CS4234 Optimisation Algorithms

Notes

AY2024/25 Semester 1 • Prepared by Tian Xiao *@snoidetx*

Problems	Deterministic	Randomized	LP + Rounding
Vertex Cover	Two special cases: ① Vertex cover on a tree; ② Known upper bound k. Deterministic Vertex Cover (2-approximation) Repeat until no remaining edge: ① Pick a random edge (u,v); ② Add both u and v to vertex cover; $G \longrightarrow G-u-v$.	Randomized Vertex Cover (2-approximation) Repeat until no remaining edge: ① Pick a random edge (u,v); ② let $z=u$ or v w.p. $\frac{1}{2}$; ③ Add z to vertex cover; $G \rightarrow G-z$.	For weighted vertex cover: $\min \sum_{v\in V} w_v x_v$ s.t. $x_u+x_v \geq 1,\ \forall (u,v)\in E$ $x_v \in \{0,1\},\ \forall v \in V. \xrightarrow{relax} [0,1]$ Rounding: If $x_v \geq \frac{1}{2}$, add v to vertex cover. (2-approximation)
Set Cover	Greedy Set Cover ($O(\log n)$-approximation) Repeat until all elements are covered: ① Choose the set S_j that covers the most uncovered elements; ② $X \longrightarrow X \setminus S_j$.	Linear Programming $\max c^T x$ — n variables s.t. $Ax \geq b$ — m constraints. $x \geq 0$ Simplex Method ($O(m^n)$).  ① Find any feasible vertex v. ② $v_1, \cdots, v_k \leftarrow$ neighbors of v. ③ Calculate $f(v)$ and $f(v_1) \cdots f(v_k)$. If $f(v)$ is the max, stop and return $f(v)$. ④ Otherwise, choose one neighbor v_j s.t. $f(v_j) > f(v)$. Set $v = v_j$. ⑤ Return to step ②. Farka's Lemma / LP Duality If $Ax \geq b$ has no solution, then $\exists \lambda \geq 0 \in \mathbb{R}^m$ s.t. $\lambda^T A x = 0$ and $\lambda^T b = 1$.	Feasibility is in NP ∩ co-NP ① No solution ⇒ ∃ polynomial λ ② A solution exists ⇒ ∃ polynomial solution Ellipsoid Method (polynomial time) $b-\varepsilon \leq Ax \leq b+\varepsilon$, where $\varepsilon \in \frac{1}{2^{poly(K)}}$ LP Duality → if both finite optimum exists $\max c^T x$ = $\min b^T y$ s.t. $A^T x \leq b;\ x \geq 0.$ ⇒ s.t. $A^T y \geq c;\ y \geq 0.$ Maximum Bipartite Matching $\left.\begin{array}{l}\max \sum_{(u,v)\in E} x_{uv} w_{uv} \\ \text{s.t. } \sum_{u:(u,v)\in E} x_{uv} = 1,\ \forall v \in V \\ 0 \leq x_{uv} \leq 1,\ \forall (u,v) \in E.\end{array}\right\}$ LP solution = ILP solution
Traveling Salesman Metric General Repeat ✓ ✓ Non-Repeat ✓	MST + DFS (for G-R, 2-approximation) ① Construct the complete graph G ② Let T be MST of G. ③ Let C be the cycle by DFS of T. Christofides Algorithm (for M, 1.5-approximation) ① $T \leftarrow$ MST of G. Add T's edges to E. ② Let O be nodes in T with odd degree. $\lvert O \rvert$ is even. ③ $M \leftarrow$ min cost perfect matching for O. ④ $G \leftarrow (X, E \cup M)$ (multigraph). ⑤ Return Eulerian cycle C for G.		Semidefinite Programming relaxation Random hyperplane rounding: 0.87856-approximation
