Finding Large Induced Subgraphs

Hemanshu Kaul

Illinois Institute of Technology

www.math.iit.edu/∼kaul

kaul@iit.edu

Joint work with

S. Kapoor (IIT)

Largest Induced Subgraph

Given a graph G, find the largest induced subgraph of G.

Ah!

Largest Induced Subgraph

Given a graph G, find the densest induced subgraph of G.

Goldberg, 1984: Polynomial-time algorithm based on network flows.

Densest k-Subgraph Problem

Given a graph G, find the largest induced subgraph of G on k-vertices.

Densest k-Subgraph Problem

Given a graph G, find the largest induced subgraph of G on k-vertices.

- NP-hard even on Chordal graphs (Corneil, Perl, 1984), on Planar graphs (Keil, Brecht, 1991)
- n/k-approximation algorithm using semi-definite programming (Goemans, 2001)
- $n^{1/3}$-approximation algorithm (Feige, Kortsarz, Peleg, 2001)
- $n^{1/4}$-approximation algorithm (Bhaskar et al., 2010)
- No PTAS in general under a complexity assumption (Khot, 2004)

Approximation Algorithms

Algorithm $\mathcal{A}$ for a maximization problem *MAX* achieves an approximation factor α if

for all inputs G, we have: $\frac{OPT(G)}{\mathcal{A}(G)} \leq \alpha$,
where $\mathcal{A}(G)$ is the value of the output generated by the algorithm $\mathcal{A}$,
and $OPT(G)$ is the optimal value.

A α-approximation algorithm for *MAX* is a polynomial time algorithm that achieves the approximation factor α.

PTAS (Polynomial Time Approximation Scheme) has $\alpha = 1 + \epsilon$.

An Extremal Problem

Given graph on n vertices and m edges, what can be said about the maximum number of edges in an induced subgraph on k vertices within any such graph?.

$f(n,m,k) = \min \{g(G;k) \mid \forall G \text{ on } n \text{ vertices and } m \text{ edges}\}$

where

$g(G;k) = \max\{|E(H)| \mid H \ \ k\text{-vertex induced subgraph of } G\}$

Studied since 1970s,

Chung-Erdős-Spencer (1985) gave a complete description of f when $k = o(n)$, e.g.,

- $f(n,m,k) = \Theta(k^2 m/n^2)$ when $k > (n^2 \log n)/m$
- $f(n,m,k) = \Theta(k \log n / \log(n^2/km))$ when $k < (n^2/m) \log n \log\log n$

Largest Induced Subgraph with Weights

We are interested in a weighted version of the densest k-subgraph problem.

Given a graph G with cost associated with each of its vertices, and benefit associated with each of its edges and vertices.

Find the induced subgraph whose cost does not exceed a given budget while its total benefit is maximized.

Graph Knapsack Problem

Graph Knapsack Problem: Given an instance $GKP(G, b, w, W)$, where $G = (V, E)$ is an undirected graph with n vertices, $w : V \to \mathbb{Z}^+$ is a weight function, $b : E \cup V \to \mathbb{Z}$ is a benefit function on vertices and edges, and W is a weight bound.

$$\begin{aligned} &\text{maximize} \quad b(G[S]) \\ &\text{such that} \\ &\text{weight}(S) \leq W \end{aligned}$$

Graph Knapsack Problem

For a graph G defined on V, the benefit of a subgraph $H = (V_H, E_H)$ is

$$b(H) = \sum_{v \in V_H} b(v) + \sum_{e \in E_H} b(e)$$

while its weight is

$$w(H) = \sum_{v \in V_H} w(v).$$

Graph Knapsack Problem

Given a subset of vertices S, we consider the subgraph induced by S, termed $G[S]$.

The Graph Knapsack Problem (GKP) asks for a subset of vertices, $S \subseteq V$ so as to maximize the benefit of the induced subgraph, $b(G[S])$ with the budget restriction that its weight $w(G[S])$ is less than W.

