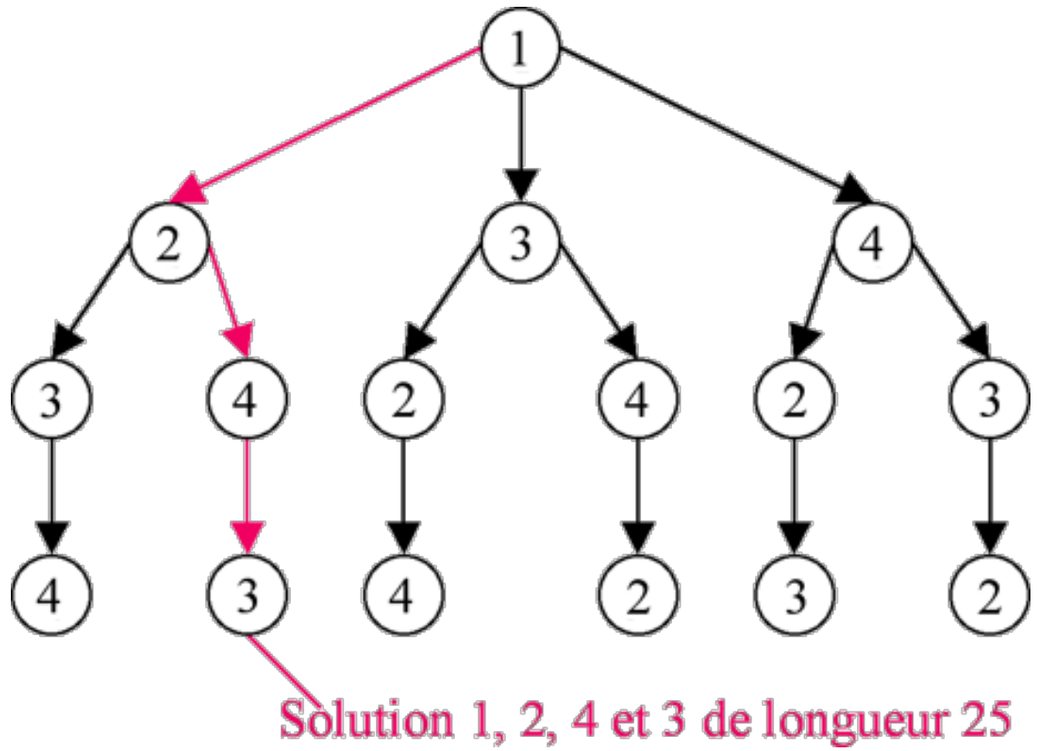
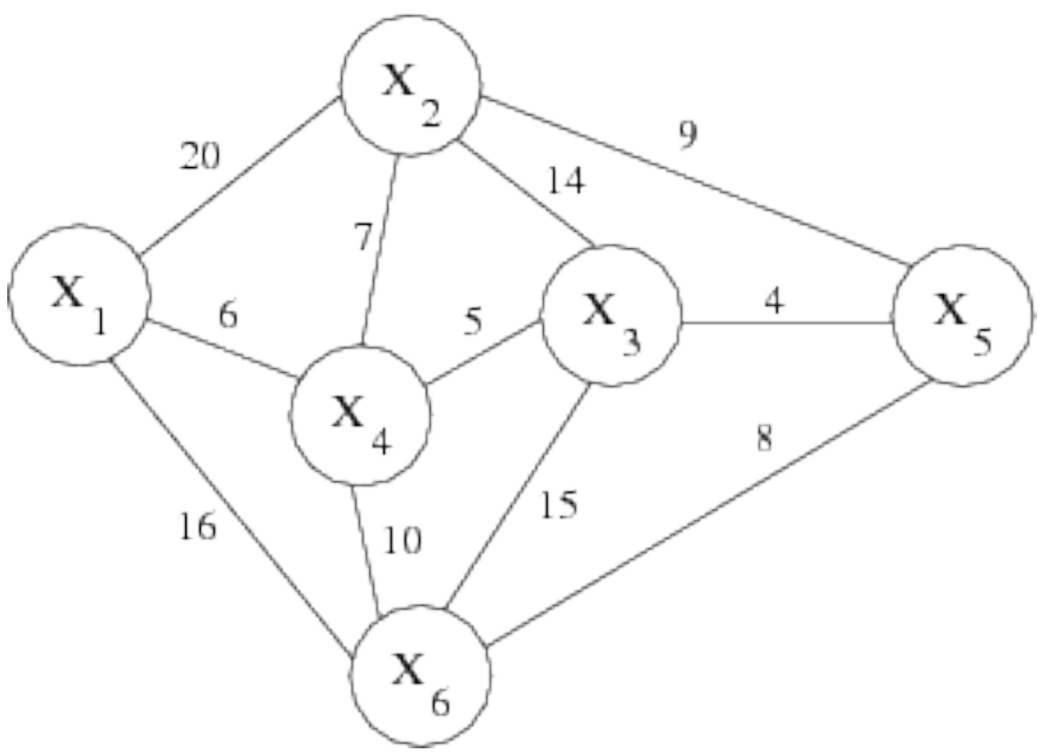
OUTILS DE SYNCHRONISATION