CNF-SAT	Greedy CNF-SAT ($(1-\frac{1}{2^k})$-approximation for k-CNF-SAT; $\frac{\sqrt{5}-1}{2}$-approximation for general CNF-SAT) For each x_i, choose $\underset{x_i \in \{0,1\}}{\arg\max}\ \mathbb{E}[W^* \mid x_1, \cdots, x_i]$.	Randomized Weighted k-CNF-SAT ($(1-\frac{1}{2^k})$-approximation) For each x_i, let $x_i = 1$ or 0 w.p. $\frac{1}{2}$. Randomized Weighted CNF-SAT ($\frac{\sqrt{5}-1}{2}$ approximation) For each x_i, let $x_i = 1$ w.p. $p = \frac{\sqrt{5}-1}{2}$.	$\max \sum_{j=1}^{m} w_j y_j$ s.t. $\sum_{+ve\ e_{ji}} x_i + \sum_{-ve\ e_{ji}} (1-x_i) \geq y_j,\ \forall j=1,\cdots,m$ $x_i \in \{0,1\},\ \forall i=1,\cdots,n$ $y_j \in (0,1\},\ \forall j=1,\cdots,m$ Rounding: $\hat{x}_i = 1$ w.p. x_i^*. ($(1-\frac{1}{e})$-approximation)
CSP-SAT	Lovász Local Lemma Let ϕ be a CSP. If $\exists \mu_c \in (0,1)$ for all $c \in \phi$ s.t. $\forall C \in \phi$, $w(C) \leq \mu_c \prod_{C' \in I_\phi(C)} (1-\mu_{c'})$, then ϕ is satisfiable. ⇒	Moser's Algorithm ($\sum_c \frac{\mu_c}{1-\mu_c}$ iterations) ① Randomly assign values to all variables ② While ∃ unsatisfied constraint C_j: Randomly assign values to all variables in C_j.	
Network Flow	MFMC Theorem Let $G=(V,E)$ with capacity c be a flow network, and fix any $s,t \in v$. There exists a feasible flow f and a cut (S,T) on G s.t. f saturates every edge from S to T and avoids any edge from T to S, with $\lvert f \rvert = \lVert S,T \rVert$. Ford-Fulkerson Algorithm $O(\lvert E \rvert f)$ ① Let $f(u \to v) \leftarrow 0,\ \forall u \to v \in E$ ② While s can reach t in G_f: (a) Use BFS or DFS to find an augmenting path P from s to t in G_f. (b) $F \leftarrow \min_{u \to v \in P} c_f(u \to v)$ (c) For all $u \to v \in P$, update its flow value: i. $f(u \to v) \leftarrow f(u \to v) + F$. ii. $f(v \to u) \leftarrow f(v \to u) - F$. ③ Return f. Application: ① Edge-disjoint paths ② Vertex capacities ③ Bipartite matching ④ Disjoint path cover of acyclic directed graphs ⑤ Minimising the teaching staft.	Flow-Decomposition Theorem Every non-negative (s,t)-flow can be written as a +ve linear combination of directed (s,t) path flows and directed cycle flows. An edge appears in a flow iff the amount of flow through the edge is +ve. #paths + #cycles $\leq \lvert E \rvert$. Edmonds-Karp Fat Pipes Algorithm $O(\lvert E \rvert^2 \log \lvert V \rvert \log \lvert f^* \rvert)$ ① Let $f(u \to v) \leftarrow 0,\ \forall u \to v \in E$ ② While s can reach t in G_f: (a) Use Dijkstra's algorithm to obtain an augmenting path P from s to t in G_f, with the largest bottleneck value F. (b) For all $u \to v \in P$, update its flow value: i. $f(u \to v) \leftarrow f(u \to v) + F$. ii. $f(v \to u) \leftarrow f(v \to u) - F$. ③ Return f.	Dinitz Algorithm $O(\lvert E \rvert^2 \lvert V \rvert)$ ⇒ Use BFS to obtain augmenting path with least no. of vertices