Laboratorio 5

Nicola Agostini, Roberto Cedolin, Lisa Parma

28 Giugno 2019

Domanda 1

Il grafico 1 rappresenta i tempi di esecuzione della versione seriale e parallela dell'algoritmo K-Means rispetto alla variazione del numero di città considerate. All'aumentare del numero di città, gli algoritmi utilizzati per il calcolo dei cluster impiegano più tempo per la computazione: gli speed-up mostrati di seguito fanno emergere un andamento insolito: vi è un aumento di prestazioni fino ad un numero di città pari a 10012; in seguito le performance decrescono leggermente mantenendo uno speed-up compreso fra 1,6 e 1,7.

Speed up con 38183 città: 1.6871667259153929
Speed up con 21789 città: 1.6837972876516774
Speed up con 10012 città: 2.420654911838791
Speed up con 5920 città: 2.427350427350427
Speed up con 2630 città: 2.150442477876106
Speed up con 763 città: 1.825
Speed up con 284 città: 1.5263157894736843
SpeedUp medio: 1.9601039457294398

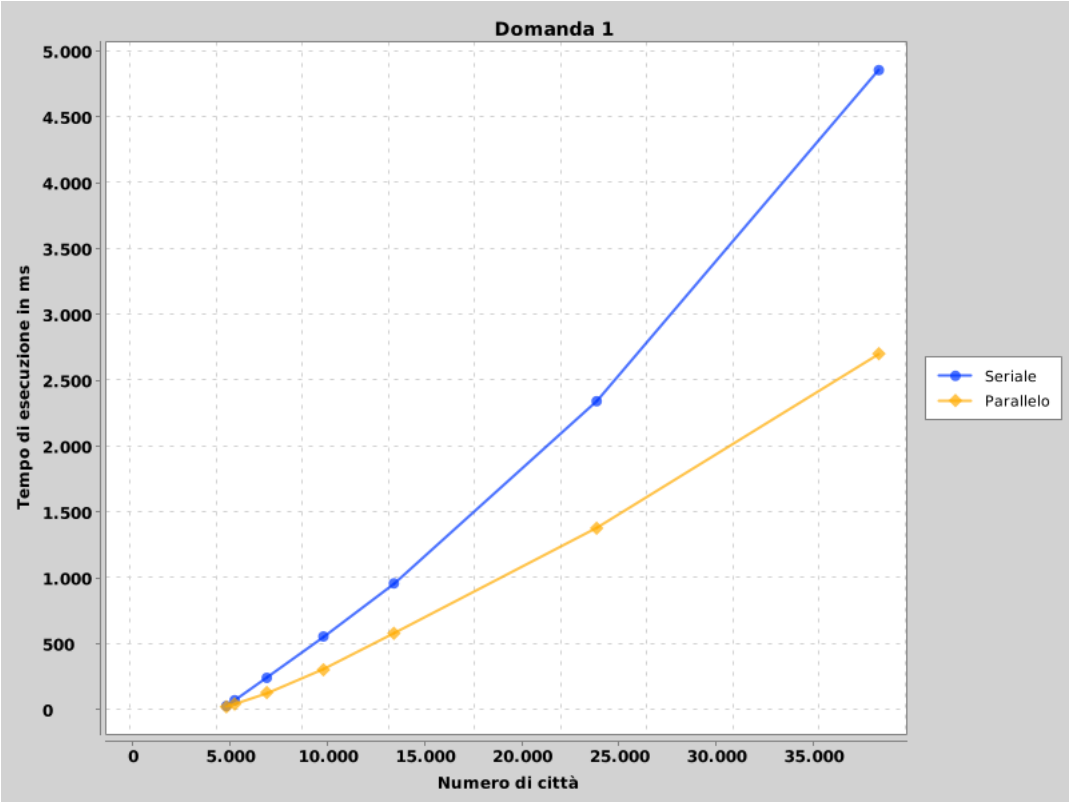

Figura 1: Prestazioni dei due algoritmi al variare del numero di città

Domanda 2

Nel grafico sottostante possiamo vedere come variano i tempi di calcolo per l'algoritmo k-means seriale e parallelo al variare del numero di cluster.
Notiamo che quando il numero dei cluster da creare è inferiore a 40 l'algoritmo seriale ha prestazioni notevolmente migliori rispetto a quello parallelo.
Al crescere del numero di cluster la situazione invece si ribalta: lo speed-up migliora fino a raggiungere un valore di 3,106.