Graph Knapsack Problem

Relationship to Large Subgraph Problems

GKP is related to the maximum clique problem. We can reduce the clique problem to the graph-knapsack problem.

Given a graph G, suppose we wish to determine if G contains a clique of size t. We define an instance of GKP on G with $W = t$, $w_i = 1$, $b_i = 0$, $b_e = 1$ for $e \in E(G)$.
Graph G has a K_t iff GKP has benefit at least $\binom{t}{2}$.

We may note that, unless $P = NP$, achieving an approximation ratio better than $n^{1-\epsilon}$ is impossible for the clique problem.

Graph Knapsack Problem

Relationship to Large Subgraph Problems

GKP also generalizes the Densest k-Subgraph problem.

This corresponds to GKP with edges of benefit 1 while vertices have zero benefit, and the weight of each vertex is 1 with $W = k$.

Graph Knapsack Problem

Relationship to Large Subgraph Problems

GKP also generalizes the Densest k-Subgraph problem.

This corresponds to GKP with edges of benefit 1 while vertices have zero benefit, and the weight of each vertex is 1 with $W = k$.

- NP-hard even on Chordal graphs (Corneil, Perl, 1984), on Planar graphs (Keil, Brecht, 1991)
- n/k-approximation algorithm using semi-definite programming (Goemans, 2001)
- $n^{1/3}$-approximation algorithm (Feige, Kortsarz, Peleg, 2001)
- $n^{1/4}$-approximation algorithm (Bhaskar et al., 2010)
- No PTAS in general under a complexity assumption (Khot, 2004)

GKP: Resource Allocation under Dependencies

Selection of Projects: We want to pick a collection of projects such that their combined "benefit" is maximized while the total cost does not exceed the given budget.

Previous Research: Choose projects such that the chosen projects have largest sum of individual benefits while their total cost does not exceed the total budget, W. This is simply the classical 0-1 Knapsack problem.

$$\max \sum_{i=1}^{n} B(i)x_i$$

subject to

$$\sum_{i} w_i x_i \leq W$$

$$x_i \in \{0, 1\}$$

GKP: Resource Allocation under Dependencies

Criticism: We have to choose multiple projects for implementation simultaneously, which means that such projects cannot be considered independent of each other. It may happen that two projects which are individually beneficial, will together negate or supplement either of their benefits.

The overall benefits of a collection of projects may be greater than, equal to, or smaller than the sum of individual benefits.

Think of highway projects on a transportation network.

GKP: Resource Allocation under Dependencies

If we have the computing resources to calculate the benefit of each possible collection of projects, we could simply pick the collection with largest value.
However this is not computationally feasible even for small values of n since there are a total of as many as 2^n different collections.

So we are limited to computation of benefits of collections of up to r projects at a time, where r is small fixed integer.
Note this only requires calculation of benefits of up to $n + n^2 + \ldots + n^r$ different collections of projects.

GKP: Resource Allocation under Dependencies

We would like to calculate $B([t]) = B(\{1, 2, \ldots, t\})$ explicitly but that may not be possible/ allowed because $t > r$.
In that case we estimate its value using the computed values of $B(I)$ where $|I| \leq r$ as follows:

GKP: Resource Allocation under Dependencies

$$B(\{1,\ldots,t\}) = \sum_{i=1}^{n} B(\{i\}) + \sum_{I\subseteq[t]\,:\,|I|\leq 2} \Delta_I + \sum_{I\subseteq[t]\,:\,|I|\leq 3} \Delta_I + \ldots + \Delta_{[t]},$$

where Δ values give an Inclusion-Exclusion-formula type description of the difference between the combined benefit of the projects and the sum of the lower order benefits,

$$\Delta_{\{i,j\}} = B(\{i,j\}) - (B(\{i\}) + B(\{j\})),$$

$$\Delta_{\{i,j,k\}} = B(\{i,j,k\}) - (B(\{i,j\}) + B(\{i,k\}) + B(\{j,k\})) + (B(\{i\}) + B(\{j\}) + B(\{k\})),$$

and so on.