Compte rendu de TP

COLEAU Victor & COUCHOUD Thomas

PRÉSENTATION DU PROBLÈME



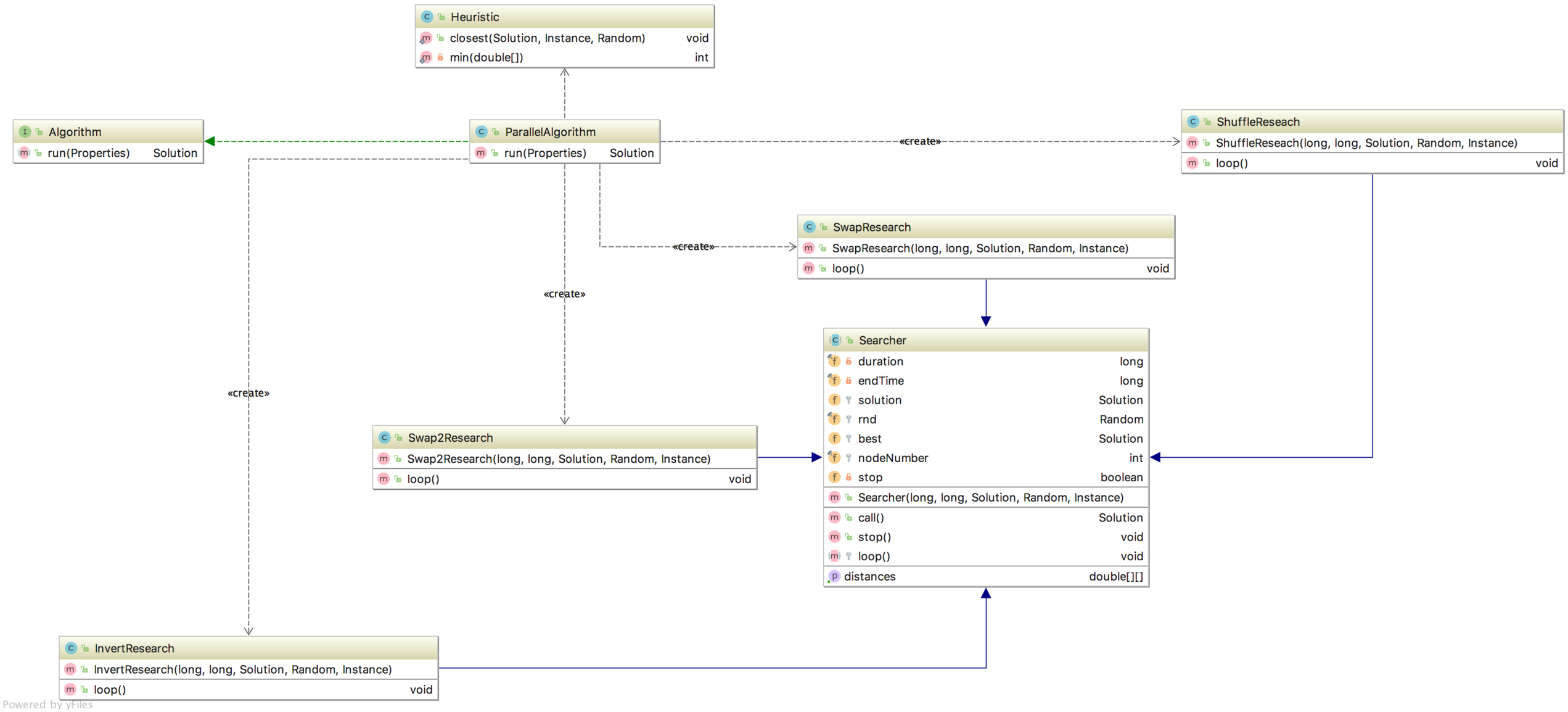


- Trouver le chemin le plus cours, en passant par toutes les villes et en revenant au départ.
- Représentation sous forme de graph, chaque arc représente une liaison entre ville possible avec sa distance.
- Impossible de tester toutes les solutions, avec 100 villes déjà 4950 issues sont possibles.

