2022/08/22 11:05 EPCPGrowth-ad

Advanced Tutorial on Implementing EPCPGrowth Algorithm

In this tutorial, we explain how the EPCPGrowth algorithm can be implemented by varying the minimum support values

Step 1: Import the EPCPGrowth algorithm and pandas data frame

```
In [1]: from PAMI.periodicCorrelatedPattern import EPCPGrowth as alg
import pandas as pd
```

Step 2: Specify the following input parameters

```
inputFile = 'temporal_T10I4D100K.csv'
seperator='\forall'
maxmunPeriodCount=5000
minimumSupportCountList = [100, 150, 200, 250, 300]
#minimumSupport can also specified between 0 to 1. E.g., minSupList = [0.005, 0.006, minAllConfCount=0.5
maxPerAllmaxPerConfCount=0.5
result = pd. DataFrame(columns=['algorithm', 'minSup', 'minAllConf', 'maxPer', 'maxPerAl #initialize a data frame to store the results of EPCPGrowth algorithm
```

Step 3: Execute the EPCPGrowth algorithm using a for loop

```
algorithm = 'EPCPGrowth' #specify the algorithm name
for minSupCount in minimumSupportCountList:
   obj = alg. EPCPGrowth('temporal_T10I4D100K.csv', minSup=minSupCount, minAllConf=mi
   obj. startMine()
   neighborFile='T10_utility_neighbour.txt'
   #store the results in the data frame
   result.loc[result.shape[0]] = [algorithm, minSupCount, minAllConfCount, maxmunPeri
```

2022/08/22 11:05 EPCPGrowth-ad

```
[21] 4248 219 1 1
[20] 4258 236 1 1
[19] 4309 327 1 1
[18] 4388 224 1 1
[17] 4438 185 1 1
[16] 4511 232 1 1
[15] 4559 189 1 1
[14] 4629 173 1 1
[13] 4681 201 1 1
[12] 4902 197 1 1
[11] 4973 174 1 1
[10] 4993 143 1 1
[9] 5057 159 1 1
[8] 5102 186 1 1
[7] 5375 204 1 1
[6] 5408 184 1 1
[5] 5835 140 1 1
[4] 5845 136 1 1
[3] 6265 112 1 1
[2] 6810 146 1 1
[1] 7057 133 1 1
[0] 7828 113 1 1
Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
150 0.5 5000 0.5
[765] 150 3742 1 1
[764] 151 3896 1 1
[763] 154 3010 1 1
[762] 155 3200 1 1
[761] 156 3836 1 1
[760] 158 3993 1 1
[759] 170 3363 1 1
[758] 171 3166 1 1
[757] 172 3492 1 1
[756] 173 3380 1 1
[755] 175 2844 1 1
[754] 179 3317 1 1
[753] 179 2927 1 1
[752] 181 3315 1 1
[751] 183 4329 1 1
[750] 184 2638 1 1
[749] 185 3478 1 1
[748] 188 2799 1 1
[747] 191 3027 1 1
[746] 192 3113 1 1
[745] 193 3523 1 1
[744] 194 3229 1 1
[743] 197 2393 1 1
[742] 197 2997 1 1
[741] 199 2515 1 1
[740] 205 2420 1 1
[739] 210 3495 1 1
[738] 211 2673 1 1
[737] 212 2321 1 1
[736] 214 2443 1 1
[735] 217 1915 1 1
[734] 220 3376 1 1
[733] 222 2562 1 1
[732] 223 2262 1 1
[731] 224 2504 1 1
[730] 225 3944 1 1
[729] 229 2738 1 1
[728] 234 2208 1 1
[727] 236 2046 1 1
```