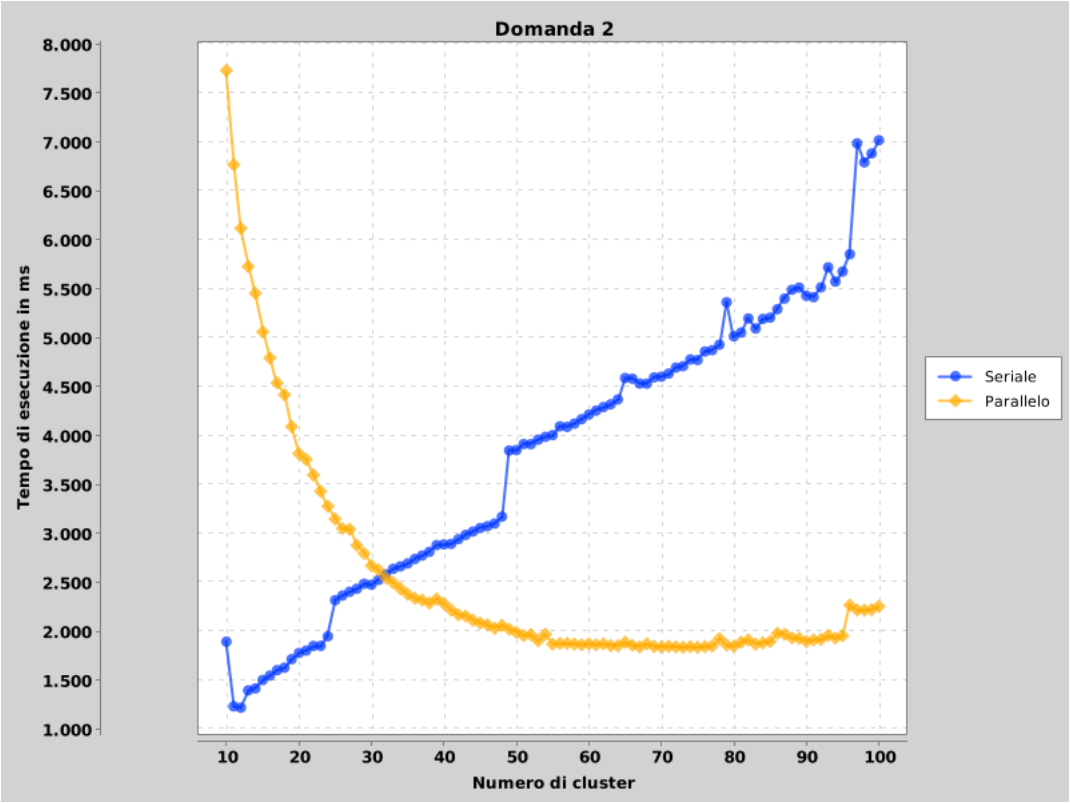

Figura 2: Prestazioni dei due algoritmi al variare del numero di cluster

Cluster	Speed up
10 cluster	0.24557321959415795
11 cluster	0.18293223091687583
12 cluster	0.2000653274538625
13 cluster	0.24450645273805371
14 cluster	0.26048726873053674
15 cluster	0.2976707461508093
16 cluster	0.3236090852260888
17 cluster	0.35409330985915494
18 cluster	0.36943251186977166
19 cluster	0.41996095656417765
20 cluster	0.4668934833813138
21 cluster	0.48086124401913877
22 cluster	0.5151262836525118
23 cluster	0.5404540162980209