GKP: Resource Allocation under Dependencies

When $t > r$, we estimate the $B(\{1, \ldots, t\})$ by using only the first r terms in this formula.

Thus we use the information about the dependency between up to r projects at a time to give a more realistic value of the benefit of a larger collection of projects.

Lets explicitly illustrate the situation when $r = 2$.

GKP: Resource Allocation under Dependencies

Given a set of items $V = \{v_1, ..., v_n\}$ (projects) and a knapsack of limited capacity W (the budget).

To each item we associate a benefit $b(v_i)$ (benefit of that project) and a positive weight w_j (cost of that project).

To each pair ($r = 2$) of items we associate a benefit $b(v_i v_j)$. $b(e) = b(uv) = B(u, v) - (B(u) + B(v))$, difference between the benefit of the two corresponding projects together and the sum of individual project benefits.

The benefits on the edges could be positive or negative. $b(e) = 0$ corresponds to no interdependency between projects.

GKP: Resource Allocation under Dependencies

We can formulate the problem as a 0-1 Quadratic Program:

$$\begin{aligned} \textit{maximize} \quad & \sum_i b(v_i)x_i + \sum_{v_iv_j \in E(G)} b(v_iv_j)x_ix_j \\ \text{such that} \quad & \textstyle\sum_i w(v_i)x_i \leq W \\ & x_i \in \{0,1\} \end{aligned}$$

Replacing the term x_ix_j by an integer variable $x_{ij} \in \{0,1\}$ and adding the constraints $x_{ij} \leq \frac{x_i+x_j}{2}$ and $x_{ij} \geq \frac{x_i+x_j-1}{2}$ provides an integer linear program (ILP) for the problem.

Hypergraph Knapsack Problem

When $r > 2$, the underlying structure considers r-wise dependencies, that is it forms a r-uniform hypergraph.

The definitions given above generalize in a straightforward manner to the Hypergraph Knapsack Problem (HKP).

Let $H = (V, E)$ be a hypergraph.
For any subset S of vertices in H, let $w(S) = \sum_{v \in S} w(v)$ and let $b(S) = \sum_{v \in S} b(v) + \sum_{e \in E : e \subseteq S} b(e)$.

As before HKP asks for a subset of vertices S that maximizes the benefit with the restriction that its weight is less than W.

Computational Study

[Z. Li, S. Kapoor, H. Kaul, and E. Veliou, B. Zhou, S. Lee, 2012], Journal of the Transportation Research Board of the National Academies.

Ongoing traffic improvement project in the financial district portion of the Chicago Central Business District (CBD), the Chicago Loop Area bounded by East Wacker Drive, West Wacker Drive, North Wacker Drive, South Wacker Drive, West Roosevelt Road, East Roosevelt Road and South Lakeshore Drive.

Computational Study

There were a total of 6 projects under consideration.

TABLE Major Investment Projects Proposed for Chicago Loop Area during 2011-2015

Project	Name	Scope	Cost
1	Lower Wacker Drive	Congress Parkway to Randolph Street	$60M
2	Upper Wacker Drive	Congress Parkway to Randolph Street	$80M
3	Interchange	Congress Parkway and Chicago River	$60M
4	Congress Parkway Modernization	Wells Street to Michigan Avenue	$15M
5	Michigan Avenue Resurfacing	Congress Parkway to Roosevelt Road	$3M
6	Lake Shore Drive Resurfacing	Randolph Street to Roosevelt Road	$6M
Total			$224M

Considering data availability of candidate projects proposed for possible implementation in the study area, the analysis period for the computational study was set from 2011-2015.

