations et c'est cet impact que l'on va étudié dans la prochaine section.

3.4.2 AG avec échange d'information

Le plan d'expérience présenté précédemment fait intervenir deux nouveaux paramètres dans le cas où il existe des échanges d'information entre les processeurs. Le premier paramètre concerne l'intervalle I, c'est-à-dire le nombre de générations entre deux échanges d'information entre deux processeurs. Le deuxième paramètre précise le taux de migration R, c'est-à-dire le pourcentage de la population qui est transmise à l'autre processeur. Les essais numériques ont été effectués en faisant varier ces deux paramètres selon la stratégie choisie (GEN ou POP). Les intervalles de migration testés sont à toutes les 1, 5, 10, 20, 50, 100 et 500 générations et les taux de migration sont de 10%, 20% et 50% de la population du processeur. Ces 21 scénarios sont évalués sur l'ensemble des problèmes.

Pour l'implémentation des échanges, les principales fonctions MPI utilisées sont :

MPI_Wtime pour le chronométrage des différentes itérations, MPI_Gather pour le rassemblement des résultats obtenus par chaque processeur, MPI_Send et MPI_Recv pour représenter les communications en anneau entre les processeurs avec le même principe décrit dans la Figure 2.9.

Le Tableau 3.10 résume la performance des différents scénarios selon la stratégie de subdivision de la population (POP). Pour chaque scénario, on a établi le nombre de problèmes dont les résultats médians sont supérieurs (>), équivalent (=) ou pires (<) que les résultats obtenus avec l'AG1. Une tolérance de 5% est encore utilisée pour cette comparaison de performance. D'après les résultats de ce tableau, on peut établir que les scénarios permettant d'améliorer ou de conserver le plus grand nombre de problèmes (résultats présentés en caractères gras) au niveau de la qualité des résultats se ressemblent selon le nombre de processeurs. En effet, les scénarios extrêmes quant à la fréquence d'échanges semblent moins performants et on note également que la fréquence d'échanges doit diminuer avec l'augmentation du nombre de processeurs. À ce qui a trait au taux de migration, il semble préférable de l'augmenter avec l'augmentation du nombre de processeurs. On remarque également, pour les scénarios performants, que le passage de 2 à 4 processeurs fait diminuer le nombre de problèmes améliorés ou atteints. C'est le même constat qui avait été observé à la Section 3.4.1 pour l'AGP0 sans communication et dont les performances sont résumées dans la dernière ligne du Tableau 3.10.

En réalisant une analyse plus détaillée de l'ensemble des résultats, on remarque également que, pour les fréquences 10 et 20, l'amélioration des problèmes concerne surtout les problèmes de taille supérieure à 25. Les améliorations ont été rapportées principalement sur les problèmes ayant les caractéristiques PTV=Low, TF=Low et DDR=Narrow (correspond aux problèmes dont le numéro se termine par 1) et également sur les problèmes ayant les caractéristiques PTV=High, TF=Low et DDR=Narrow (correspond aux problèmes dont le numéro se termine par un 5). En effet, pour chaque groupe de problèmes de taille supérieure à 25 commandes, tous les problèmes ayant ces caractéristiques ont été améliorés.

Sur la base des résultats du Tableau 3.10, on peut donc établir que les échanges entre les processeurs sont généralement bénéfiques pour l'amélioration de la qualité des résultats. Un des facteurs majeurs affectant la qualité de solution dans un algorithme génétique est la

diversité de la population. Le maintien de cette diversité, malgré la diminution de la taille des sous-populations, tout au long de l'exécution de la version parallèle est assuré par les échanges d'individus [70]. Les algorithmes coopèrent et se diversifient les uns les autres pour faire émerger de meilleures solutions.