CHOIX DE PARALLÉLISATION

- Parallélisation de la recherche d'un chemin optimal à partir de différents chemins originels traités en parallèle.
- Exploration des différents chemins en fonction des solutions de départs de manière équilibrée.
- Les tâches ne sont pas crées pour des actions très courtes, cela permet de rentabiliser le temps de création de celle-ci.
- Permet une exploration rapide sur de nombreux départs et éventuellement plus longue sur un faible nombre de départs.
- Chaque tâche calcule une solution optimale de manière locale puis le meilleur résultat de chaque tâche est retenu.

DIAGRAMME DE CLASSES



ORGANISATION DES CLASSES

- ParallelAlgorithm est notre point d'entrée, implémente Algorithm. Celui-ci lance les différentes tâches de recherche et traite leur résultat.
- Searcher représente une tâche de recherche. Celle-ci cherche durant un temps donné ou jusqu'à ce qu'un temps limite soit atteint. La partie de recherche est effectué dans le méthode loop représentant une itération de recherche.
- Différentes classes étendes Searcher et implémente différents procédés de générer un voisinage.

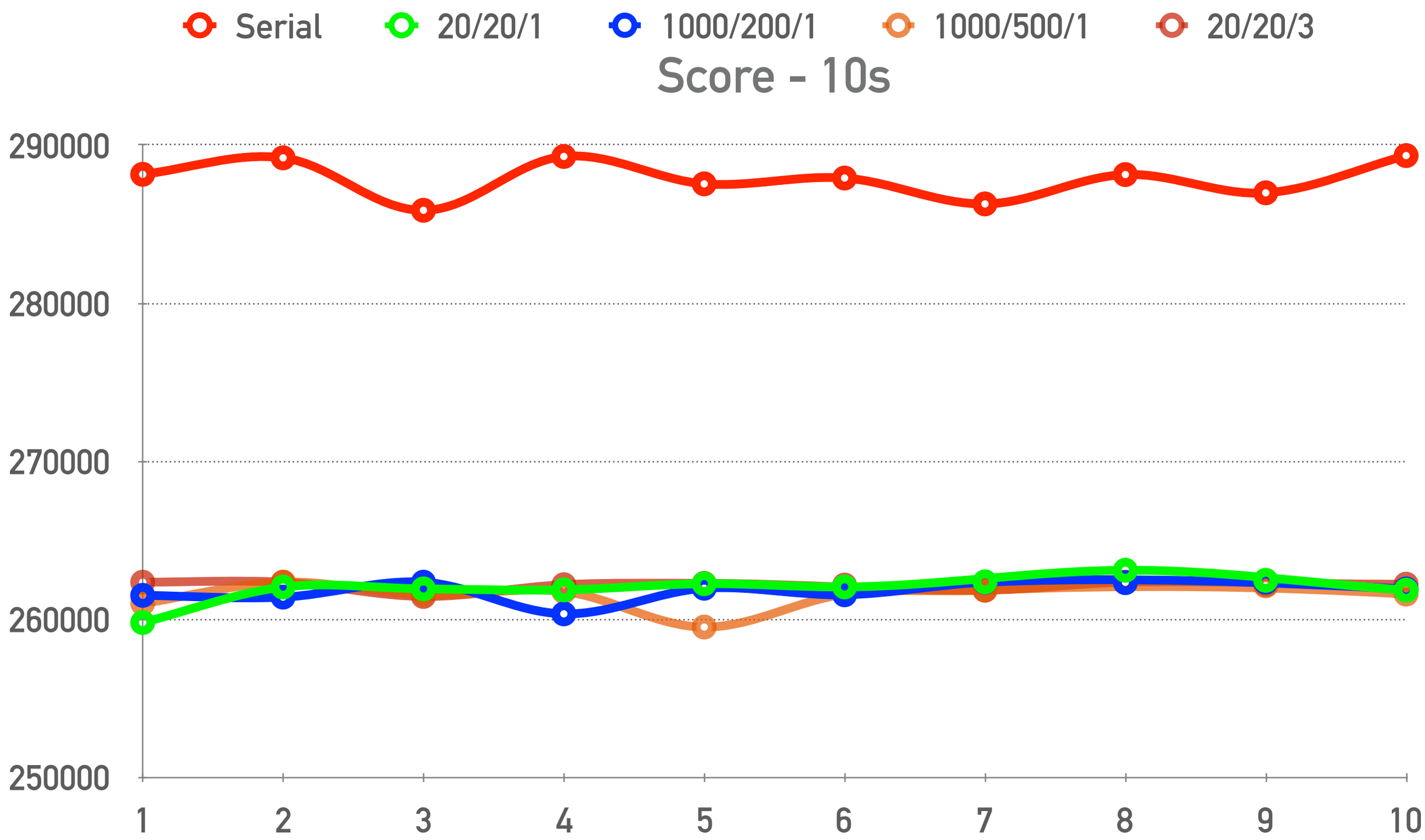
SEARCHERS

- ShuffleResearch mélange la liste de manière aléatoire à chaque itération.
- SwapResearch inverse deux villes consécutives de manière aléatoire. Par exemple la suite 1 - 2 - 3 deviendra 1 - 3 - 2 si 2 est tiré aléatoirement.
- Swap2Research inverse deux villes de manière aléatoire.
- InvertResearch déplace une ville aléatoire à une position aléatoire.

