Interacting Knapsack Problem in Designing Resource Bundles

Truong-Huy D. Nguyen, Pradeep Varakantham, Hoong-Chuin Lau, Shih-Fen Cheng

Singapore Management University
80 Stamford Road, 178902, Singapore
dthnguyen,pradeepv,hclau,sfcheng@smu.edu.sg

1 Business Data Analytics with Graphical Models

In many real-life businesses, the service provider/seller keeps a log of the visitors' behavior as a way to assess the efficiency of the current business/operation model and find room for improvement. For example, by tracking when visitors entering attractions in a theme park, theme park owners can detect when and where congestion may occur, thus having contingency plans to reroute the visitors accordingly. Similarly, a Cable TV service provider can track channel switching events at each household to identify uninteresting channels. Subsequently, the repertoire of channels up for subscription can evolve over time to better serve the entertainment demand of subscribers. All tracking activities can be done anonymously so as to protect the customers' privacy. Given a large amount of data about customer behavior, we would like to succinctly represent it so that insights can be inferred more quickly and conveniently.

Graphical models [3] is a promising option to serve this purpose. In such a network, the nodes represent the services presented to customers, whereas the value-attached edges represent the customers' usage behavior of these services. Note that the mentioned models are not limited to binary graphs; they can be hypergraphs. For example, in the domain of theme park, as the nodes represent attractions, the strength of a hyperedge that connects multiple nodes indicate how often the involved group of attractions are visited. These models act as query-able knowledge bases that can be further augmented with properties of business interest such as profit or operation cost to aid the sellers' decision making.

In the next section, we zoom specifically into the problem of bundle design in the domain of revenue management; the goal is to bundle resources or services offered to customers in order to maximize revenue [1, 2, 4] In particular, given information on profit and appealingness of items, we introduce a new mathematical formulation to model the practical problem of creating service packages that are, on the one hand, appealing to customers, while on the other hand maximally profitable. In our work, profit and appealingness information is represented graphically from logs of customers' usage behavior.

2 Interacting Knapsack Problem

Similar to the classic 0/1 Knapsack problem (KP), the Interacting Knapsack Problem (IKP) is concerned about selecting a set of items such that their total weight fits within a budget whereas the total reward is maximized. However, the weight and reward functions in IKP are conditioned on the bundle in consideration, reflecting the intra-bundle interactivity effect. Using the same formulation as KP, IKP is characterized as Formulation (1), in which in which $\mathbf{x}$ is an 0-1 vector that indicates the item to be picked, and p_i and w_i are individualized profit and weight functions for item i. Note that the second constraint in Formulation (1) reflects the fact that the total number of items in the bundle is limited to K. Clearly, when p_i and w_i are not conditioned on the selected items, the problem is reduced to classic KP.

$$\left[\begin{array}{ll} \max_{\mathbf{x}} & \sum_i p_i(\mathbf{x})x_i \\ \text{s.t.} & \sum_i w_i(\mathbf{x})x_i \le c, \\ & \sum_i x_i = K, \quad 1 \le K \le n \\ & x_i \in \{0,1\}, \quad i \in N = \{0,...,n\} \end{array}\right. \quad (1) \qquad \left[\begin{array}{ll} \max & \sum_{j=1}^{|\mathbb{Y}_K|} y_j \sum_i p_i(y_j = 1) \\ \text{s.t.} & \sum_{j=1}^{|\mathbb{Y}_K|} y_j \sum_i w_i(y_j = 1) \le c, \\ & \sum_j y_j \le 1, \\ & y_j \in \{0,1\}, j = 1,...,|\mathbb{Y}_K| \end{array}\right. \quad (2)$$

In the general case, since p_i and w_i change with different configurations of $\mathbf{x}$, the formulation can be

nonlinear. To linearize the formulation, let us denote $\mathbb{Y}_K$ as the set of all K-sized bundles constructed from the set N of n items, and $j \in [1, |\mathbb{Y}_K|]$ uniquely identifying one such bundle. In Formulation (2), the equivalent linearized form of Formulation (1), variable y_j is used to indicate whether bundle j is selected, and $p_i(y_j = 1)$ the profit of item i in bundle j.