[726] 236 2711 1 1

2022/08/22 11:05 EPCPGrowth-ad

```
[21] 4248 219 1 1
[20] 4258 236 1 1
[19] 4309 327 1 1
[18] 4388 224 1 1
[17] 4438 185 1 1
[16] 4511 232 1 1
[15] 4559 189 1 1
[14] 4629 173 1 1
[13] 4681 201 1 1
[12] 4902 197 1 1
[11] 4973 174 1 1
[10] 4993 143 1 1
[9] 5057 159 1 1
[8] 5102 186 1 1
[7] 5375 204 1 1
[6] 5408 184 1 1
[5] 5835 140 1 1
[4] 5845 136 1 1
[3] 6265 112 1 1
[2] 6810 146 1 1
[1] 7057 133 1 1
[0] 7828 113 1 1
Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
200 0.5 5000 0.5
[740] 205 2420 1 1
[739] 210 3495 1 1
[738] 211 2673 1 1
[737] 212 2321 1 1
[736] 214 2443 1 1
[735] 217 1915 1 1
[734] 220 3376 1 1
[733] 222 2562 1 1
[732] 223 2262 1 1
[731] 224 2504 1 1
[730] 225 3944 1 1
[729] 229 2738 1 1
[728] 234 2208 1 1
[727] 236 2046 1 1
[726] 236 2711 1 1
[725] 237 2389 1 1
[724] 239 2383 1 1
[723] 240 2786 1 1
[722] 240 2383 1 1
[721] 241 2005 1 1
[720] 243 3400 1 1
[719] 245 2221 1 1
[718] 247 2185 1 1
[717] 248 3175 1 1
[716] 257 2252 1 1
[715] 257 2169 1 1
[714] 260 1866 1 1
[713] 266 1749 1 1
[712] 266 2383 1 1
[711] 269 3209 1 1
[710] 270 2237 1 1
[709] 270 2140 1 1
[708] 270 3158 1 1
[707] 272 2442 1 1
[706] 272 1980 1 1
[705] 274 1719 1 1
[704] 280 2021 1 1
[703] 281 2115 1 1
[702] 283 2046 1 1
[701] 284 2532 1 1
```

```
[60] 2976 284 1 1
[59] 3012 275 1 1
[58] 3014 228 1 1
[57] 3043 257 1 1
[56] 3044 254 1 1
[55] 3053 298 1 1
[54] 3063 407 1 1
[53] 3069 261 1 1
[52] 3085 271 1 1
[51] 3090 355 1 1
[50] 3103 254 1 1
[49] 3134 263 1 1
[48] 3135 294 1 1
[47] 3151 259 1 1
[46] 3219 299 1 1
[45] 3281 290 1 1
[44] 3361 302 1 1
[43] 3385 255 1 1
[42] 3415 237 1 1
[41] 3420 242 1 1
[40] 3470 223 1 1
[39] 3507 231 1 1
[38] 3605 231 1 1
[37] 3626 264 1 1
[36] 3649 223 1 1
[35] 3667 277 1 1
[34] 3686 287 1 1
[33] 3690 219 1 1
[32] 3710 288 1 1
[31] 3735 271 1 1
[30] 3771 228 1 1
[29] 3864 182 1 1
[28] 3883 206 1 1
[27] 3921 190 1 1
[26] 3982 283 1 1
[25] 4037 242 1 1
[24] 4082 229 1 1
[23] 4132 182 1 1
[22] 4137 243 1 1
[21] 4248 219 1 1
[20] 4258 236 1 1
[19] 4309 327 1 1
[18] 4388 224 1 1
[17] 4438 185 1 1
[16] 4511 232 1 1
[15] 4559 189 1 1
[14] 4629 173 1 1
[13] 4681 201 1 1
[12] 4902 197 1 1
[11] 4973 174 1 1
[10] 4993 143 1 1
[9] 5057 159 1 1
[8] 5102 186 1 1
[7] 5375 204 1 1
[6] 5408 184 1 1
[5] 5835 140 1 1
[4] 5845 136 1 1
[3] 6265 112 1 1
[2] 6810 146 1 1
[1] 7057 133 1 1
Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
250 0.5 5000 0.5
[716] 257 2252 1 1
```