Scénario	Stratégie POP					
	2 PROC			2 PROC		
	>	=	<	>	=	<
I1R10	1	34	8	0	22	21
I1R20	4	30	9	1	29	13
I1R50	0	20	23	0	19	24
I5R10	**7**	**35**	1	6	30	7
I5R20	**7**	**34**	2	**5**	**35**	3
I5R50	4	**38**	1	**5**	**36**	2
I10R10	**8**	**33**	2	8	28	7
I10R20	**8**	**35**	0	**7**	**33**	3
I10R50	**8**	**34**	1	**7**	**33**	3
I20R10	**9**	**33**	1	6	31	6
I20R20	**6**	**37**	0	**6**	**34**	3
I20R50	**9**	**34**	0	**7**	**35**	1
I50R10	**7**	**35**	1	4	30	9
I50R20	**7**	**35**	1	**4**	**36**	3
I50R50	**7**	**35**	1	**3**	**39**	1
I100R10	**5**	**35**	1	0	39	4
I100R20	**5**	**38**	0	1	36	6
I100R50	**7**	**36**	0	1	35	7
I500R10	1	36	6	0	18	25
I500R20	1	34	8	0	19	24
I500R50	1	36	6	0	21	22
NoCom	0	23	20	0	14	29

Tableau 3.10 – Résultats AGP0 avec échanges (POP)

Le Tableau 3.11 présente la performance des différents scénarios selon la stratégie de réduction du nombre de générations (GEN). Les scénarios permettant d'améliorer ou de conserver le plus grand nombre de résultats médians de l'AG1 sont présentés en caractères gras. On peut noter que l'échange à toutes les générations ne semble pas très performant et qu'il ne faut pas, non plus, trop espacer l'échange compte tenu de la réduction du nombre de générations. L'augmentation du nombre de processeurs entraîne également l'utilisation d'un échange plus fréquent pour obtenir de bons résultats. Pour les scénarios avec les fréquences d'échanges produisant les meilleurs résultats, on note que le taux de migration n'a pas beaucoup d'influence. Toutefois, le passage de 2 à 4 processeurs fait chuter dramatiquement le nombre de

problèmes dont les résultats médians de l'AG1 sont améliorés. En effet, seul le scénario I5R50 avec l'utilisation de 4 processeurs permet d'améliorer un problème. Les différents scénarios permettent, dans l'ensemble, d'obtenir des résultats de meilleure qualité que la version sans communication.

Globalement, les essais numériques réalisés selon les deux stratégies permettent ainsi d'établir que la stratégie POP s'avère supérieure à la stratégie GEN pour l'amélioration des résultats médians pour l'ensemble des problèmes. Compte tenu de la taille des problèmes, il semble également préférable de limiter le nombre de processeurs à 2.

Scénario	Stratégie GEN					
	2 PROC			2 PROC		
	>	=	<	>	=	<
I1R10	1	35	7	0	32	11
I1R20	1	31	11	0	31	12
I1R50	0	21	22	0	21	22
I5R10	**4**	**37**	2	**0**	**36**	7
I5R20	**4**	**40**	0	**0**	**37**	6
I5R50	**5**	**36**	2	**1**	**38**	4
I10R10	**5**	**36**	2	**0**	**36**	7
I10R20	**6**	**36**	1	**0**	**37**	6
I10R50	**3**	**37**	3	**0**	**37**	6
I20R10	**5**	**36**	2	0	32	11
I20R20	**4**	**37**	2	0	35	8
I20R50	**4**	**38**	1	0	34	9
I50R10	**1**	**39**	3	0	28	15
I50R20	2	37	4	0	28	15
I50R50	2	36	5	0	28	15
I100R10	3	35	5	0	26	17
I100R20	1	38	4	0	25	18
I100R50	2	35	6	0	24	19
I500R10	0	35	8	0	17	26
I500R20	0	33	10	0	16	27
I500R50	0	33	10	0	16	24
NoCom	0	26	17	0	17	26

Tableau 3.11 – Résultats AGP0 avec échanges (GEN)

Du point de vue de l'accélération, le graphique 3.2 illustre l'impact des communications sur le temps d'exécution de l'algorithme. On constate que, pour 2 et 4 processeurs, la stratégie de réduction des générations (GEN) s'avère plus rapide que celle par subdivision de la population (POP), quelque soit le scénario. Les fréquences d'échanges qui, dans les tableaux précédents, ont donné les meilleurs résultats possèdent des accélérations intéressantes. L'aspect croissant

des courbes de gauche à droite est la conséquence directe de la fréquence des échanges. Dans la partie gauche du graphe, les échanges s'effectuent à toutes les générations et sont de moins en moins fréquents lorsqu'on progresse sur l'axe des abscisses. Cependant, l'augmentation progressive de l'accélération est aussi synonyme de diminution de la qualité comme on l'a montré précédemment. Paradoxalement, lorsque il y trop d'échanges, l'accélération et la qualité sont toutes les deux affectées puisque les algorithmes sur chaque processeur n'ont pas le temps de travailler dans leur espace de recherche respectif et toutes les sous-populations vont finir par se ressembler [28].