Computational Study

TABLE Benefits, Costs, Benefit-to-Cost ratios, and Best Sub-Collection of Projects

Budget Level ($224M)	Benefits	Costs	Benefit-to-Cost Ratio	Best Sub-Collection of Projects
10%	6,357,902	1,642,568	3.87	4+5+6
20%	6,357,902	1,642,568	3.87	4+5+6
30%	6,357,902	1,642,568	3.87	4+5+6
40%	12,283,083	4,744,517	2.59	156+4
50%	12,283,083	4,744,517	2.59	156+4
60%	12,374,624	5,778,500	2.14	2+456
70%	18,185,954	7,846,466	2.32	156+3+4
80%	18,522,072	8,880,449	2.09	124+5+6
90%	18,522,072	8,880,449	2.09	124+5+6
100%	18,548,533	11,982,398	1.55	14+23+56

Computational Study

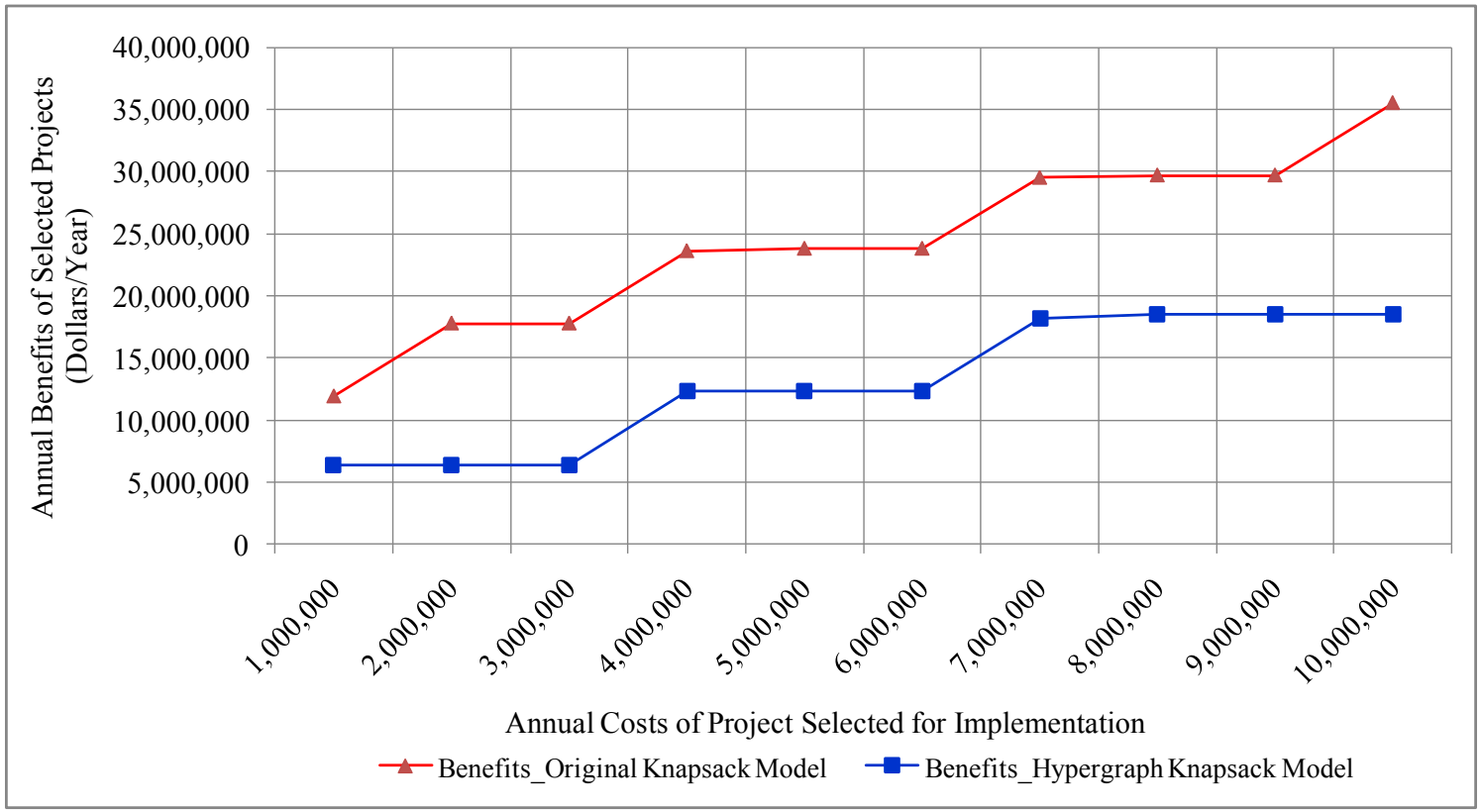

FIGURE Comparison of total benefits of project selection with and without project interdependency considerations.

