DataMining-GSP

Hash^2 GSP algorithm in data mining. This work is done by Keren Zhou, Qiang Li, Gang zeng, and Wuzhao Zhang. You can contact us by sending an email to `kerenzhou@outlook.com`.

If you want to learn the whole structure of our work, please read `report.pdf`.

Features:

General:

1. Support max gap and min gap.
2. Hash-tree, with `state pruning` and `adaptive adjust hash` method.
3. Time-list matching algorithm.

Specific:

1. There are two types of input data, you can switch by specifying the `-file_type` argument, where `0` represents the common input data, and `1` represents input data the same as the spmf project. We provide some of the test data in the `data` directory.

   common input data:

   Examples are `100.txt`, `seq.txt` and `gen.data`, the format is as follow:

   sequence_id number_of_items item_id item_id …

   sequence_id number_of_items item_id item_id …

   sequence_id number_of_items item_id item_id …

   Where each line represents an itemset, consist of distinct items.

   spmf input data:

   Exapmles are `BMS1_spmf.txt`, `kosarak10k.txt`, `kosarak25k.txt`, and `small.txt`, the format is as follow:

   item_id -1 item_id item_id -1 … -1 item_id -2

   item_id -1 item_id item_id -1 … -1 item_id -2

   item_id -1 item_id item_id -1 … -1 item_id -2

   Where each item_set is seperated by `-1` , and `-2` indicates the end of a sequence.

2. There is an experimental `data_provider` in the source code, but the features are not fully developed. Currently it supports the *Gaussian distribution* and *Even distribution*. You must modify the parameters in the file if you want to use it, and recompiliation is also required.

Usage:

```
./gsp -i [file_name] -t [support: float] -sequNUM [unsigned int32] -min
[unsigned int32] -max [unsigned int32] -eventNUM [unsigned int32] -
file_type [0:common, 1:spmf]
```

For instance, if you want to use `100.txt` in `../data/` directory, you should use the following command:

```
./gsp -i ../data/100.txt -t 0.5 -sequNUM 100 -min 2 -max 4 -eventNUM 100 -
file_type 0
```

Advantages:

1. Up to 10 times faster than the original GSP algorithm encountering large data. Faster than Prefixspan algorithm. The experiment result is shown in the following table.

kosarak10k.txt 10000 sequences

algorithm	dataset	support	time(s)
hash^2 gsp	kosarak10k	0.05	0.133
gsp	kosarak10k	0.05	0.235
prefix	kosarak10k	0.05	0.232

hash^2 gsp	kosarak10k	0.04	0.099
gsp	kosarak10k	0.04	0.382
prefix	kosarak10k	0.04	0.23
hash^2 gsp	kosarak10k	0.03	0.15
gsp	kosarak10k	0.03	0.454
prefix	kosarak10k	0.03	0.24
hash^2 gsp	kosarak10k	0.02	0.207
gsp	kosarak10k	0.02	1.217
prefix	kosarak10k	0.02	0.272
hash^2 gsp	kosarak10k	0.01	0.484
gsp	kosarak10k	0.01	4.373
prefix	kosarak10k	0.01	0.372

kosarak25k.txt 25000 sequences

algorithm	**dataset**	**support**	**time(s)**
hash^2 gsp	kosarak25k	0.05	0.299
gsp	kosarak25k	0.05	0.436
prefix	kosarak25k	0.05	0.345
hash^2 gsp	kosarak25k	0.04	0.233
gsp	kosarak25k	0.04	0.631
prefix	kosarak25k	0.04	0.42
hash^2 gsp	kosarak25k	0.03	0.323
gsp	kosarak25k	0.03	0.921
prefix	kosarak25k	0.03	0.425
hash^2 gsp	kosarak25k	0.02	0.339
gsp	kosarak25k	0.02	2.22
prefix	kosarak25k	0.02	0.591

prefix	kosarak25k	0.02	0.591
hash^2 gsp	kosarak25k	0.01	0.611
gsp	kosarak25k	0.01	8.95
prefix	kosarak25k	0.01	0.7

BMS1_spmf60k.txt 60000 sequences

algorithm	dataset	support	time(s)
hash^2 gsp	BMS1_spmf60k	0.05	0.093
gsp	BMS1_spmf60k	0.05	0.188
prefix	BMS1_spmf60k	0.05	0.188
hash^2 gsp	BMS1_spmf60k	0.04	0.078
gsp	BMS1_spmf60k	0.04	0.796
prefix	BMS1_spmf60k	0.04	0.191
hash^2 gsp	BMS1_spmf60k	0.03	0.109
gsp	BMS1_spmf60k	0.03	0.797
prefix	BMS1_spmf60k	0.03	0.235
hash^2 gsp	BMS1_spmf60k	0.02	0.124
gsp	BMS1_spmf60k	0.02	2.58
prefix	BMS1_spmf60k	0.02	0.312
hash^2 gsp	BMS1_spmf60k	0.01	0.453
gsp	BMS1_spmf60k	0.01	22.9
prefix	BMS1_spmf60k	0.01	0.485

Progress:

12/9/2014:

1. Adaptive hash-tree branches functions.
2. Multi-thread.
3. Fix data generator.