Relatório 2ª entrega de CPD – Grupo 11

66936 – Ana Silva
69656 – Carlos Bartolomeu
69879 – André Rodrigues

À semelhança da primeira entrega , também o desenvolvimento da versão mpi do projecto se revelou bastante desafiante. Mais uma vez nos deparámos com variados problemas nomeadamente :

1. Conseguir sincronização entre os vários processos e coerência entre as sub-gerações e gerações no seu todo;
2. Conseguir Load balancing .

Soluções adoptadas para os diferentes problemas :

1. Como já havíamos optado na primeira entrega , nesta mantivemos o uso de duas matrizes (mas agora por cada processo) para gerir as sub-geraçoes e para que a cada movimento de um animal o cálculo da posição destino não fosse atrapalhado pelo movimento de outros animais . No entanto no desenvolvimento desta entrega deparámo-nos com um problema, como cada processo tem o seu "sub-mundo" , no processamento de uma célula que se encontrasse nas suas fronteiras não haveria hipótese de testar as células destino fora das fronteiras , pois apenas o processo a correr o "sub-mundo" adjacente as teria. Então a solução que arranjámos foi cada processo ter duas linhas correspondentes às suas fronteiras contendo apenas o tipo da célula (gelo, árvore, etc) , afim de as enviar aos processos a correr os "sub-mundos " adjacentes para que estes possam fazer os testes necessários apontados anteriormente. E ainda duas linhas correspondentes às linhas adjacentes às fronteiras, contendo não só o tipo mas também os valores respectivos a essa célula (breeding level, etc) afim de se aquando do término do processamento do seu "sub-mundo" houver animais que se tenham movido para fora das fronteiras, os processos responsáveis por correr os "sub-mundos" adjacentes possam fazer merge dessas linhas com as respectivas linhas do seu sub-mundo" (tratando de possíveis conflitos).

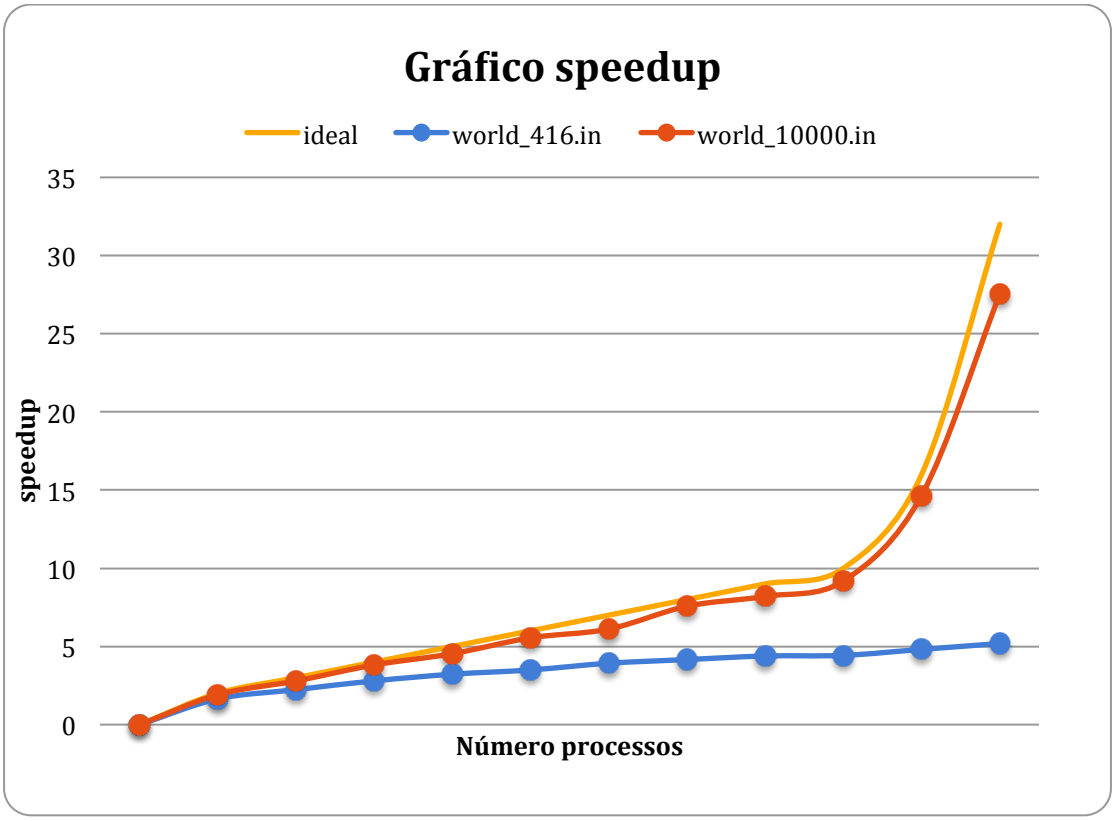
2. Optámos por fazer uma implementação row-wise, e para que o trabalho entre processos fosse equitativo , optámos por dividir o número de linhas do mundo de forma equitativa pelos processos .