Computational Study

Main observations:

- The network-wide benefits with project interdependency considerations tend to be lower than the corresponding benefits without interdependency considerations by 38-64 percent.
- The network-wide benefits with project interdependency considerations begin to flatten out when the annualized budget reach approximately $7.5M. No additional benefits are generated with higher levels of investment budgets.

Quadratic Knapsack Problem

The Quadratic Knapsack Problem (QKP) is the appropriate problem for comparison with GKP. They are essentially the same problem when benefits are non-negative.

$$\begin{aligned} \textit{maximize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} x_i x_j \\ \text{such that} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ & x_i \in \{0, 1\} \end{aligned}$$

Quadratic Knapsack Problem

No approximation algorithms or FPTAS are known for the general QKP. The focus has been on IP-based exact methods.

Rader and Woeginger (2002) developed a FPTAS for the case when all benefits are non-negative and the underlying graph is a series-parallel graph.
They also show that when QKP has both negative and non-negative benefits, it can not have any fixed approximation unless $P = NP$.

Note that the Hypergraph Knapsack Problem (HKP) can **not** be reduced to the QKP or some version of it.

Generalized Knapsack Problems

Relationship to other Knapsack Problems

The idea of using discrete structures like graphs, digraphs, posets to generalize the classical knapsack problem by modeling some sort of dependency among the items is not a new one.

However all such generalizations of the Knapsack problem restrict the choice of subset of items that can be picked. While our model does not restrict the choices directly, instead it modifies the benefit function so that the benefit on the edge between a pair of items could act as a penalty (if its negative) or an inducement (if its positive) towards the choice of those two items.

Generalized Knapsack Problems

Relationship to other Knapsack Problems

The Knapsack Problem with Conflict Graph is a knapsack problem where each edge in the underlying conflict graph on the items introduces the constraint that at most one of those two items can be chosen.
This can be modeled as the Graphical Knapsack problem by putting large negative benefit on the edges of the conflict graph and using that as the underlying graph for GKP.

Generalized Knapsack Problems

Relationship to other Knapsack Problems

The Constrained Knapsack Problem in which dependencies between items are given by a graph.
In the first version, an item can be selected only if at least one of its neighbors is also selected.
In the second version, an item can be selected only when all its neighbors are also selected.
These can also be modeled as GKP.

Also, Precedence-Constrained Knapsack Problem, Subset-Union Knapsack, etc.

Greedy Algorithm

Fix an integer t. The greedy algorithm can be defined naturally as:

1. Initialize $S = \emptyset$
2. Pick a subset T of $V(G) - S$ of cardinality at most t such that its benefit (the sum of the benefits of the vertices and edges induced by T in $S \cup T$) to weight ratio is highest
3. Update $S = S \cup T$ if weight of $S \cup T$ satisfies the budget constraint, and then go to step 2. Otherwise pick whichever of S or T has larger benefit as the final solution.

When $t = 1$, the worst case benefit ratio can be made arbitrarily bad.

Greedy Algorithm

The difficulty in analyzing this greedy algorithm:

- Handling two kinds of “weights”
- Each step depends on partial solution from previous steps in an involved manner due to edges that go across.

Main idea:

- An arbitrary instance of GKP with greedy solution A and optimal solution O defines a new instance of GKP which has disjoint greedy and optimal solutions with its greedy solution same as A and benefit of its optimal solution no worse than $b(O)$.
- Apply averaging arguments on this new instance, and use the disjointness of the two solutions and their relation to original instance to get the bound on the ratio of original benefits.