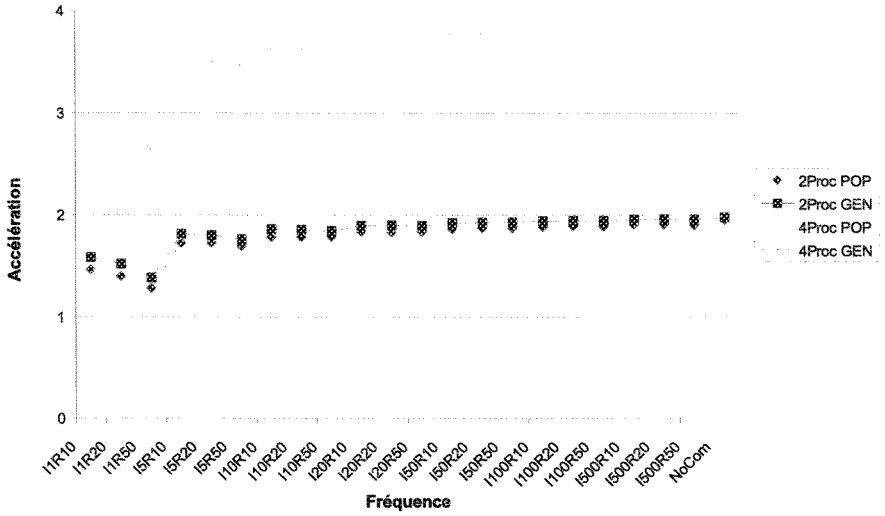

Figure 3.2 – Accélérations des petits problèmes sur 2 et 4 processeurs

Examinons maintenant la performance de l'AGP0 avec les 21 scénarios pour les problèmes de 250 et 500 commandes avec un plus grand nombre de processeurs. Les résultats présentés aux Tableaux 3.12 et 3.13 expriment l'écart de la solution médiane trouvée par rapport au résultat médian l'AG1, ceci avec la stratégie POP.

Au Tableau 3.12, on constate, tout d'abord, qu'aucun scénario ne permet d'améliorer la qualité de solution de l'AG1 et que peu d'entre eux réussissent à la conserver. En utilisant une marge de 5%, seuls les résultats indiqués en caractères gras au Tableau 3.12 expriment une qualité de solution équivalente à l'AG1. Les meilleurs résultats obtenus avec chaque groupe de processeurs sont soulignés. On note ainsi que les meilleurs résultats tendent à apparaître pour des fréquences plus petites avec l'augmentation du nombre de processeurs. Globalement,

la meilleure performance est obtenue en utilisant seulement 2 processeurs avec des fréquences d'échanges entre 5 et 20 et sans vraiment d'impact sur le choix du taux de migration.

Avec le problème de 500 commandes dont les résultats sont présentés au Tableau 3.13, il est possible d'améliorer la solution trouvée par l'AG1 avec certains scénarios jusqu'à 16 processeurs. Les meilleurs résultats obtenus avec chaque groupe de processeurs sont soulignés. Plusieurs scénarios permettent également de conserver un niveau de qualité équivalent à l'AG1. Ces résultats sont indiqués en caractères gras au Tableau 3.13. Tout comme le problème de 250 commandes, on note ainsi que les meilleurs résultats tendent à apparaître pour des fréquences plus petites avec l'augmentation du nombre de processeurs. L'analyse des problèmes de 250 et 500 commandes nous amène à un constat différent par rapport aux petits problèmes, à savoir que l'augmentation du nombre de processeurs n'est pas forcement néfaste à la qualité, et selon la fréquence, on a pu atteindre de très bons résultats. On ne peut cependant tirer de conclusion quant à la fréquence idéale à utiliser pour ce genre de problèmes. L'apport des communications semble améliorer la qualité des résultats mais sans tendance particulière.