24 cluster	0.5948249619482496
25 cluster	0.7368421052631579
26 cluster	0.7755969905135754
27 cluster	0.7904230895375532
28 cluster	0.8466874783211932
29 cluster	0.8896822563370225
30 cluster	0.9271572655958162
31 cluster	0.9619916381603952
32 cluster	1.141121128969033
33 cluster	1.563324011186577
34 cluster	1.92956592956593
35 cluster	1.131656184486373
36 cluster	1.171075085324232
37 cluster	1.1956989247311829
38 cluster	1.2284219703574542
39 cluster	1.2358732876712328
40 cluster	1.2643075578855396
41 cluster	1.304093567251462
42 cluster	1.3549425287356323
43 cluster	1.3840814437760296
44 cluster	1.4275601698914582
45 cluster	1.4645933014354067
46 cluster	1.4857694163048722
47 cluster	1.526031434184676
48 cluster	1.537754114230397
49 cluster	1.8999013806706115
50 cluster	1.9333333333333333
51 cluster	1.9974515800203874
52 cluster	1.9878234398782344
53 cluster	2.75392670157068
54 cluster	2.20748987854251
55 cluster	2.14034151547492
56 cluster	2.1785334750265677
57 cluster	2.1758767268862913
58 cluster	2.2004264392324093
59 cluster	2.232744783306581
60 cluster	2.2488012786361216
61 cluster	2.279828785446763
62 cluster	2.2880724174653886
63 cluster	2.326519634211942
64 cluster	2.3546824542518836
65 cluster	2.425554382259768
66 cluster	2.4621578099838968

67 cluster	2.458288190682557
68 cluster	2.4158679446219384
69 cluster	2.485143165856294
70 cluster	2.497560975609756
71 cluster	2.5048596112311015
72 cluster	2.5446670276123444
73 cluster	2.5597176981541803
74 cluster	2.588744588744589
75 cluster	2.5925122083559415
76 cluster	2.6311519740400215
77 cluster	2.6282327586206895
78 cluster	2.556476683937823
79 cluster	2.8757364756293518
80 cluster	2.7076591154261056
81 cluster	2.671066525871172
82 cluster	2.7141366718831508
83 cluster	2.72145144076841
84 cluster	2.753177966101695
85 cluster	2.7397476340694005
86 cluster	2.6685138539042823
87 cluster	2.73582995951417
88 cluster	2.8344507478081487
89 cluster	2.853154084798345
90 cluster	2.856992639327024
91 cluster	2.827334376630151
92 cluster	2.870514820592824
93 cluster	2.9164544065206317
94 cluster	2.8802271553949406
95 cluster	2.8975012748597653
96 cluster	2.5776506819181697
97 cluster	3.1452991452991452
98 cluster	3.594059405940595
99 cluster	3.920107719928187
100 cluster	3.106590004422822

Speed up medio: 1.81864232560725

Domanda 3

Il grafico 3 mostra le prestazioni dell'algoritmo k-means seriale e parallelo al variare del numero di iterazioni, da 10 a 1000.
Le prestazioni dell'algoritmo parallelo risultano migliori di quelle dell'algoritmo seriale già con poche iterazioni ma, osservando i valori di speed-up, notiamo

che essi continuano a crescere fino al valore di 1.926 poichè aumenta sempre di più il divario tra tempo della computazione parallela rispetto alla computazione seriale.

