Resource Allocation under Dependencies with application in transportation networks

Hemanshu Kaul

Illinois Institute of Technology

www.math.iit.edu/∼kaul

kaul@math.iit.edu

Transportation Network

Transportation Network: A network of roads and highways (*edges or links*) connecting locations and intersections (*vertices or nodes*) under study.

Problem: Given a total budget, how to choose a collection of transportation projects to implement on the given network so as to improve its overall performance?

Transportation Network

Transportation Network: A network of roads and highways (*edges or links*) connecting locations and intersections (*vertices or nodes*) under study.

Problem: Given a total budget, how to choose a collection of transportation projects to implement on the given network so as to improve its overall performance?

Improving a Transportation Network

Characteristics of a transportation network:

- the overall structure of the network - the nodes and the links
- capacity, maximum speed (and other characteristics) of each link

What is a project?

- Remove a link (e.g. change a 2-way street into 1-way)
- Add a new link (new road/highway)
- Change an existing link by modifying its characteristics

Each project has a cost associated with it.

Improving a Transportation Network

Characteristics of a transportation network:

- the overall structure of the network - the nodes and the links
- capacity, maximum speed (and other characteristics) of each link

What is a project?

- Remove a link (e.g. change a 2-way street into 1-way)
- Add a new link (new road/highway)
- Change an existing link by modifying its characteristics

Each project has a cost associated with it.

Improving a Transportation Network

Characteristics of a transportation network:

- the overall structure of the network - the nodes and the links
- capacity, maximum speed (and other characteristics) of each link

What is a project?

- Remove a link (e.g. change a 2-way street into 1-way)
- Add a new link (new road/highway)
- Change an existing link by modifying its characteristics

Each project has a cost associated with it.

Benefit of a Transportation Project

Benefit of a Transportation Project measure benefits gained by the transportation network as viewed from economic (the transportation agency and the user costs), social (traffic mobility and safety), and environmental (vehicle emissions) dimensions.

Typically computed as net reductions in cost concerning

- preservation, expansion, and maintenance of physical facilities (such as pavement, preservation, expansion, and travel safety hardware),
- vehicle operation, travel time, crashes,
- and, vehicle emissions

during the service life-cycle of the facility after project implementation.

Benefit of a Transportation Project

Previous Research: Measure the impact of a project on the growth or reduction in traffic volume on the network links within the project physical range.

The cost computations are based on this local change in traffic volume.

Benefit of a Transportation Project

Previous Research: Measure the impact of a project on the growth or reduction in traffic volume on the network links within the project physical range.
The cost computations are based on this local change in traffic volume.

Criticism: Local changes in a transportation network can lead to agglomerative changes in its global behavior.

For instance, expanding the capacity of a single roadway link typically improves traffic operations of the link. However, it may lead to better or worse traffic conditions elsewhere, leading to a much larger or smaller overall network-level gain.

Modeling a Transportation Network

Transportation Network is $N = (G, (S, T), O\text{-}D, u)$, where

- G is a directed graph corresponding to the physical network,
- (S, T) are source and sink pairs, vertices in S correspond to starting points of the traffic (sources like suburbs and other residential areas) and those in T correspond to destinations of the traffic (sinks like downtown and other office locations),
- O-D is the demand matrix, for each $(s, t) \in S \times T$ there is a traffic demand from s to t which is summarized in the O-D matrix
- each edge (transportation link) e has an upper bound, $u(e)$, on its traffic capacity.

We think of each of these (s, t) traffic flows as a distinct commodity flow, i.e., the traffic flow in this transportation network is modeled as a multi-commodity network flow.

Modeling a Transportation Network

Transportation Network is $N = (G, (S, T), O\text{-}D, u)$, where

- G is a directed graph corresponding to the physical network,
- (S, T) are source and sink pairs, vertices in S correspond to starting points of the traffic (sources like suburbs and other residential areas) and those in T correspond to destinations of the traffic (sinks like downtown and other office locations),
- $O\text{-}D$ is the demand matrix, for each $(s, t) \in S \times T$ there is a traffic demand from s to t which is summarized in the $O\text{-}D$ matrix
- each edge (transportation link) e has an upper bound, $u(e)$, on its traffic capacity.

We think of each of these (s, t) traffic flows as a distinct commodity flow, i.e., the traffic flow in this transportation network is modeled as a multi-commodity network flow.