	Stratégie POP				
	PREX 250				
Scénario	2 PROC	4 PROC	8 PROC	16 PROC	32 PROC
I1R10	10,4	10,7	12,5	21,4	86,3
I1R20	12,5	10,5	10,2	11,6	28,6
I1R50	19,5	16,1	15,8	14,1	17,7
I5R10	5,7	6,7	7,1	13,4	86,3
I5R20	5,3	6,0	**4,6**	11,4	21,9
I5R50	**4,9**	**4,0**	6,1	10,3	22,6
I10R10	**5,0**	7,0	8,3	18,0	86,7
I10R20	**4,4**	5,3	8,1	15,3	32,2
I10R50	**4,3**	5,7	7,9	17,6	31,9
I20R10	6,8	7,1	13,3	26,9	86,4
I20R20	**4,5**	6,0	12,3	24,5	43,3
I20R50	**4,3**	7,4	12,9	24,2	44,4
I50R10	5,6	11,6	21,8	38,1	86,4
I50R20	6,3	10,7	21,0	38,4	58,5
I50R50	5,4	11,2	23,2	39,6	60,3
I100R10	8,1	15,8	29,1	48,2	86,4
I100R20	7,5	16,2	29,6	47,5	68,2
I100R50	7,9	16,6	29,1	49,5	69,6
I500R10	12,2	26,0	41,6	62,0	86,7
I500R20	13,7	25,0	41,8	62,6	81,1
I500R50	11,5	24,7	43,1	62,6	81,0
NoCom	15,4	29,6	47,6	67,9	86,5

Tableau 3.12 – Résultats AGP0 sur le problème 250 avec échanges (POP)

Il est également intéressant de constater la différence entre les deux problèmes 250

et 500 dans les fréquences qui donnent les meilleurs résultats. Un anneau constitué de 4 processeurs a produit les meilleurs résultats médians de la série de tests du problème 250, alors qu'il a fallu un anneau de 16 processeurs pour le problème 500.

	Stratégie POP				
	PREX 500				
Scénario	2 PROC	4 PROC	8 PROC	16 PROC	32 PROC
I1R10	13,7	9,6	**3,0**	5,5	61,8
I1R20	13,7	**-1,0**	**-4,3**	**-5,9**	**12,7**
I1R50	39,6	13,5	**2,2**	**4,3**	19,8
I5R10	5,5	4,1	9,1	24,0	63,9
I5R20	**1,5**	**-2,0**	**3,5**	13,8	46,7
I5R50	**2,4**	**-5,5**	**-5,1**	9,3	49,0
I10R10	**4,6**	10,0	23,1	49,8	100,6
I10R20	**4,2**	6,2	16,6	39,9	88,7
I10R50	8,1	**1,9**	10,5	31,9	85,3
I20R10	13,5	19,6	43,0	79,9	151,7
I20R20	9,0	17,8	31,3	74,8	135,5
I20R50	10,8	12,5	32,1	66,3	132,2
I50R10	20,3	36,6	68,0	124,6	215,1
I50R20	22,5	32,6	64,4	116,0	205,0
I50R50	11,4	28,1	65,4	117,5	207,1
I100R10	22,5	48,5	87,6	156,6	256,3
I100R20	27,7	45,9	85,9	155,5	250,1
I100R50	22,3	50,4	87,8	156,3	253,4
I500R10	35,7	68,4	123,2	203,2	315,5
I500R20	42,3	70,9	124,6	205,6	315,0
I500R50	39,0	69,0	124,1	209,3	316,3
NoCom	51,0	79,4	140,4	233,7	349,9

Tableau 3.13 – Résultats AGP0 sur le problème 500 avec échanges (POP)

Du point de vue de l'accélération, on remarque d'après les graphiques 3.3 et 3.4 que les fréquences qui ont donné les meilleurs résultats au niveau de la qualité sont celles avec les accélérations les moins bonnes. Globalement, on constate un coût de communication important lorsque les échanges sont importants et les accélérations linéaires ne sont pas aussi fréquentes que pour les petits problèmes. L'aspect croissant des courbes est maintenu avec un net décalage vers le bas lorsqu'on compare les accélérations de 250 et 500. Le fait que la taille du problème est multipliée par 2 fait augmenter la taille des messages échangés entre les processeurs et, par conséquent, fait augmenter les coûts de communications qui freinent l'accélération.

