External Memory Spanning Forests and Connected Components

Dominik Schultes

September 2003

Abstract

The external Minimum Spanning Trees implementation presented in [Sch03] can be easily modified in order to determine a Spanning Forest of an unweighted graph. Furthermore, an implementation of the Connected Components algorithm presented in [SM] can be integrated.

The results of the test runs show that computing a spanning forest or the connected components of an unweighted graph is about twice as fast as the computation of a minimum spanning tree of an weighted graph.

1 Spanning Forest

The following modifications of the MST implementation have been done:

- The attribute `weight` has been removed from the data structures (`EdgeWithoutSource`).
- During the node reduction phase, for each node the edge that leads to the node with the smallest index is selected instead of the shortest edge.
  (`REWS_SparingStack::determineMinEdge()`)
- The base case (Kruskal's algorithm) has been simplified so that sorting is skipped.

Currently, only the Buckets implementation is supported.

2 Connected Components

We implemented the Connected Components algorithm presented in [SM, p. 7]. The input is an unweighted graph (given as a list of edges), the output is a list of edges so that there is an edge from each node to the canonical node of the corresponding connected component. The canonical node of a connected component is not necessarily the node with the smallest index because of two reasons: first, the base case dispenses with sorting, each root of a tree of the union-find data structure becomes a canonical node, and the root does not have to be the node with the smallest index; second, randomization prevents that a certain node is selected as canonical node. Still, $\min(v)$ is used to denote the (preliminary) canonical node of the connected component that the node v belongs to.

The integration of the Connected Components algorithm into the existing Spanning Forests implementation has been done in the following way. We already had b buckets and upper bounds $u_0 < u_1 < u_2 < \ldots < u_b, u_0 = 0, u_b \geq n$, so that bucket $i \in \{1, \ldots, b\}$ contains the edges of the nodes with the identifiers from $u_{i-1} + 1$ to u_i in an arbitrary sequence. Additionally, we have now b questions buckets and b answers buckets with the same upper bounds. A *question* is a tuple $(v, \min(v))$ that represents the assignment of a node v to a *preliminary* canonical node $\min(v)$. An *answer* is a tuple $(v, \min(v))$ that represents the assignment of a node to an *ultimate* canonical node. We have a function $f : \{1, \ldots, n\} \to \{1, \ldots, b\}$ that maps a node ID to the corresponding bucket ID according to the upper bounds.

During the Processing of each Bucket i

if the list of v is empty **then** $\min(v) := v$
else $\min(v) :=$ smallest entry in the list of v;

After the Processing of each Bucket i

for $v := u_{i-1} + 1$ **to** u_i **do**
    if $\min(v) \leq u_{i-1}$ **then**
        add $(v, \min(v))$ to `Questions`$[f(\min(v))]$;
    else
        $\min(v) := \min(\min(v))$;
        if $\min(v) \leq u_{i-1}$ **then**
            add $(v, \min(v))$ to `Questions`$[f(\min(v))]$;
        else
            add $(v, \min(v))$ to `Answers`$[f(v)]$;

Post-Processing

for $i := 1$ **to** b **do**
    read `Answers`$[i]$;
    foreach $(v, \min(v)) \in$ `Questions`$[i]$ **do**
        $\min(v) := \min(\min(v))$;
        add $(v, \min(v))$ to `Answers`$[f(v)]$;
    write `Answers`$[i]$ to result;

The first answers bucket is an exception as it is substituted by the union-find data structure of the base case. For each node v in the union-find data structure, there is an implicit answer $(v, \text{find}(v))$.

Implementation Details

In order to reduce the memory usage of the new data structures during the first and second phase (= node reduction and base case), the creation of the answers buckets is postponed to the third phase (post-processing). During the first and second phase only one temporary answers bucket is used. In the third phase this answers bucket is read and the answers are distributed to the created buckets. This causes an increase of the running time by less than 2 %.