Modeling a Transportation Network

Transportation Network is $N = (G, (S, T), O\text{-}D, u)$, where

- G is a directed graph corresponding to the physical network,
- (S, T) are source and sink pairs, vertices in S correspond to starting points of the traffic (sources like suburbs and other residential areas) and those in T correspond to destinations of the traffic (sinks like downtown and other office locations),
- $O\text{-}D$ is the demand matrix, for each $(s, t) \in S \times T$ there is a traffic demand from s to t which is summarized in the $O\text{-}D$ matrix
- each edge (transportation link) e has an upper bound, $u(e)$, on its traffic capacity.

We think of each of these (s, t) traffic flows as a distinct commodity flow, i.e., the traffic flow in this transportation network is modeled as a multi-commodity network flow.

Modeling a Transportation Network

Transportation Network is $N = (G, (S, T), O\text{-}D, u)$, where

- G is a directed graph corresponding to the physical network,
- (S, T) are source and sink pairs, vertices in S correspond to starting points of the traffic (sources like suburbs and other residential areas) and those in T correspond to destinations of the traffic (sinks like downtown and other office locations),
- $O\text{-}D$ is the demand matrix, for each $(s, t) \in S \times T$ there is a traffic demand from s to t which is summarized in the $O\text{-}D$ matrix
- each edge (transportation link) e has an upper bound, $u(e)$, on its traffic capacity.

We think of each of these (s, t) traffic flows as a distinct commodity flow, i.e., the traffic flow in this transportation network is modeled as a multi-commodity network flow.

Modeling a Transportation Network

Transportation Network is $N = (G, (S, T), O\text{-}D, u)$, where

- G is a directed graph corresponding to the physical network,
- (S, T) are source and sink pairs, vertices in S correspond to starting points of the traffic (sources like suburbs and other residential areas) and those in T correspond to destinations of the traffic (sinks like downtown and other office locations),
- $O\text{-}D$ is the demand matrix, for each $(s, t) \in S \times T$ there is a traffic demand from s to t which is summarized in the $O\text{-}D$ matrix
- each edge (transportation link) e has an upper bound, $u(e)$, on its traffic capacity.

We think of each of these (s, t) traffic flows as a distinct commodity flow, i.e., the traffic flow in this transportation network is modeled as a multi-commodity network flow.

Modeling a Transportation Network

Transportation Network is $N = (G, (S, T), O\text{-}D, u)$, where

- G is a directed graph corresponding to the physical network,
- (S, T) are source and sink pairs, vertices in S correspond to starting points of the traffic (sources like suburbs and other residential areas) and those in T correspond to destinations of the traffic (sinks like downtown and other office locations),
- O-D is the demand matrix, for each $(s, t) \in S \times T$ there is a traffic demand from s to t which is summarized in the O-D matrix
- each edge (transportation link) e has an upper bound, $u(e)$, on its traffic capacity.

We think of each of these (s, t) traffic flows as a distinct commodity flow, i.e., the traffic flow in this transportation network is modeled as a multi-commodity network flow.

Benefit of a Transportation Project

[S. Kapoor, H. Kaul, Z. Li, and M. Pelsmajer]

Given

- a portfolio of n projects labeled by $[n] = \{1, \ldots, n\}$
- a total budget W
- the building cost w_i of each of the n projects

Denote by $N(I)$ the modified network obtained from the original network N by making the modifications corresponding to the collection of projects $I \subseteq [n]$.

Benefit of a Transportation Project

[S. Kapoor, H. Kaul, Z. Li, and M. Pelsmajer]

Given

- a portfolio of n projects labeled by $[n] = \{1, \dots, n\}$
- a total budget W
- the building cost w_i of each of the n projects

Denote by $N(I)$ the modified network obtained from the original network N by making the modifications corresponding to the collection of projects $I \subseteq [n]$.

Define unadjusted benefit of I, $D(I)$, as the minimum value of $C(x)$ subject to the usual multicommodity flow constraints on this network $N(I)$ using the O-D demands.

$C(x)$ is a nonlinear objective function that models the total cost (travel time, ecological cost, vehicle operating cost, travel time, maintenance cost, etc.) of traffic flow in this network.