Greedy Algorithm

S. Kapoor, H. Kaul, 2013+

For any fixed $t \geq 0$, the greedy algorithm is a $(8\min(n, W)/t)$-factor polynomial time ($O(2^{t+1}\binom{n+1}{t+1})$-running time) approximation algorithm for $GKP(G, b, w, W)$ with n vertices, when b is a non-negative function.

This analysis is sharp.
We can construct a family of instances of GKP where ratio of the optimal solution to the greedy solution is $\Omega(\frac{n}{t})$.

No such results are known for Quadratic Knapsack Problem.

Greedy Algorithm

S. Kapoor, H. Kaul, 2013+

For any fixed $t \geq 0$, the greedy algorithm is a $(8 \min(n, W)\, W/t)$-factor polynomial time ($O(2^{t+1}\binom{n+1}{t+1})$-running time) approximation algorithm for $GKP(G, b, w, W)$ with n vertices, when b can take both negative and non-negative values.

This analysis is sharp.
We can construct a family of instances of GKP where ratio of the optimal solution to the greedy solution is $\Omega(\frac{n^2}{t})$ where $W = \Theta(n)$.

Again, no such results are known for Quadratic Knapsack Problem.

Greedy Algorithm

Why the extra factor of W when negative benefits are possible?

When benefits are non-negative, we can show that $w(v) \leq W/2$ for all v which implies that $W/w(A) \leq 2$.

When benefits are negative, this ratio can be as bad as essentially W.

Greedy Algorithm for Hypergraph Knapsack

The definition of the greedy algorithm works for Hypergraph Knapsack problem as well.

However, taking $t < r$ (where r is the largest size of an edge in the underlying hypergraph) can make the worst case benefit ratio arbitrarily bad.

Greedy Algorithm for Hypergraph Knapsack

S. Kapoor, H. Kaul, 2013+

For any fixed $t \geq r$, the greedy algorithm is a $\left(8\left(\frac{\min(n,W)}{t-r+1}\right)^{r-1}\right)$-factor polynomial time ($O(2^t\binom{n}{t})$-running time) approximation algorithm for $HKP(H, b, w, W)$ with n vertices and r-uniform edges, when b is a non-negative function.

This analysis is essentially sharp.
We can construct a family of instances of HKP where ratio of the optimal solution to the greedy solution is $\Omega(\frac{(n-r+1)^{r-1}}{t^{r-1}})$.

Greedy Algorithm for Hypergraph Knapsack

S. Kapoor, H. Kaul, 2013+

The greedy algorithm is a $\left(2W\left(\frac{\min(n,W)}{t-r+1}\right)^{r-1}\right)$-factor polynomial time ($O(2^t\binom{n}{t})$-running time) approximation algorithm for $HKP(H,b,w,W)$ with n vertices and r-uniform edges, when b can take both negative and non-negative values.

This analysis is essentially sharp.
We can construct a family of instances of HKP where ratio of the optimal solution to the greedy solution is $\Omega(\frac{(n-r+1)^{r-1}n}{t^{r-1}})$.

FPTAS for bounded tree-width graphs

S. Kapoor, H. Kaul, 2013+

Let G be a graph with tree-width at most k. Then $GKP(G, b, w, W)$ can be approximated to within a factor of $(1+\epsilon)$ in time $O(\frac{2^k n^9 \log n}{\epsilon^2})$.

This result extends to HKP with hypergraph of bounded tree-width.
Both based on a pseudo-polynomial dynamic programming algorithm with lots of book-keeping.

Previous result:
[Rader and Woeginger, 2002] FPTAS for QKP when the underlying graph is series-parallel, which is a family of graphs with tree-width 2.

Randomized Approximation Algorithm for GKP

S. Kapoor, H. Kaul, 2013+

GKP (with b non-negative) can be approximated within the factor $O(w_{max} \frac{n}{\beta(n)} \log^2 n)$ using a randomized polynomial time algorithm, where $W \geq \beta(n)$, and w_{max} is the maximum weight of a vertex.