2022/08/22 11:05 EPCPGrowth-ad

```
[11] 4973 174 1 1
[10] 4993 143 1 1
[9] 5057 159 1 1
[8] 5102 186 1 1
[7] 5375 204 1 1
[6] 5408 184 1 1
[5] 5835 140 1 1
[4] 5845 136 1 1
[3] 6265 112 1 1
[2] 6810 146 1 1
[1] 7057 133 1 1
[0] 7828 113 1 1
Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
300 0.5 5000 0.5
[691] 302 2700 1 1
[690] 303 1548 1 1
[689] 305 1619 1 1
[688] 306 1972 1 1
[687] 308 2112 1 1
[686] 309 2326 1 1
[685] 312 1863 1 1
[684] 313 2261 1 1
[683] 314 1784 1 1
[682] 316 2622 1 1
[681] 317 1561 1 1
[680] 318 1771 1 1
[679] 318 2119 1 1
[678] 318 2326 1 1
[677] 319 2233 1 1
[676] 323 1676 1 1
[675] 324 2234 1 1
[674] 325 1371 1 1
[673] 326 2069 1 1
[672] 327 1654 1 1
[671] 327 1678 1 1
[670] 329 2118 1 1
[669] 331 2004 1 1
[668] 337 2132 1 1
[667] 342 2793 1 1
[666] 344 1746 1 1
[665] 346 1869 1 1
[664] 346 1536 1 1
[663] 347 1937 1 1
[662] 347 1579 1 1
[661] 348 2642 1 1
[660] 353 2085 1 1
[659] 354 2535 1 1
[658] 356 2081 1 1
[657] 357 2123 1 1
[656] 358 1757 1 1
[655] 360 1329 1 1
[654] 362 1458 1 1
[653] 362 1570 1 1
[652] 363 1625 1 1
[651] 365 1771 1 1
[650] 366 1613 1 1
[649] 370 2016 1 1
[648] 371 1781 1 1
[647] 372 1811 1 1
[646] 373 1789 1 1
[645] 376 2200 1 1
[644] 381 1828 1 1
[643] 382 2758 1 1
```

[642] 383 2107 1 1

2022/08/22 11:05 EPCPGrowth-ad

[1] 7057 133 1 1 [0] 7828 113 1 1

Periodic Frequent patterns were generated successfully using PFPGrowth algorithm

In [4]: print(result)

	algorithm	minSup	minAllConf	maxPer	maxPerAllConf	patterns	runtime	¥
0	EPCPGrowth	100	0.5	5000	0.5	0	5.082180	
1	EPCPGrowth	150	0. 5	5000	0. 5	0	4.664614	
2	EPCPGrowth	200	0.5	5000	0. 5	0	4.866936	
3	EPCPGrowth	250	0. 5	5000	0. 5	0	5. 025409	
4	EPCPGrowth	300	0. 5	5000	0.5	0	4. 544416	

memory

- 0 572940288
- 1 572137472
- 2 570265600
- 3 567861248
- 4 564633600

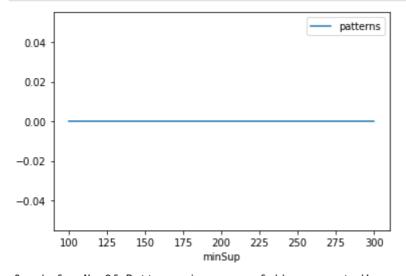
Step 5: Visualizing the results

Step 5.1 Importing the plot library

In [5]: from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt

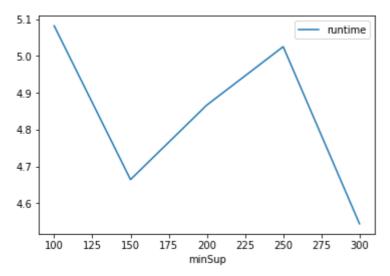
Step 5.2. Plotting the number of patterns

In [6]: ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()

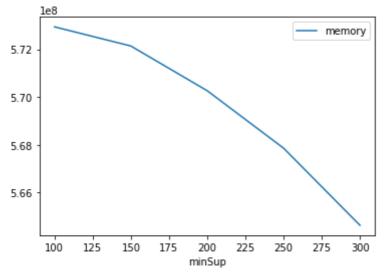


Graph for No Of Patterns is successfully generated!

2022/08/22 11:05 EPCPGrowth-ad



Graph for Runtime taken is successfully generated!



Graph for memory consumption is successfully generated!

Step 6: Saving the results as latex files

In [7]: from PAMI.extras.graph import generateLatexFileFromDataFrame as gdf gdf.generateLatexCode(result)

Latex files generated successfully