Benefit of a Transportation Project

$$\min C(\mathbf{x})$$

subject to

$$\sum_k x_k(e) \le u(e)$$ capacity constraint for each edge e

$$\sum_{e:h(e)=u} x_k(e) = \sum_{e:t(e)=u} x_k(e)$$ preservation of flow for each k and each $u \notin S \cup T$

$$\sum_{e:t(e)=u} x_k(e) = OD_k$$ traffic outflow for each k and each $u \in S$

$$\sum_{e:h(e)=u} x_k(e) = OD_k$$ traffic inflow for each k and each $u \in T$

$$x_k(e) \ge 0$$

Note: k indicates an s,t-traffic flow, a commodity flow.

Benefit of a Transportation Project

Define the benefit of a project collection I as $B(I) = D - D(I)$, where D is defined analogously to $D(I)$ for the original network N.

The change in the life-cycle cost of the whole transportation network after implementation of projects in I.

Selection of Projects

Selection of Transportation Projects: We want to pick a collection of projects I such that its benefit $B(I)$ as calculated above is maximum while the total cost does not exceed the given budget.

Selection of Projects

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

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

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

Selection of Projects

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 to the network, will together negate 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.

Selection of Projects with Interdependencies

[S. Kapoor, H. Kaul, Z. Li, and M. Pelsmajer]

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.

Selection of Projects with Interdependencies

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:

Selection of Projects with Interdependencies

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

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.

Selection of Projects with Interdependencies

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 $r = 2$.

Graph Knapsack Problem

[S. Kapoor, H. Kaul, and M. Pelsmajer]

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 $b(v_i v_j)$.
$b(e) = b(uv) = \Delta_{\{u,v\}} = B(u, v) - (B(u) + B(v))$,
difference between the benefit of the two corresponding projects together and the sum of individual project benefits.

Graph Knapsack Problem

So we have graph G with vertices corresponding to projects and edges corresponding to pairs of projects. Both vertices and edges have benefits, while vertices also have cost associated with them.

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

For example:

$$b(K_2) = b(u) + b(v) + b(uv)$$
$$= B(u) + B(v) + (B(u,v) - B(u) - B(v))$$
$$= B(u,v)$$

$$b(K_3) = b(u) + b(v) + b(w) + b(uv) + b(vw) + b(wu)$$
$$= B(u) + B(v) + B(w) + (B(u,v) - B(u) - B(v)) + (B(v,w) - B(v) - B(w)) + (B(w,u) - B(w) - B(u))$$
$$= B(u,v) + B(v,w) + B(w,u) - (B(u) + B(v) + B(w))$$

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

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

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

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], to be published in The Journal of the Transportation Research Board.

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

Travel demand data from Chicago Metropolitan Agency for Planning (CMAP) with information on hourly travel demand for a typical day for approximately 2000 internal and external traffic analysis zones (TAZs) for the entire Chicago Metropolitan Area.

The data for Chicago Loop Area covering 19 TAZs, including 10 internal and 9 external TAZs, was extracted from the city data.

Computational Study

The highway network within the study area is comprised of 486 freeway, expressway, arterial, and collector links (highway segments) and 205 nodes (intersections).

The Google earth photo images were used to accurately create link-node connectivity for through and left/right turning movements.

Details of travel lane widths and speed limits were also obtained to help determine the link capacities and base free flow travel times.

Further, the in-flow nodes and out-flow nodes that act as the sources and sinks of the 100 internal-internal O-D pairs and 81 external-external O-D pairs were identified.

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 Results

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 Results

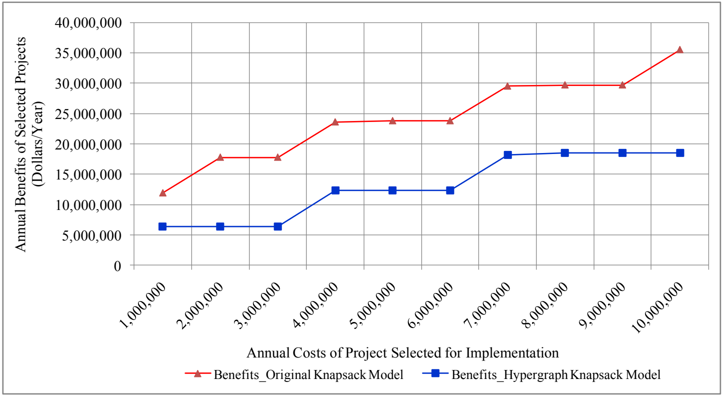

FIGURE Comparison of total benefits of project selection with and without project interdependency considerations.

Computational Results

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.

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.