Note that this algorithm works well when W is large (so $\beta(n)$ can be made large).
Recall the greedy algorithm, it works well for small values of W.

S. Kapoor, H. Kaul, 2013+

For any fixed $t \geq 0$, the greedy algorithm is a $(8 \min(n, W)/t)$-factor polynomial time approximation algorithm for *GKP*.

So, why not combine these two algorithms?

Randomized Approximation Algorithm for GKP

A combination of these two algorithms gives:

S. Kapoor, H. Kaul, 2013+

GKP can be approximated to within a factor of $O(w_{max}\ \sqrt{n}\ \ log^{2+\gamma} n)$ in polynomial time, where γ is an arbitrary small positive constant.

Randomized Approximation Algorithm for GKP

In fact all these results generalize to a bigger class of optimization problems that we call
PIQP (Positive Integer Quadratic Program):
For $X \in \mathbb{Z}^n, X = (x_1, \ldots x_n)$:

$$\begin{aligned} \max\ & X^T B X & + & \quad b^T X \\ \text{subject to} & & & \\ & a_i^T X & \leq & \quad W_i, \ \ i = 1, \ldots p \\ & x_i & \in & \quad \{0, 1\} \end{aligned}$$

with non-negative integer vectors a_i and b, and the $n \times n$ non-negative-integer matrix B.

For our results to hold, number of constraints can be as large as $p \leq O(\lg n)$.
The approximation factor increases by a factor of p.

Tools for Analysis

Main Tools for the Randomized Algorithm:

- Greedy Algorithm
- Multidimensional Knapsack Problem
- Hyperbolic relaxation of GKP; a Second Order Cone Program (SOCP)
- Chernoff-Hoeffding tail bounds
- Kim-Vu polynomial concentration

Tools for Analysis

We generate 4 solutions and take the best of them:

Solution 1: The Greedy solution.

Solution 2: An edge with maximum benefit.

Solution 3:

Solution 4:

Both these solutions use a SOCP, hyperbolic Program.

Tools for Analysis

Solve the relaxation of the hyperbolic program to get an optimal solution x_u^*

$$\begin{aligned} \textit{maximize} \quad & \sum_{uv \in E(G)} b(uv) x_{uv} \\ \text{such that} \quad & \textstyle\sum_u w(u) x_u \leq W \\ & x_u x_v \geq x_{uv}^2 \\ & 0 \leq x_u \leq 1 \end{aligned}$$

Generate a random 0-1 solution Y:
$Y_u = 1$ with probability $\sqrt{x_u^*}/\lambda$

Tools for Analysis

Choose a scaling factor λ so that $\mathbf{E}[w(Y)] \leq \lambda W$

This needs

Lemma

Let $W \geq \beta(n)$. Then for each i,
$\sum_u a_{i,u}\sqrt{x_u^*} \leq 2W\sqrt{a_{max}n/\beta(n)}$, where $a_{max} = \max\{a_{ij}\}$.

This gives $\lambda = 2\sqrt{a_{max}n/\beta(n)}$.

Tools for Analysis

The fourth solution is Z:

First find $v \in V(G)$ such that
$v = \text{argmax}_w \sum_{u \in N_G(w)} b_{uw} \sqrt{x_u^*}/\lambda$.

For this fixed v, define $Z_v = 1$, $Z_w = 0$ at every vertex, $w \notin N_G(v) \cup \{v\}$

Z_u for $u \in N_G(v)$ is determined by solving a "local" 0-1 Knapsack problem with multiple constraints, whose items are the neighbors of v and benefit of each such item equals the benefit of the edge incident to it and v.

