SPFA (Shortest Path Faster Algorithm) 之分析以及優化探討

SPFA (Shortest Path Faster Algorithm) analysis and optimization discussion

指導教授	莊坤達教授
專題成員	林冠圻
開發工具	C++
測試環境	FreeBSD 13.1

一、簡介：

從高中開始便持續接觸資訊競賽，其中對於最短路演算法有些許心得，特別是 SPFA，其作為 Bellman-Ford 演算法的擴充延伸並得到大幅度的優化，平均執行時間與 Dijkstra 相當甚至更快。

但因為 SPFA 是由 Bellman-Ford 優化改進而成，最差情況的時間複雜度仍與 Bellman-Ford 相當，因此在某些情況下反而是 Dijkstra 優於 SPFA。

不同於 Bellman-Ford，SPFA 並不會盲目的嘗試鬆弛所有節點，而是維護一個備選節點的佇列，並且僅在該節點被鬆弛後才會加入佇列。這個動作會不斷重複，直到佇列中沒有任何可鬆弛節點。

另外因 SPFA 會容易被構造出特殊測資使得其時間複雜度與 Bellman-Ford 相當，測試時會另外加入不同的優化方法一起比較，優化方法如下：

1. Random Shuffle：開始尋找最短路前，先將所有的邊隨機打亂。
2. In Queue：將節點加入佇列前，會確認是否在佇列中。
3. Small Label First (SLF)：將節點 v 加入佇列前，會將 d(v) 以及 d(q.front()) 做比較，d(v) 較小時會將 d(v) 加入佇列前端，反之加入後端。為了處理此操作，會將佇列（queue）改成雙向佇列（deque）。
4. Large Label Last (LLL)：在每次取出佇列前端時，會先更新佇列，使得前端的節點的距離總是小於佇列中的平均，並且將大於平均的節點放到佇列後端。
5. D’Esopo-Pape：將節點加入佇列前，會先檢查此節點是否鬆弛過，如果還未被鬆弛過則加入佇列後端，反之會加入前端，例如節點 u 被重複鬆弛時，u 周圍的節點也可能需要重新鬆弛，假設有一個節點 v 與 u 相連並且在佇列中，如果 d(v) 的值會因為 u 被重複鬆弛而影響，那在處理與 v 之前先將 d(v) 的值更新一次，所需要花費的時間會比重複處理兩次 v、以及重複處理延伸出去更多的結點所需要花費的時間更少。

6. SLF and LLL：同時使用 SLF 優化以及 LLL 優化。
7. LLL and D’Esopo-Pape：同時使用 LLL 優化以及 D’Esopo-Pape 優化。

前兩種不太算是優化方法，而是 SPFA 的實作技巧，而後面五種則是比較常用的幾種優化方法。

其中 SLF, LLL, D’Esopo-Pape, SLF and LLL, LLL and D’Esopo-Pape 都有使用 In Queue 以及 Random Shuffle 的技巧。

另外由於 D’Esopo-Pape 以及 SLF 都是在將節點加入佇列前檢查，符合某些條件則加入至前端或後端，但由於條件不相容無法放在一起使用。

二、測試結果：

測資一共分成五種前四種分別為（節點數量/雙向邊數量）：10/250, 1000/2500, 10000/25000, 100000/2500000，而最後一項是使 SPFA 退化的測資。

時間 (ms)	10/250	1000/2500	10000/25000	100000/250000	counter-SPFA
SPFA	11.4	5.82	38.1	357.68	5571.8
SPFA-inQ	4	4.95	26.2	339.96	24752.9
SPFA-random	0.5	0.99	23.6	336.82	24438.75
SPFA-SLF	1.2	3.96	26.5	320.32	1817.95
SPFA-LLL	1.6	2.63	26.7	317.64	3669.4
SPFA-DEsopo-Pape	1.1	2.42	32.5	421.28	33791.1
SPFA-SLF-and-LLL	2	2.42	25.3	321.44	1966.7
SPFA-LLL-and-Pape	2.1	1.87	27.4	375.28	3597.3
Dijkstra	2.4	3.63	27.8	333.9	332.45
Bellman-Ford	0.1	12.3	798.7	215304.04	153764.45

從測試結果我們可以看到，單純的 SPFA 在各項測資料（第一項以及最後一項除外），皆比 Bellman-Ford 快上不少，並且與 Dijkstra 速度相當。並且在各項優化中，SLF 以及 LLL 的優化效果不錯，但比起 Dijkstra 的穩定度來說仍不足。