Graph Knapsack Problem

[S. Kapoor, H. Kaul, and M. Pelsmajer]

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

Relationship to Large Subgraph Problems

From a graph theoretic point of view, it 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 Dense k-Subgraph problem, which requires finding an k-vertex induced subgraph of an edge-weighted graph with maximum density.

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

For the dense k-subgraph problem, an approximation factor of $n^{1/3}$ has been achieved.

Graph Knapsack Problem

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.

Graph Knapsack Problem

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.

Graph Knapsack Problem

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.

Graph 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.

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

Graph Knapsack Problem

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

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

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

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, and M. Pelsmajer, 2011+

The greedy algorithm is a $(16\min(n, W)/t)$-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.

For $t = \log n$ we get a $(\frac{8n}{logn})$-factor (and $(\frac{8W}{\log n})$-factor) linear quasipolynomial-time algorithm.

This analysis is sharp.
We can give an instance 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, and M. Pelsmajer, 2011+

The greedy algorithm is a $(4\min(n, W)W/t)$-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 give an instance 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, and M. Pelsmajer, 2011+

The greedy algorithm is a $\left(16\left(\frac{\min(n,W)}{t-r+1}\right)^{r-1}\right)$-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 give an instance 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, and M. Pelsmajer, 2011+

The greedy algorithm is a $\left(4W\left(\frac{\min(n,W)}{t-r+1}\right)^{r-1}\right)$-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 give an instance 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

Given a graph $G = (V, E)$, a tree decomposition of width k of G consists of a tree T whose nodes correspond to a collection of subsets of $V(G)$, $X_1, X_2, \dots, X_r$ such that

1. $V(G) = \cup X_i$,
2. for each $uv \in E(G)$, there is an i s.t. $u, v \in X_i$,
3. for all $u, v, w \in V(T)$ with v lying on a $u - w$-path in T, $X_u \cap X_w \subset X_v$, and
4. $|X_i| \leq k + 1$.

Many commonly arising families of graphs have bounded tree-width.

FPTAS for bounded tree-width graphs

S. Kapoor, H. Kaul, and M. Pelsmajer

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 at most 2.

Randomized Approximation Algorithm for GKP

S. Kapoor and H. Kaul, 2011+

A polynomial-time randomized algorithm that approximates GKP to the factor $O(n^{1/2}w_{max})$ when b is non-negative.

Main Tools:

- Greedy Algorithm (useful when $W < n^{1/2}$)
- Hyperbolic relaxation of GKP
- Chernoff-Hoeffding tail bounds
- Kim-Vu polynomial concentration

Randomized Approximation Algorithm for GKP

Solve the relaxation of the hyperbolic program (HP^*) to get optimal solution x_u^*

$$\begin{array}{ll} maximize & \sum\limits_{uv \in E(G)} b(uv) x_{uv} \\ \text{such that} & \sum_i w(u) x_u \leq W \\ & x_u x_v \geq x_{uv}^2 \end{array}$$

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

Randomized Approximation Algorithm for GKP

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

Lemma

$\sum_u w(u)\sqrt{x_u^*} \leq 2\sqrt{w_{max}}Wn^{1/4}$, *where* $w_{max} = \max_u w(u)$.

Take $\lambda = 2\sqrt{w_{max}}n^{1/4}$.

Use Chernoff-Hoeffding to show the concentration of the weight around its mean, so the budget can be satisfied w.h.p.

Randomized Approximation Algorithm for GKP

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

Randomized Approximation Algorithm for GKP

- 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 $\varepsilon_0 < \varepsilon_1 \log^4 n$, a different randomized solution based on the dense local neighborhood solution: center of the neighborhood is picked with probability 1, neighbors are picked by solving a classical Knapsack on the neighborhood, and nothing else is picked, which is shown to be concentrated via Chernoff-Hoeffding.

Generalizing the Randomized Approximation Algo

We can extend this approach to solving the following class of quadratic integer programs where $X \in Z^n, X = (X_1, \ldots X_n)$:

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

for bounded number of constraints p and non-negative coefficients in Q and c and in the constraint matrix.

Extremal Problems on Maximum induced subgraphs

Classical results of F.Chung, P. Erdos, and J. Spencer (1985) on maximum number of edges in an induced subgraph on k vertices within a given graph on n vertices and m edges.

Preliminary results (with D. West) on extending these extremal results to graphs with bounded chromatic number (t-partite), a generalization of the Zarankiewicz problem.