Only Connected Components, No Spanning Forest

We can accelerate the computation of the connected components by omitting the computation of a spanning forest. In this case we can dispense with the attributes `original source` and `original target` as we do not have to store this information. It is sufficient to invert the randomization before the result is printed. Theoretically, this is always possible as the randomization function is a bijection. Practically, this is even quite easy because it is simple to invert the Feistel permutation.

Due to this measure the I/O volume is reduced and the memory usage of the internal buckets is decreased so that the number of internal buckets and the size of the write buffers can be increased.

3 Evaluation

For the evaluation we used the same graph families and sizes as for the evaluation of the MST implementation [Sch03, ch. 5]. The sole difference is the fact that we deal with unweighted graphs now.

For each graph instance, there is one test run that computes only a spanning forest (Table 1), one run that computes only the connected components (Table 2) and one run that computes both a spanning forest and the connected components (Table 3).

References

[Sch03] D. Schultes. External memory minimum spanning trees. Bachelor thesis, Max-Planck-Institut f. Informatik and Saarland University, 2003.

[SM] J. Sibeyn and U. Meyer. External connected components and beyond. *unpublished*.

type	$n/10^6$	$m/10^6$	$t[s]$	$t/m[\mu s]$	$p/10^6$	$p/E(p)$	d/m
grid	40	80	26	0.32			
grid	80	160	52	0.32			
grid	160	320	104	0.32			
grid	320	640	1 152	1.80	686	71 %	2 %
grid	640	1 280	2 445	1.91	2 045	55 %	7 %
grid	1 280	2 560	5 007	1.96	4 923	45 %	13 %
random	40	80	81	1.01			
random	80	160	182	1.14			
random	160	320	405	1.26			
random	320	640	1 347	2.10	680	70 %	0 %
random	640	1 280	2 961	2.31	2 000	54 %	0 %
random	1 280	2 560	6 649	2.60	4 773	43 %	0 %
random	20	80	63	0.79			
random	40	160	135	0.85			
random	80	320	304	0.95			
random	160	640	671	1.05			
random	320	1 280	2 739	2.14	1 361	70 %	0 %
random	640	2 560	6 284	2.45	3 962	53 %	0 %
random	10	80	56	0.69			
random	20	160	113	0.71			
random	40	320	241	0.75			
random	80	640	555	0.87			
random	160	1 280	1 195	0.93			
random	320	2 560	5 848	2.28	2 720	70 %	0 %
geometric	40	75	61	0.82			
geometric	80	149	136	0.91			
geometric	160	298	302	1.01			
geometric	320	596	1 137	1.91	633	70 %	5 %
geometric	640	1 190	2 226	1.87	1 824	53 %	11 %
geometric	1 280	2 390	4 490	1.88	4 204	41 %	14 %
geometric	20	71	43	0.61			
geometric	40	141	93	0.66			
geometric	80	282	195	0.69			
geometric	160	564	480	0.85			
geometric	320	1 130	1 981	1.76	1 200	70 %	7 %
geometric	640	2 260	4 142	1.84	3 476	53 %	16 %
geometric	10	68	33	0.49			
geometric	20	135	69	0.51			
geometric	40	270	142	0.52			
geometric	80	540	311	0.58			
geometric	160	1 080	669	0.62			
geometric	320	2 160	3 493	1.62	2 296	70 %	8 %

n nodes, m edges, t elapsed time, p processed edges, $E(p)$ expected value of p according to [SM, Corollary 1], d duplicates (parallel edges) removed