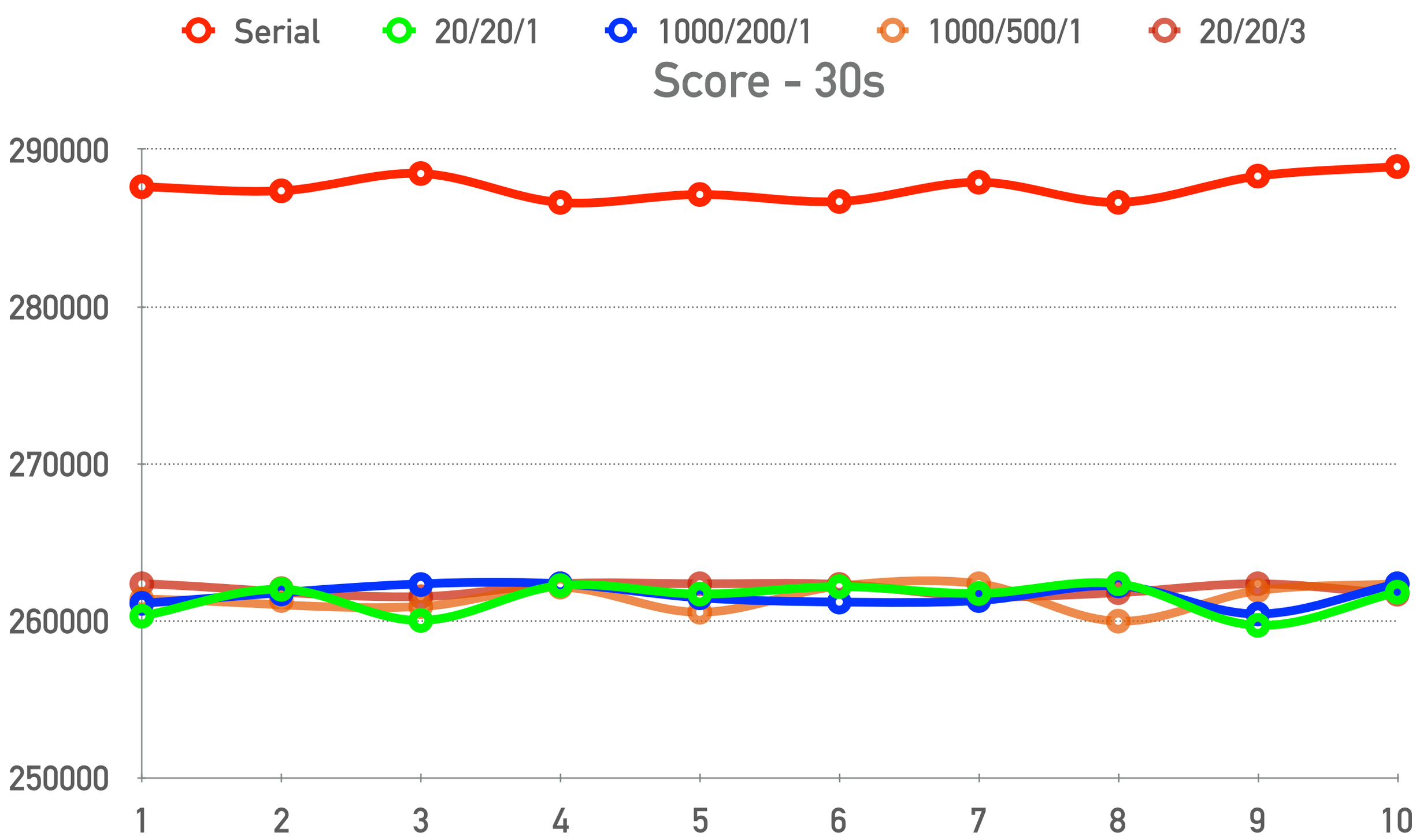
CONFIGURATION

- startingPoints représente le nombre de solutions (nombre de tâches) de départ à explorer.
- nbThreads représente le nombre de threads qui seront créés.
- searchID représente le Searcher à utiliser (0: ShuffleResearch, 1: SwapResearch, 2: InvertResearch, 3: Swap2Research)