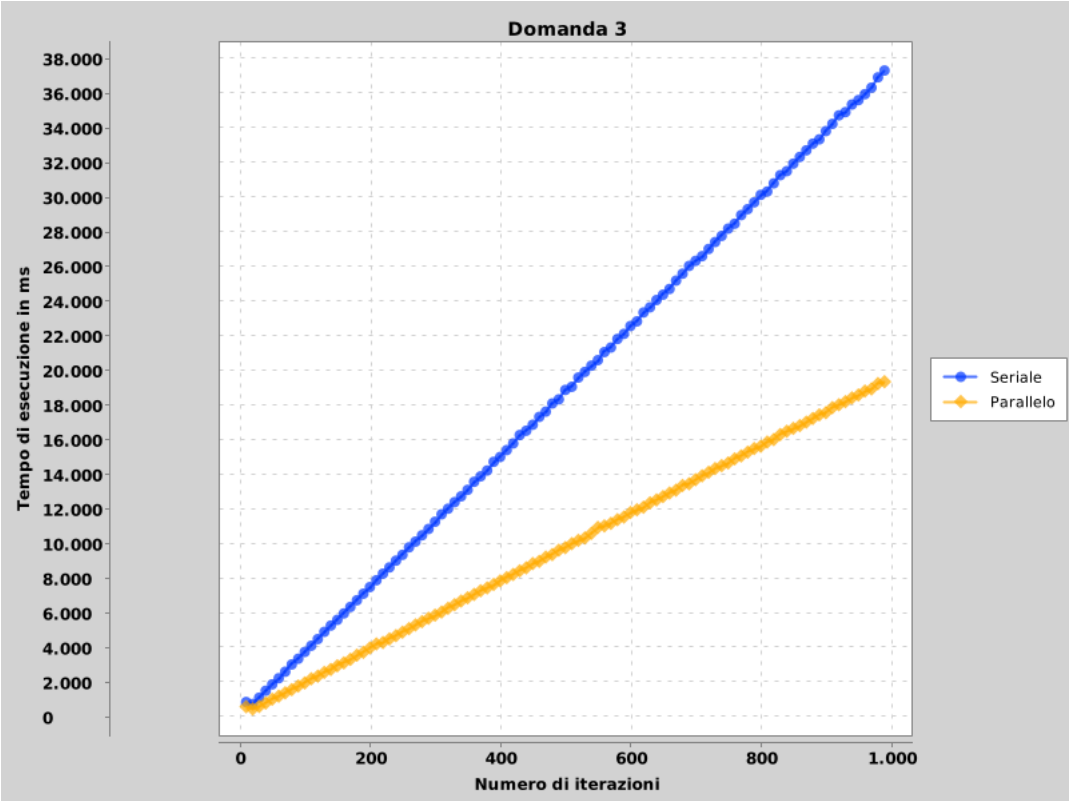

Figura 3: Prestazioni dei due algoritmi al variare del numero di iterazioni

Iterazioni	Speed up
10	1.4242902208201893
20	1.576
30	1.778120184899846ì
40	1.845605700712589
50	1.802249297094658
60	1.835483870967742
70	1.851360781577111
80	1.8980222496909764
90	1.8610503282275712
100	1.8932135728542914
110	1.8494432071269489

120	1.8939457202505219
130	1.8955223880597014
140	1.9105282069708946
150	1.886439545758183
160	1.9035115469788042
170	1.899821640903686
180	1.8793677204658903
190	1.8968716861081654
200	1.8737260750683569
210	1.8645980253878702
220	1.9156293222683265
230	1.9082508250825083
240	1.914853158673146
250	1.9059205190592052
260	1.9126459143968872
270	1.898729446935725
280	1.8993324914306333
290	1.8982223771348903
300	1.9115956727518593
310	1.9256889763779528
320	1.91019955654102
330	1.9079754601226995
340	1.8973711569879697
350	1.9004191357132534
360	1.9138318807984256
370	1.9014469014469015
380	1.9048064085447263
390	1.9202860858257478
400	1.9061431285623813
410	1.910913140311804
420	1.9148421434619571
430	1.9245015925445323
440	1.915240518038853
450	1.9011693276366088
460	1.9244070050986477
470	1.9057173678532902
480	1.9265986828128319
490	1.9002896152254862
500	1.9276469988790381
510	1.9011546884332073
520	1.918409224154778
530	1.923937144509785
540	1.90985994924335

550	1.8792569659442724
560	1.9065775207806288
570	1.9051270619705751
580	1.9131728665207877
590	1.9132049399775455
600	1.9092137488387806
610	1.9064922076839736
620	1.9251728679618043
630	1.910241935483871
640	1.9180458306810948
650	1.9131797523899075
660	1.9081585441085749
670	1.9214715515271537
680	1.9121772869720937
690	1.9313384193763425
700	1.9195544554455446
710	1.9081661891117478
720	1.9099830556340016
730	1.9093121913902218
740	1.9115968928301368
750	1.919942864916338
760	1.9087500836848095
770	1.9199814679992058
780	1.9155517848985186
790	1.914708722240103
800	1.9262096259415293
810	1.9090451945439688
820	1.9215002180549499
830	1.9158712677435144
840	1.9081780788773288
850	1.9149649679621534
860	1.919805445162821
870	1.91969048596049
880	1.9185292414591777
890	1.9096161546822263
900	1.9236931818181817
910	1.915232454179705
920	1.9249211007142462
930	1.9163879598662208
940	1.9173302995828594
950	1.9137273849787941
960	1.9091825367354518
970	1.9165173395172341