Pour terminer le plan d'expérience, examinons les résultats des grands problèmes avec la stratégie GEN. Les Tableaux 3.14 et 3.15 présentent les écarts des résultats médians de AGP0 par rapport à l'optimum trouvé par AG1. On constate d'après ces tableaux que les résultats obtenus ne sont pas de la même qualité que la précédente stratégie. En considérant

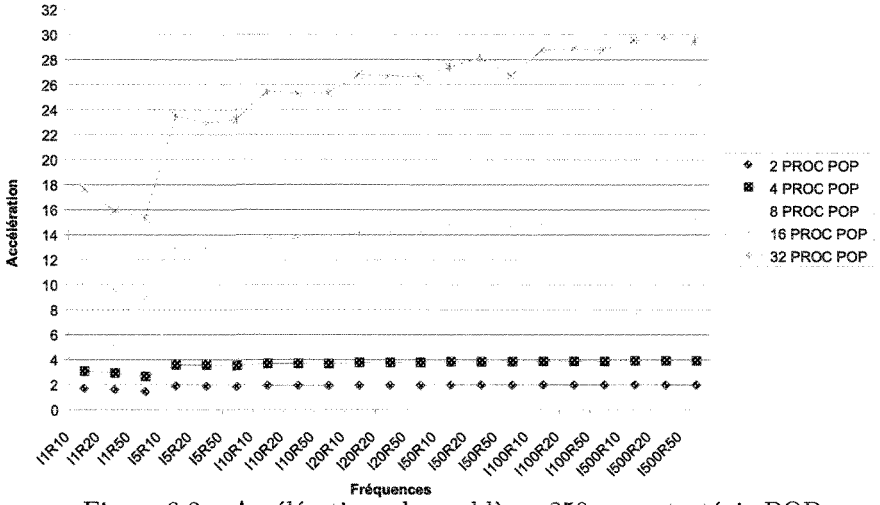

Figure 3.3 – Accélérations du problème 250 avec stratégie POP

la même marge de 5%, aucun scénario n'obtient un résultat équivalent à AG1. Les meilleurs résultats par nombre de processeur sont soulignés et cet écart grandit avec la taille de l'anneau.

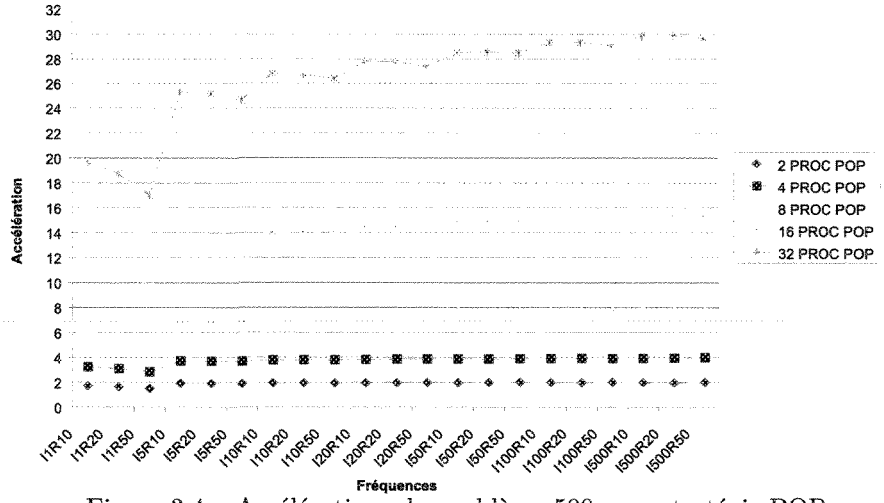

Figure 3.4 – Accélérations du problème 500 avec stratégie POP

En comparant les accélérations obtenues avec cette stratégie, on remarque le maintien de la tendance croissante en diminuant la fréquence des échanges. De plus, les accélérations illustrées dans les Figures 3.5 et 3.6 sont plus regroupées vers une moyenne et pour un nombre de processeurs supérieur à 4 sont toutes sous-linéaires, autant pour le problème 250 que 500.