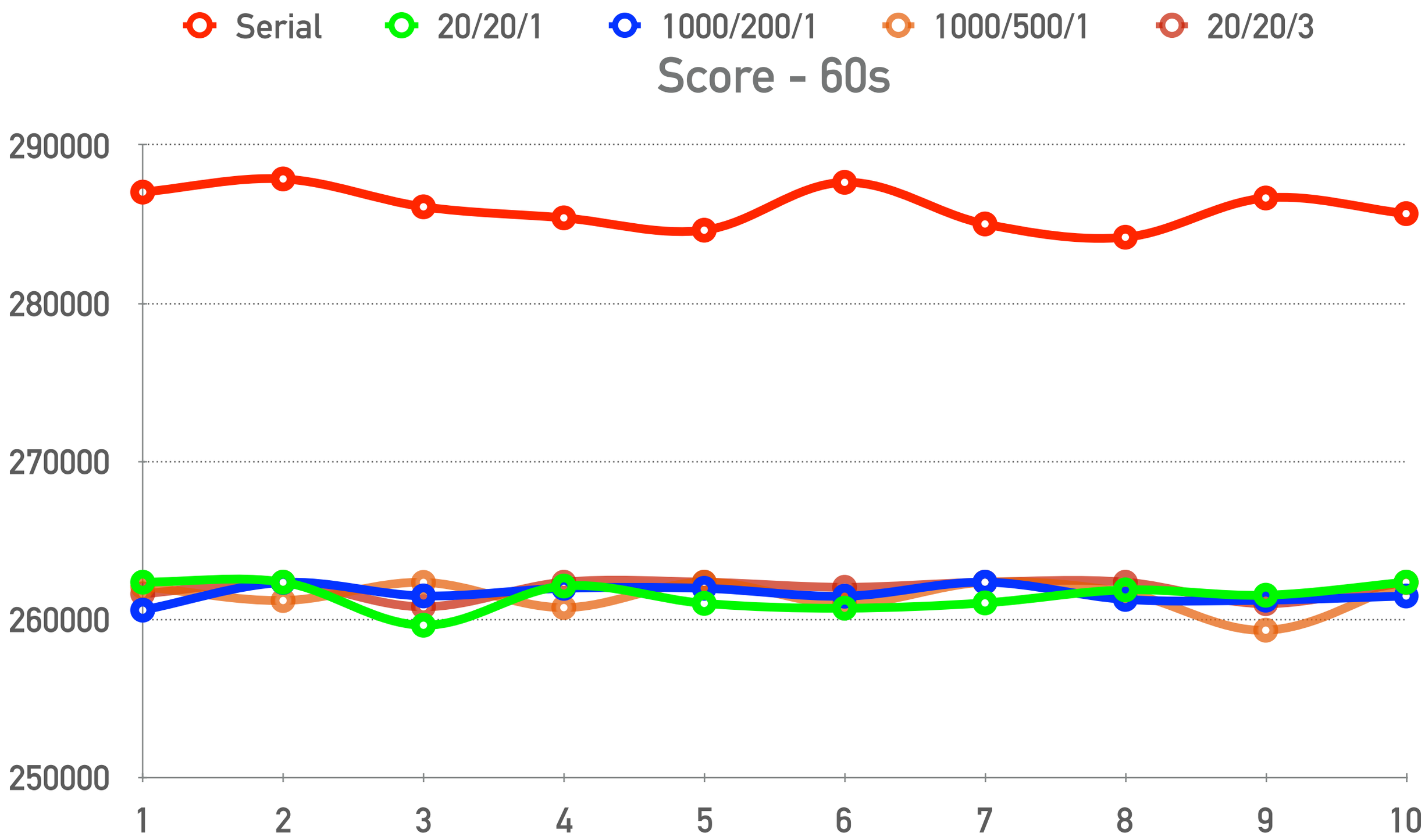
RÉSULTATS – LU980



RÉSULTATS – LU980



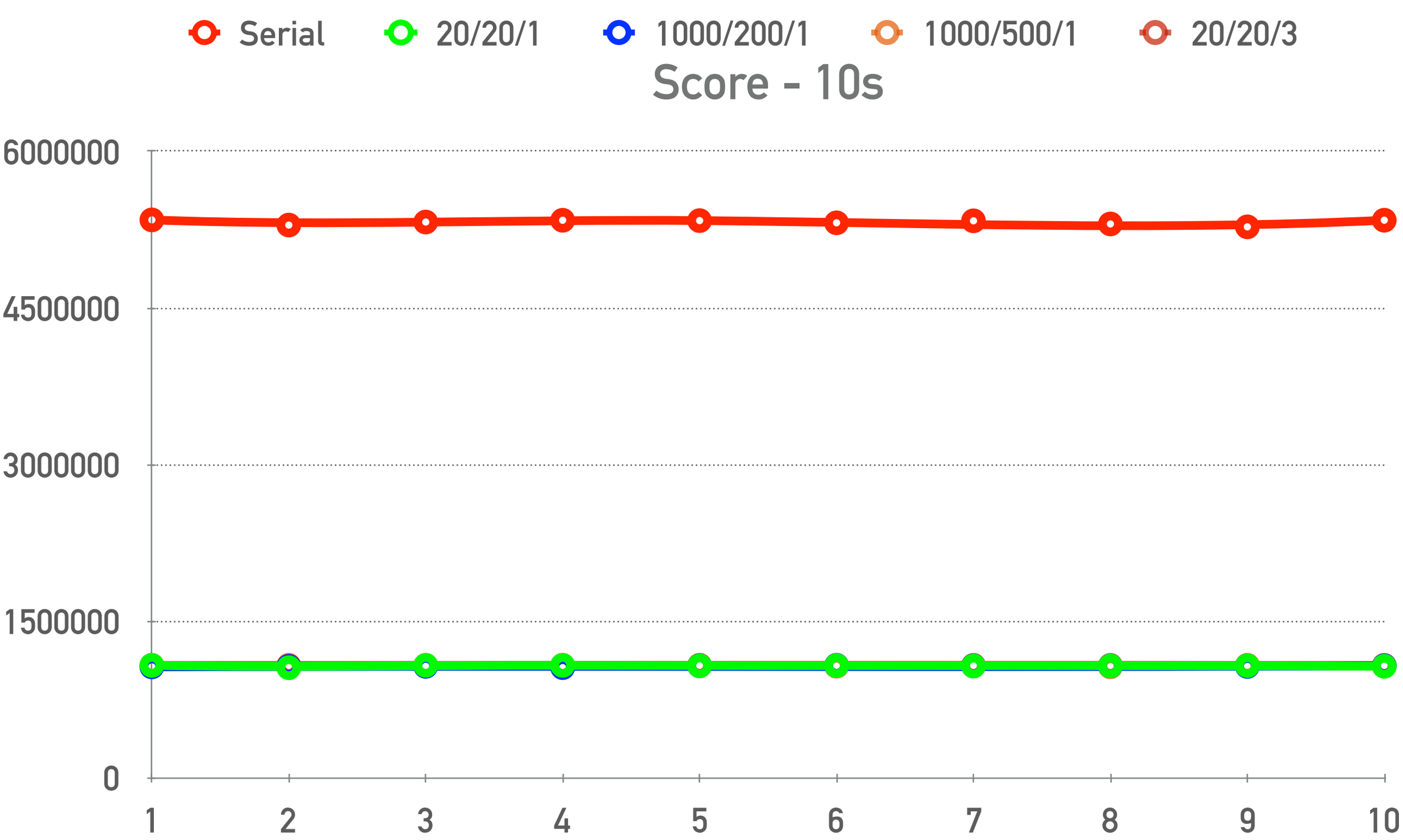
RÉSULTATS – LU980



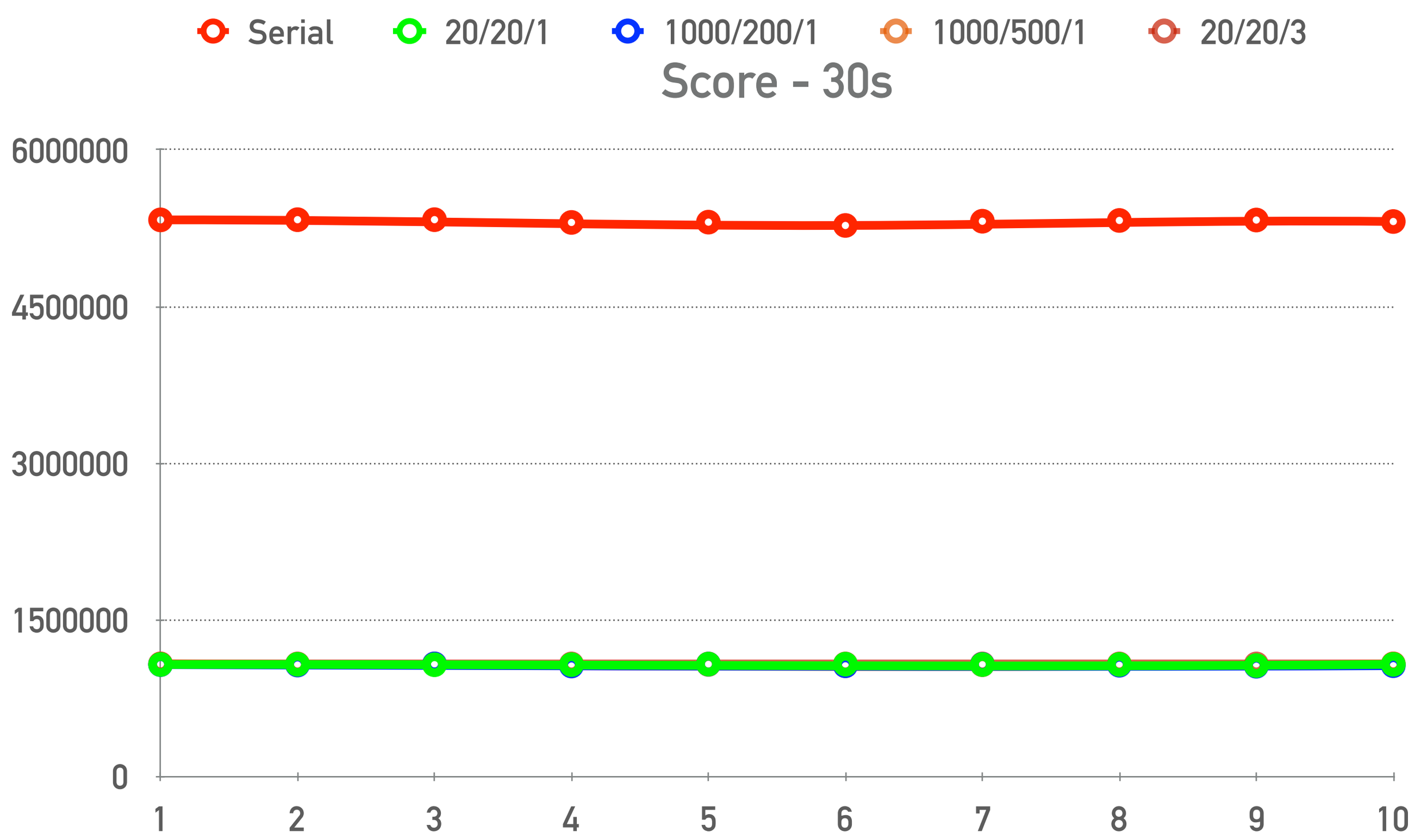
ANALYSE OBSERVATIONS

- Le temps t'execution semble affecter l'algorithme serial, mais très peu celui en parallèle.
- Le rapport nombre de threads / nombre de tâches semble peu affecter les résultats des différentes executions en parallèle.
- Les creux des courbes des exécutions parallèles semble être dû à la solution de départ.
- Le temps d'exécution des algorithme suit relativement bien le temps demandé. Le delta moyen est aux alentours de 0s tandis que l'algorithme serial est souvent à +1s de décalage. Le maximum observé est de +7s sur un test avec 500 threads.