Analysis of Solutions

For $0 < \alpha < 1$, with P denoting the 0-1 hyperbolic program for solving QKP, and P^* its relaxation, we have that

$$\begin{aligned} & \mathbf{P}[F(Y) < (1-\alpha)OPT(P)/\lambda^2] \\ \leq\ & \mathbf{P}[F(Y) < (1-\alpha)OPT(P^*)/\lambda^2] \\ =\ & \mathbf{P}[F(Y) < (1-\alpha)\mathbf{E}[F(Y)]] \\ =\ & \mathbf{P}[\mathbf{E}[F(Y)] - F(Y) > \alpha\mathbf{E}[F(Y)]] \end{aligned}$$

Analysis of Solutions

For $0 < \alpha < 1$, with P denoting the 0-1 hyperbolic program for solving QKP, and P^* its relaxation, we have that

$$\begin{aligned} & \mathbf{P}[F(Y) < (1-\alpha)OPT(P)/\lambda^2] \\ \leq\ & \mathbf{P}[F(Y) < (1-\alpha)OPT(P^*)/\lambda^2] \\ =\ & \mathbf{P}[F(Y) < (1-\alpha)\mathbf{E}[F(Y)]] \\ =\ & \mathbf{P}[\mathbf{E}[F(Y)] - F(Y) > \alpha\mathbf{E}[F(Y)]] \end{aligned}$$

If we can show that this probability is small, then that would prove that $F(Y)$, the value of the solution Y is within a factor $\lambda^2/(1-\alpha)$ of the optimal (as long as the budget constraints are satisfied).

Analysis of Solutions

Define

$\varepsilon_0 = \mathbf{E}[b(Y)]$, i.e., $OPT(HP^*)/\lambda$, a measure of global solution.

$\varepsilon_1 = \max_v(\sum_{u \in N(v)} \mathbf{P}[Y_u = 1])$, a measure of dense local neighborhood solution.

$\varepsilon_2 = \max_{uv \in E(G)} b(uv)$, a measure of most beneficial edge.

J-H. Kim, Van Vu, 2001

$$\mathbf{P}[\mathbf{E}[Y] - Y > t^2] < 2e^2 \ exp\left(-t/32(2\varepsilon\varepsilon')^{1/4} + \log n\right)$$

where $\varepsilon = \max\{\varepsilon_0, \varepsilon_1, \varepsilon_2\}$, and $\varepsilon' = \max\{\varepsilon_1, \varepsilon_2\}$.

This applies to any non-linear random variable of the form: $Y = \sum_{e \in E(G)} b_e \prod_{v \in e} y_v$ where G is an underlying graph, b is non-negative weight function on the edges, and y_v are independent $0-1$ random variables.

Analysis of Solutions

- When $\varepsilon_2 > \varepsilon_1$ and $\varepsilon_0 < \varepsilon_2 \log^4 n$, $\max b_{uv}$ works as a good solution
- When $\varepsilon_2 > \varepsilon_1$ and $\varepsilon_0 > \varepsilon_2 \log^4 n$, Kim-Vu applies to Y, the randomized solution
- When $\varepsilon_1 > \varepsilon_2$ and $\varepsilon_0 > \varepsilon_1 \log^4 n$, Kim-Vu applies to Y, the randomized solution
- When $\varepsilon_1 > \varepsilon_2$ and $\varepsilon_0 < \varepsilon_1 \log^4 n$, Z, solution 4 is used, which is shown to be concentrated via Chernoff-Hoeffding.

Analysis of Solutions

Finally, the solution is shown to exceed the budget W with probability $o(n)$ using Chernoff-Hoeffding, so that

If we repeat the randomized algorithm $k = O(n^c)$ times this probability becomes $e^{-(\delta^2 k/6\sqrt{n})}$.

Analysis of Solutions

Finally, the solution is shown to exceed the budget W with probability $o(n)$ using Chernoff-Hoeffding, so that

If we repeat the randomized algorithm $k = O(n^c)$ times this probability becomes $e^{-(\delta^2 k/6\sqrt{n})}$.

S. Kapoor, H. Kaul, 2013+

PIQP, with $\min_i W_i \geq \beta(n)$ and $p \leq O(\lg n)$, can be approximated within the factor $O(w_{max}\ p\ \frac{n}{\beta(n)} \log^2 n)$ using a randomized polynomial-time algorithm, where a_{max} is the maximum entry in the constraint matrix A, .