	Stratégie GEN				
	PREX 250				
Scénario	2 PROC	4 PROC	8 PROC	16 PROC	32 PROC
I1R10	15,0	23,7	40,5	66,7	96,6
I1R20	17,5	21,6	37,7	63,9	94,8
I1R50	25,6	28,7	42,3	65,9	96,0
I5R10	9,9	22,9	48,5	81,0	108,6
I5R20	10,2	22,3	46,3	78,6	107,0
I5R50	9,3	20,1	46,2	78,3	106,8
I10R10	10,4	25,1	54,8	85,3	111,1
I10R20	8,2	24,6	52,9	83,8	110,1
I10R50	8,7	23,7	52,0	83,4	110,2
I20R10	9,2	28,8	57,8	88,0	114,0
I20R20	8,0	28,7	57,9	87,8	114,0
I20R50	8,4	27,4	56,5	86,5	113,2
I50R10	11,7	32,0	62,4	90,8	115,3
I50R20	11,4	30,9	61,7	91,0	115,4
I50R50	10,7	31,4	61,9	90,3	115,6
I100R10	11,5	35,6	62,8	93,0	115,5
I100R20	12,1	35,1	63,2	92,5	115,3
I100R50	14,0	35,1	63,5	92,0	115,4
I500R10	16,6	38,9	65,8	92,9	115,6
I500R20	15,9	38,8	66,3	92,9	115,8
I500R50	16,2	39,7	65,9	93,3	115,6
NoCom	18,0	39,3	66,5	92,5	115,8

Tableau 3.14 – Résultats AGP0 sur le problème 250 avec échanges (GEN)

	Stratégie GEN				
	PREX 500				
Scénario	2 PROC	4 PROC	8 PROC	16 PROC	32 PROC
I1R10	83,4	151,9	256,0	408,5	575,9
I1R20	78,8	156,8	247,7	402,2	573,6
I1R50	123,6	197,9	276,7	431,2	582,3
I5R10	78,0	168,3	316,1	493,2	640,5
I5R20	71,6	158,5	300,1	492,6	639,1
I5R50	73,1	154,6	304,6	485,5	633,6
I10R10	80,2	184,7	346,3	526,5	660,7
I10R20	75,6	178,1	342,2	518,3	658,1
I10R50	77,4	173,8	343,9	518,1	656,4
I20R10	100,8	206,5	377,7	547,5	676,3
I20R20	96,1	202,5	372,6	544,8	679,3
I20R50	87,2	198,6	372,2	542,6	676,1
I50R10	112,7	227,9	403,2	559,1	688,7
I50R20	104,5	223,3	399,7	562,6	688,9
I50R50	107,6	222,9	399,5	557,6	689,8
I100R10	112,0	243,2	411,4	573,9	687,1
I100R20	116,4	234,1	416,2	575,4	687,1
I100R50	111,5	241,5	408,0	574,7	686,4
I500R10	126,5	256,1	423,8	573,8	687,7
I500R20	120,1	260,7	422,6	571,3	688,7
I500R50	123,2	257,5	428,7	576,1	687,6
NoCom	135,0	260,1	425,8	575,5	688,8

Tableau 3.15 – Résultats AGP0 sur le problème 500 avec échanges (GEN)