980	1.9166666666666667
990	1.9263098184818481

SpeedUp: 1.896272388061413

Domanda 4

Nel grafico 4 vediamo il variare delle prestazioni dell'algoritmo k-means parallelo variando la soglia di cutoff in cui viene utilizzata una procedura seriale.
Notiamo che le prestazioni risultano estremamente pessime nel caso in cui venga considerata una soglia di 2, caso in cui l'input viene diviso in due metà e poi si procede serialmente sulle due metà.
Utilizzando un cutoff con un valore tra 70 e 100 invece si ottimizzano le prestazioni dell'algoritmo.

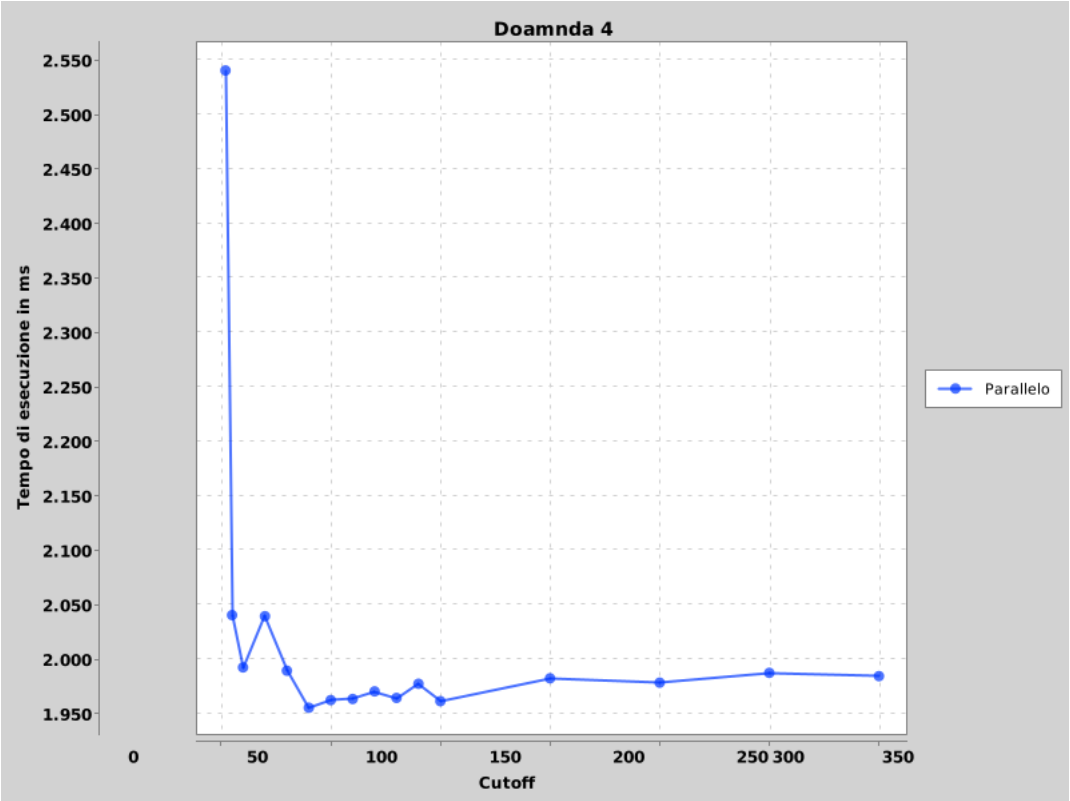

Figura 4: Prestazioni dell'algoritmo k-means parallelo al variare del parametro cutoff

Domanda 5

Prestazioni hardware: processore i5 4300u 1.9 GHz con 4 core.