Table 1: Spanning Forests

type	$n/10^6$	$m/10^6$	$t[s]$	$t/m[\mu s]$	$p/10^6$	$p/E(p)$	d/m
grid	40	80	31	0.39			
grid	80	160	62	0.39			
grid	160	320	128	0.40			
grid	320	640	1 186	1.85	728	69 %	2 %
grid	640	1 280	2 523	1.97	2 095	54 %	7 %
grid	1 280	2 560	5 109	2.00	4 986	44 %	14 %
random	40	80	87	1.09			
random	80	160	198	1.24			
random	160	320	431	1.35			
random	320	640	1 335	2.09	721	68 %	0 %
random	640	1 280	2 905	2.27	2 048	53 %	0 %
random	1 280	2 560	6 259	2.44	4 838	43 %	0 %
random	20	80	65	0.81			
random	40	160	140	0.88			
random	80	320	315	0.98			
random	160	640	694	1.08			
random	320	1 280	2 455	1.92	1 442	68 %	0 %
random	640	2 560	5 511	2.15	4 050	52 %	0 %
random	10	80	57	0.71			
random	20	160	115	0.72			
random	40	320	247	0.77			
random	80	640	551	0.86			
random	160	1 280	1 211	0.95			
random	320	2 560	5 036	1.97	2 880	68 %	0 %
geometric	40	75	73	0.98			
geometric	80	149	157	1.05			
geometric	160	298	343	1.15			
geometric	320	596	1 188	1.99	670	68 %	6 %
geometric	640	1 190	2 367	1.98	1 861	51 %	11 %
geometric	1 280	2 390	4 734	1.98	4 241	40 %	15 %
geometric	20	71	47	0.66			
geometric	40	141	98	0.69			
geometric	80	282	210	0.75			
geometric	160	564	453	0.80			
geometric	320	1 130	1 866	1.65	1 270	68 %	8 %
geometric	640	2 260	3 860	1.71	3 550	52 %	17 %
geometric	10	68	35	0.51			
geometric	20	135	72	0.53			
geometric	40	270	147	0.55			
geometric	80	540	321	0.59			
geometric	160	1 080	694	0.64			
geometric	320	2 160	3 088	1.43	2 431	68 %	10 %

Table 2: Connected Components

type	$n/10^6$	$m/10^6$	$t[s]$	$t/m[\mu s]$	$p/10^6$	$p/E(p)$	d/m
grid	40	80	31	0.39			
grid	80	160	63	0.39			
grid	160	320	125	0.39			
grid	320	640	1 375	2.15	728	69 %	2 %
grid	640	1 280	2 990	2.34	2 095	54 %	7 %
grid	1 280	2 560	6 219	2.43	4 986	44 %	14 %
random	40	80	88	1.10			
random	80	160	195	1.22			
random	160	320	435	1.36			
random	320	640	1 555	2.43	721	68 %	0 %
random	640	1 280	3 456	2.70	2 048	53 %	0 %
random	1 280	2 560	7 645	2.99	4 838	43 %	0 %
random	20	80	67	0.83			
random	40	160	142	0.88			
random	80	320	316	0.99			
random	160	640	698	1.09			
random	320	1 280	2 974	2.32	1 442	68 %	0 %
random	640	2 560	6 794	2.65	4 050	52 %	0 %
random	10	80	58	0.72			
random	20	160	118	0.74			
random	40	320	250	0.78			
random	80	640	557	0.87			
random	160	1 280	1 223	0.96			
random	320	2 560	6 175	2.41	2 880	68 %	0 %
geometric	40	75	73	0.98			
geometric	80	149	158	1.06			
geometric	160	298	343	1.15			
geometric	320	596	1 344	2.25	670	68 %	6 %
geometric	640	1 190	2 737	2.30	1 861	51 %	11 %
geometric	1 280	2 390	5 599	2.35	4 241	40 %	15 %
geometric	20	71	47	0.67			
geometric	40	141	98	0.70			
geometric	80	282	210	0.75			
geometric	160	564	453	0.80			
geometric	320	1 130	2 205	1.95	1 270	68 %	8 %
geometric	640	2 260	4 706	2.08	3 550	52 %	17 %
geometric	10	68	35	0.51			
geometric	20	135	72	0.53			
geometric	40	270	147	0.54			
geometric	80	540	320	0.59			
geometric	160	1 080	692	0.64			
geometric	320	2 160	3 798	1.76	2 431	68 %	10 %

Table 3: Spanning Forests and Connected Components