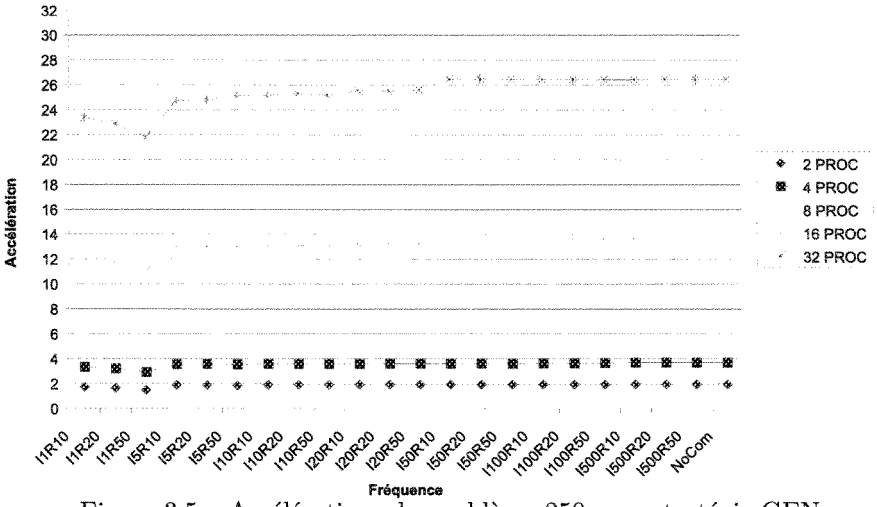

Figure 3.5 – Accélérations du problème 250 avec stratégie GEN

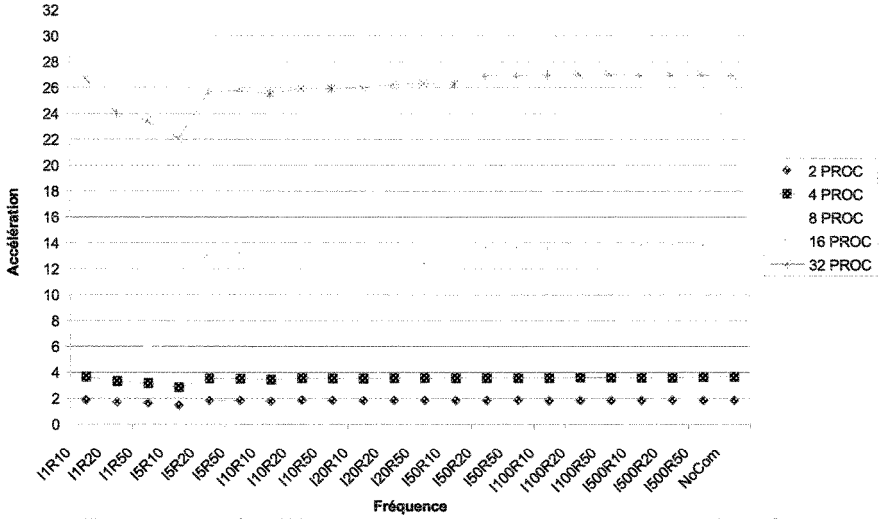

Figure 3.6 – Accélérations du problème 500 avec stratégie GEN

3.5 Conclusion

La parallélisation sans échanges d'informations a apporté des résultats peu convaincants mais a permis de poser les premières hypothèses quant à la comparaison de la performance de la stratégie de subdivision de la population par rapport à la stratégie de réduction du nombre de générations. En effet, la stratégie de réduction du nombre de générations a permis d'obtenir de meilleurs résultats et des accélérations linéaires dans la plupart des cas.

Lors de l'ajout des communications entre les processeurs sous forme d'anneau unidirectionnel, la stratégie de subdivision de population s'est avérée la plus efficace et a permis de confirmer les conclusions obtenues par d'autres auteurs [28]. D'un autre coté, les deux stratégies présentent des accélérations acceptables et aucun ralentissement par rapport à la version séquentielle n'est observé.

Pour pousser encore plus les limites de l'algorithme développé, des problèmes de grande taille ont été générés et testés. Il a été constaté que le comportement de l'algorithme pour ces problèmes n'est pas le même et que la qualité de solution et les bonnes accélérations étaient plus difficiles à obtenir. Il est également intéressant de constater que les meilleures solutions ont été générées pour des fréquences plus élevées que celles des petits problèmes. On peut ainsi déduire une relation directe entre la taille de la population et la fréquence d'échange.

La parallélisation des algorithmes génétiques est une voie intéressante pour obtenir des résultats de qualité dans des temps raisonnables. Cependant, il faut savoir définir les paramètres appropriés pour en tirer le meilleur profit. De plus, il a été démontré que l'augmentation du nombre de processeurs est bénéfique jusqu'à un certain point au-delà duquel aucune amélioration de la qualité n'est possible.