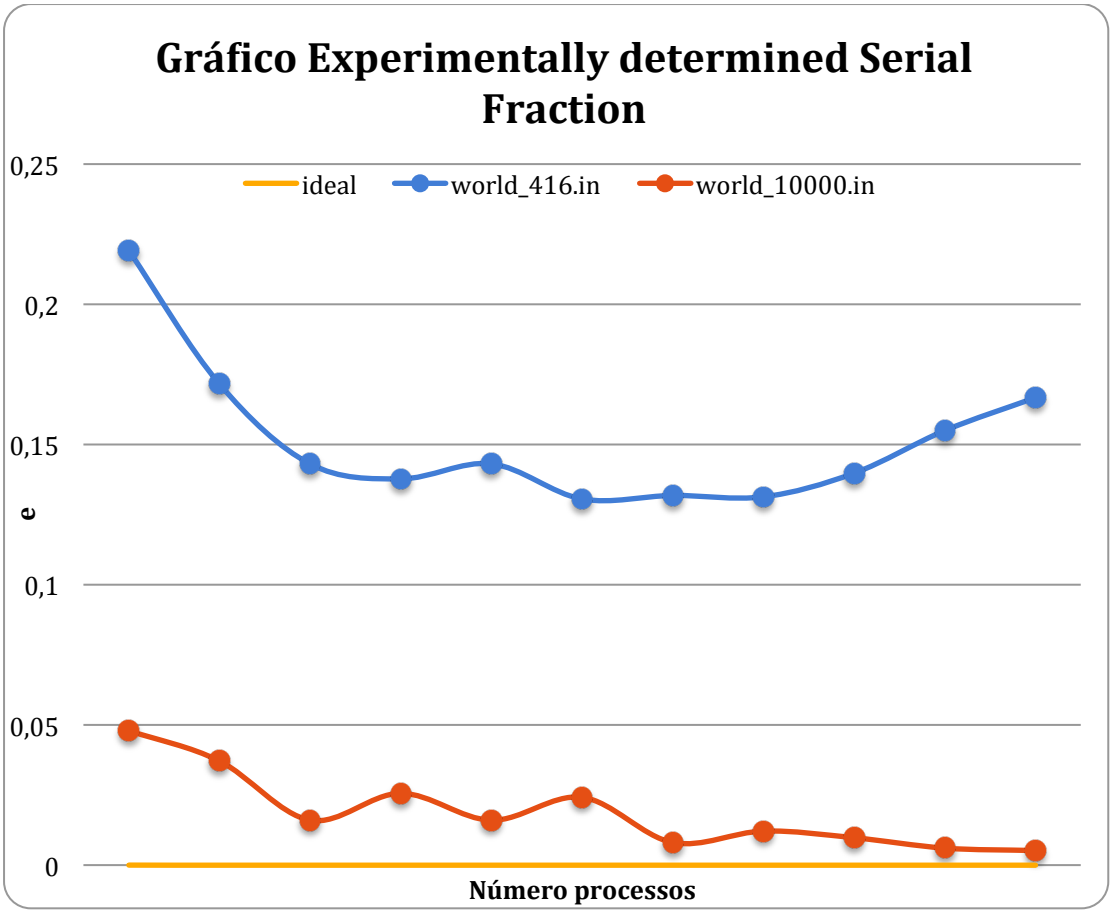
Medimos a performance do nosso programa para um número de processos variável, correndo dois testes, um com matriz de tamanho 416 e outro com tamanho 10000.

Obtivemos os seguintes resultados correndo o teste world_416.in e world_10000.in respectivamente :

NProcs	2	3	4	5	6	7	8	9	10	16	32
Tempo (s)	98,4	67,3	49,2	41,4	33,8	30,7	24,8	22,9	20,4	12,8	6,8
Speedup	1,9	2,8	3,8	4,5	5,6	6,1	7,6	8,2	9,2	14,7	27,5
Serial Fraction	0.05	0.04	0.02	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.01

NProcs	2	3	4	5	6	7	8	9	10	16	32
Tempo (s)	271,5	199,5	159,2	138,2	127,4	113,5	107,1	101,5	100,6	92,6	85,9
Speedup	1,6	2,2	2,8	3,2	3,5	3,9	4,2	4,4	4,4	4,8	5,2
Serial Fraction	0,2	0,2	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,2	0,2





Num geral, pela observação dos resultados , conseguimos um bom speedup . No entanto não conseguimos tirar conclusões objectivas do valor ***e*** , uma vez que , como se pode verificar no gráfico, não obtivemos nem um valor constante nem um valor crescente. Portanto pela observação deste valor não é possível chegar a uma conclusão que revele o porquê de um paralelismo ineficiente e consequentemente o porquê de não termos speedups perfeitos e constantes.