Algoritmos da família Ford–Fulkerson – Exemplo de um plano de teste

Os objetivos experimentais são:

- Verificar a complexidade dos algoritmos experimentalmente.
- Comparar os algoritmos.

Como sempre, isso deve ser entendido como: i) a implementação respeita os limites teóricos, e ii) uma analise da precisão da complexidade pessimista, i.e. o quanto a complexidade empírica observada fica abaixa do caso pessimista.

1 Complexidade por contar operações

Nos algoritmos da família Ford-Fulkerson temos dois níveis: as iterações principais do algoritmo e a busca de um caminho s–t.

1.1 Nível do algoritmo

No nível do algoritmos nos temos um número máximo de iterações $\bar{I}$: para Ford-Fulkerson pleno com pesos inteiros é C, para algum limite superior C do fluxo máximo, para Edmonds-Karp é $nm/2$, para "caminho mais gordo" é $m \log C$. Logo uma validação pode determinar $r = I/\bar{I}$, para um número de iterações I, e relatar valores médios $\bar{r}$ sobre um conjunto de instâncias teste.

1.1.1 Detalhes Edmonds-Karp (opcional)

A complexidade é determinada por dois fatores: temos m arcos e cada arco pode ficar crítico no máximo $n/2$ vezes. Logo, uma análise mais detalhada pode contar o número de vezes C_a vezes cada arco estava crítico. (É suficiente manter valores 0 implícitos.) Com isso podemos quantificar o defeito nos dois fatores. Temos

$$C = \sum_{a \in A} [C_a > 0]/m$$

para a fração de arcos que ficou pelo menos uma vez crítico, para cada arco $a \in A$

$$r_a = C_a/(n/2)$$

a fração de criticalidade de a, e logo

$$\bar{r} = \sum_{a \in A} r_a/(Cm)$$

a criticalidade média dos arcos que ficaram crítica pelo menos uma vez. Corretude requer $C \leq 1$ e $r_a \leq 1$, para todos $a \in A$.
Os valores C e $\bar{r}$ podem ser plotados, por exemplo, para diversos grafos teste com n e m diferente.

1.2 Nível de busca do caminho s–t

O Ford-Fulkerson simples e o algoritmo de Edmonds-Karp usam buscas simples, e.g. em profundidade ou em largura no caso do Edmonds-Karp, ambas de complexidade $O(n + m)$. O valor m aqui inclui eventuais arcos para atrás. Logo podemos contar o número n'_i e m'_i de vértices e arcos visitados na iteração i e analisar frações $s_i = n'_i/n$ e $t_i = m'_i/m$. Corretude requer $s_i, t_i \leq 1$. É interessante observar s_i e t_i ao longo das iterações e também o defeito total

$$\bar{s} = \sum_i s_i/I; \qquad \bar{t} = \sum_i t_i/I$$

sobre todas I iterações para diversos grafos teste com n e m diferente. Nota também que no caso do algoritmo de Edmonds-Karp $I \leq (r/2)nm$ mas pode ser menor, caso tem múltiplos arcos críticos por iteração.
Para o "caminho mais gordo" a complexidade é $O(n \log n + m)$ (via heap Fibonacci, por exemplo) ou $O((n+m) \log n)$ (para um heap k-ário, por exemplo). Neste caso podemos, similarmente, contar o número de operações `insert`, `deletemin` e `update`, para quantificar a fração de operações necessárias em cada busca, e relatar frações médias, igual ao primeiro trabalho.

1.3 Complexidade por contar operações

We introduce the following quantities:

- the shortest path length l and its average $\bar{l}$ over all iterations.

We have a finer model of the complexity: it should (maybe with some adequate constants) be

$$I(\bar{s}n + \bar{t}m + \bar{l}).$$

2 Complexidade por medir o tempo

Finalmente podemos medir o tempo real $T(n, m)$ e comparar com a complexidade esperada, e comparar os algoritmos. Temos duas formas. A comparação com a complexidade pessimista

$$\frac{T(n, m)}{nm(n + m)}$$

ou com a complexidade por contar operações

$$\frac{T(n,m)}{\bar{r}/2nm \cdot (\bar{s}n + \bar{t}m)}$$

ou ainda

$$\frac{T(n,m)}{I(\bar{s}n + \bar{t}m)}$$

para ver se existem complexidades residuais não explicadas.
Metrics in the literature. Friis e Olesen (2014) use plain time $T(n)$, for FF-type algorithms normalized time $(T/mI)(m)$ and $(TnI)(n)$ per iteration and arcs or vertices, for PR-type algorithms $(T/nm)(n)$. There are little justifications; for $(T/mI)(m)$ they note "Since this chart is mostly flat, mI is a good approximation of the running time".

3 Instâncias de teste

Além do gerador "washington" temos as instâncias do DIMACS. Nos trabalhos de Boykov & Kolmogorov e Baumstark tem exemplos de outras classes instâncias.
Friis.Olesen/2014 is useful. There Dinic and even more PR wins.

Referências

[1] Jakob Mark Friis e Steen Beier Olesen. "An experimental comparison of maxflow algorithms". Tese de mestrado. Aarhus University, 23 de fev. de 2014. URL: https://cs.au.dk/~gerth/advising/thesis/jakob_mark_friis_steffen_beier_olesen.pdf.