IKP can be used to model the *resource bundling* problem in revenue management. For example, in the context of bundling attractions in a theme park, if the *appeal* of an attraction is interpreted to the weight of that attraction, IKP formulates the problem of how *K* attractions can be packaged and offered to the consumers such that its monetary yield is maximized while remaining appealing to the consumers, i.e., the total *appeal* exceeds a certain threshold c. Similarly, IKP naturally models the bundling problem of channel TV subscriptions or sightseeing passes in a City Tour.

3 Complexity of IKP and its subclasses

Note that in the general case when p_i and w_i take arbitrary forms, we need to describe these functions explicitly in tabular form. Since it takes a constant number of bits to indicate the weight and value of each K-sized bundle, the description length L of IKP is $O(n^K)$. The optimization problem can then be solved trivially since an exhaustive, comparison-based search through $\mathbb{Y}_K$ suffices to obtain the most profitable bundle that satisfies the budget constraint; this takes time polynomial in L. IKP in its tabular form is therefore in the **P** complexity class.

The problem becomes much more interesting in cases when p_i and w_i are structured and can be compactly represented. For instance, Proposition 3.1 posits that IKP-poly, a class of IKP in which p_i and w_i can be represented in space polynomial to the number of items n, is NP-hard.

Proposition 3.1. *IKP-poly is **NP**-hard.*

Note that the complexity increment in solving IKP-poly is due to the efficient representation of the profit and weight functions, which reduces the encoding length exponentially. Consider another class of IKP, in which the profit and weight functions have locality effects.

Definition 1. *An IKP is called **locally interacting**, abbreviated as LIKP, if there exists a subset $B_i \subset N, \forall i$ such that $p_i(\mathbf{x}) = p_i(\mathbf{x}(B_i))$ and $w_i(\mathbf{x}) = w_i(\mathbf{x}(B_i)), \forall \mathbf{x}$; $\mathbf{x}(B_i)$ indicates the setting of variables in B_i with respect to* **x**.

The idea of local interactivity is akin to the concept of Markov Blankets in probabilistic graphical models [3], whereby the conditional probability of a node only depends on its immediate neighborhood. The existence of B_i imposes some degree of factoredness on the profit and weight functions, giving rise to a compact representation of the profit and weight function. Indeed, if we denote $b = \max_i |B_i|$, Proposition 3.2 posits an upper bound on the number of encoding bits required to describe an LIKP.

Proposition 3.2. *The description length of LIKP is $O(n2^b + \log K + \log c)$.*

If we treat b, c as constants and $K = O(n)$, LIKP is a subclass of IKP-poly, and similar complexity argument as Proposition 3.1 thus applies, showing that LIKP is also NP-hard. LIKP is an interesting subclass of IKP; it can compactly represent the optimization problem while appropriately reflecting the local intra-bundle interactivity. From the perspective of bundle design in revenue management, the weight of items can represent the items' *appeal* to customers, as the seller tries to create packages that achieve a certain level of appealingness while maximizing profits. Oftentimes, an item's attractiveness is affected by the *types* of items already in the bundle. For instance, in some cases, the existence of similar types boost the bundle's desirability, while in others, type variety is more sought after by the customers.