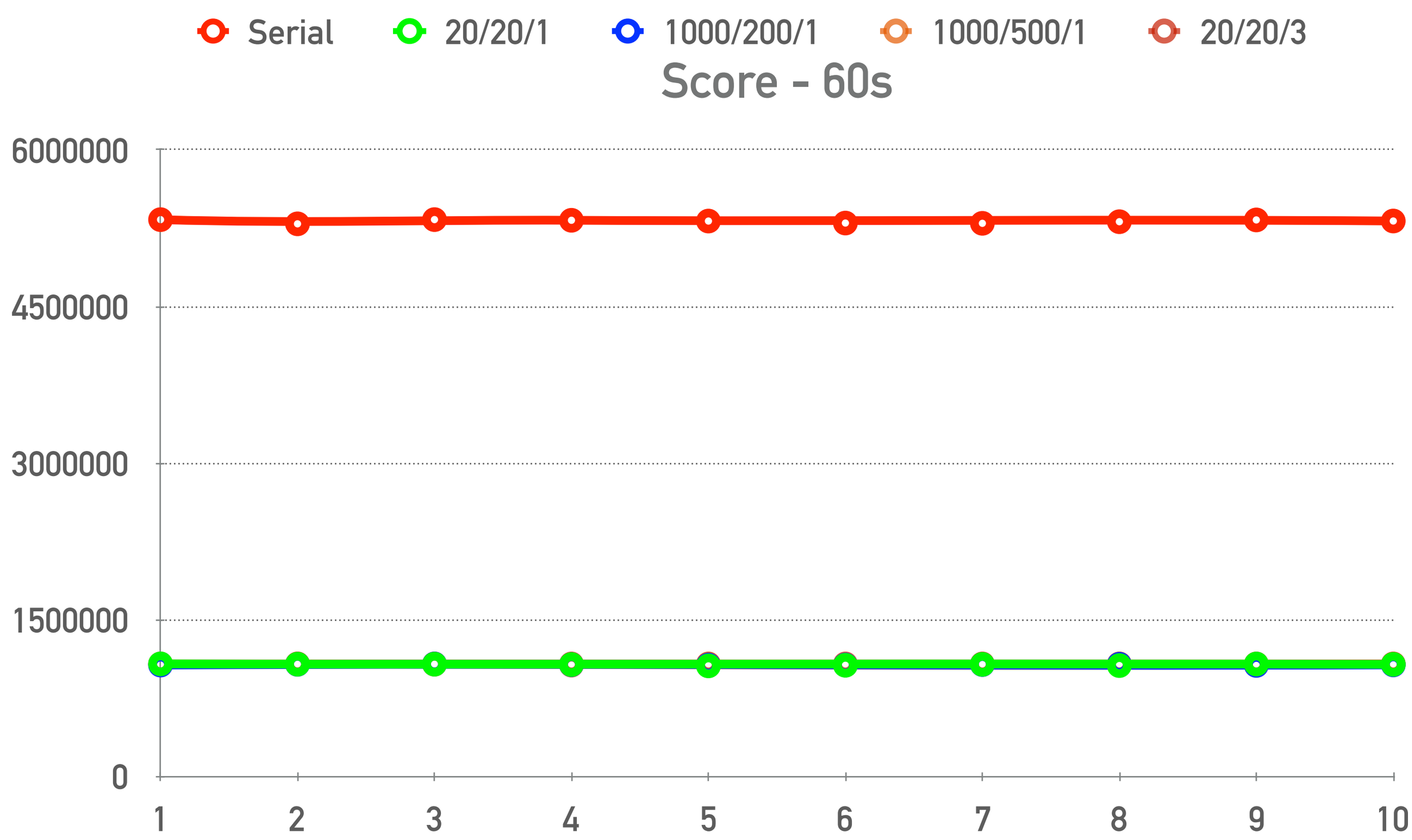
RÉSULTATS – MU1979



RÉSULTATS – MU1979



RÉSULTATS – MU1979



ANALYSE OBSERVATIONS

- Résultats parallèles toujours meilleurs.
- Stabilisation des différentes configuration du parallèle.
- Des deltas sur les temps plus important. Moyenne d'environ +2s (toujours +1s sur le serial), avec un maximum à +10s.