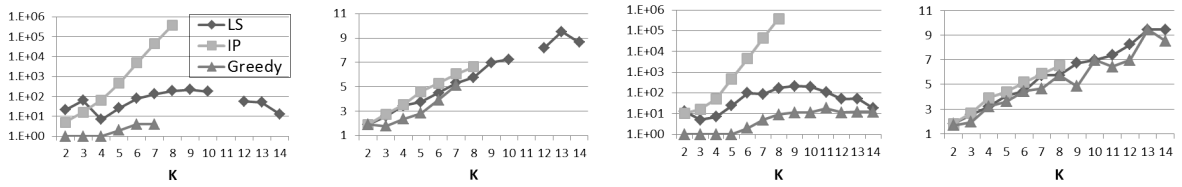
4 Heuristic Solutions for IKP

Formulation 2 can be implemented using Integer Programming (IP), the solution of which is optimal. However, it requires $O(n^K)$ time, which can be exponential in n if K is large, e.g., $K = n/2$. In

this section, we detail two heuristic approaches in solving IKP that yield better running time: Greedy Bundling with time $O(nK^2)$, and Local Search $O(nKT)$; T is the number of search steps.

Greedy Bundling (Greedy). This algorithm constructs the solution bundle by adding items into an initially empty set, one item per step, greedily with respect to profit while making sure the partial weight constraints are satisfied at all steps. The partial constraint is such that $\frac{Lc}{K} \geq \sum_{i \in B_L} w_i(\mathbf{x}(B_L) = \mathbf{1}, \mathbf{x}(N \setminus B_L) = \mathbf{0})$ with B_L being the partial bundle of size L. Greedy therefore incurs only $O(n + 2n + ... + Kn) = O(nK^2)$ time, much more scalable than IP. However, that is achieved at the expense of losing guarantees on solution quality. In fact, because of the way the partial threshold constraint is constructed, it is possible that greedy may not find a solution even when one exists.

Local Search (LS). The method starts with an initial choice of the bundle and tries to improve this set incrementally, one attraction at a time. At each iteration where the bundle does not satisfy the weight constraint, we replace the item with the most weight contribution with one that helps satisfy the constraint; otherwise, a random walk is taken. Each time an improved bundle is encountered, it is saved as the current best solution. Since the algorithm takes at most nK time in considering items to swap in and out at each step, its total running time is $O(nKT)$. LS is guaranteed to converge to the global optimal solution given sufficient time, as it always keeps track of bundles with non-descending rewards.



(a) Time in IKP instance 1 (b) Profit in IKP instance 1 (c) Time in IKP instance 2 (d) Profit in IKP instance 2

Figure 1: IP, Greedy Bundling and Local Search in IKP of size 15. Figures 1a and 1c show the average running time in milliseconds, displayed in log-scale, while 1b and 1d respective profit.

Numerical Experiments. Figure 1 depicts the preliminary performance comparison of Greedy, LS[1] and IP in two randomly generated, synthetic IKP instances with 15 items[2]. As demonstrated in Figure 1, with small values of K, IP always succeeds in producing optimal solutions. However, this comes at the cost of exponential running time, making the approach unfeasible for $K > 7$. Greedy and LS on the other hand are very efficient in terms of running time, i.e., less than one second for any K. However, both Greedy and Local Search (LS) bear the risk of not being able to obtain a feasible solution (instance 1). For Greedy, the short-sightedness in selecting items can produce partial bundles that satisfy partial weight constraint early on, but are impossible to build upon at the later stage. For LS, while guaranteeing to reach a feasible solution if there exists one, the limited number of search steps might be insufficient for such purpose, hence leading to its failure.

Acknowledgment. This research/project is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

[1] W. J. Adams and J. L. Yellen. Commodity bundling and the burden of monopoly. *The Quarterly Journal of Economics*, 90(3):475—498, 1976.

[2] R. E. Dansby and C. Conrad. Commodity bundling. *The American Economic Review*, 74(2):377—381, 1984.

[3] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[4] R. Schmalensee. Gaussian demand and commodity bundling. *The Journal of Business*, 57(1):S211—S230, 1984.

[1] For Local Search, we use Greedy to initialize bundles and set the maximum number of steps T to 100.

[2] The profit and weight functions are randomly generated with values in the interval $(0, 1)$.