Utilizing Housing Resources for Homeless Youth Through the Lens of Multiple Multi-Dimensional Knapsacks

Hau Chan[*] and **Long Tran-Thanh**[†] and **Bryan Wilder, Eric Rice, Phebe Vayanos & Milind Tambe**[‡]

Abstract

There are over 1 million homeless youth in the U.S. each year. To reduce homelessness, U.S. Housing and Urban Development (HUD) and housing communities provide housing programs/services to homeless youth with the goal of improving their long-term situation. Housing communities are facing a difficult task of filling their housing programs, with as many youths as possible, subject to resource constraints for meeting the needs of youth. Currently, the assignment is manually done by humans working in the housing communities.

In this paper, we consider the problem of assigning homeless youth to housing programs subject to resource constraints. We provide an initial abstract model for this setting and show that the problem of maximizing the total assigned youth to the programs under this model is APX-hard. To solve the problem, we non-trivially formulate it as a multiple multi-dimensional knapsack problem (MMDKP), which is not known to have any approximation algorithm. We provide a first interpretable and easy-to-use greedy algorithm with logarithmic approximation ratio for solving general MMDKP. We conduct experiments on random and realistic instances of the housing assignment settings and show that our algorithm is efficient and effective in solving large instances (up to 1 million youth).

1 Introduction

There are over 1 million homeless youth in the United States each year. These are young people between the age of 13 and 24 who are homeless, unaccompanied by family, living outdoors, in places not fit for human habitation, and in emergency shelters (Toro, Lesperance, and Braciszewski 2011). The consequences of youth homelessness are many, including large number of preventable problems such as exposure to violence, trauma, substance use, and sexually transmitted disease (Toro, Lesperance, and Braciszewski 2011). A critical solution to improve long term outcomes for homeless youth is to quickly and efficiently help the homeless youth find safe and stable housing situations.

[*]Corresponding author. University of Nebraska-Lincoln. Email: hchan3@unl.edu.

[†]Corresponding author. University of Southampton. Email: l.tran-thanh@soton.ac.uk.

[‡]University of Southern California. Email: {bwilder,ericr,phebe.vayanos,tambe}@usc.edu.

Many non-profit organizations and public sectors provide temporary housing programs to house eligible homeless youth within their local communities. In almost all communities in the United States, the number of youth experiencing homelessness exceeds the capacity of the housing resources available to youth (Housing and Urban Development (HUD) 2015). This situation leaves communities with the terrible predicament of trying to decide (1) who to prioritize for the precious few spots in housing programs which are available at any given time, and (2) how to assign prioritized youth to these spots subject to various resource constraints.

To tackle these questions, most communities have moved to what is referred to as a Coordinated Entry System. In such systems, most agencies within a community pool their housing resources in a centralized system. Persons who are seeking housing are first assessed for eligibility for housing, which usually includes HUD-defined "chronic homelessness," other criteria such as veteran status, and "vulnerability." Based on these assessments, persons are prioritized for housing and placed on waiting lists until appropriate housing becomes available in the community (Housing and Urban Development (HUD) 2015) (see (Chan et al. 2017) for a more detail overview of the current system). *Despite these efforts on prioritizing youth, almost all of the housing placement and assignment decisions of assigning homeless youth to housing programs are made by humans manually working in the housing communities with low efficiencies.*

Against this background, this paper investigates whether this housing assignment problem can be supported by an AI-based decisionmaker, which can automate the process in an efficient way, both computationally and performance-wise. To do so, we provide an initial model of the current housing assignment process under resource capacity constraints and decision tools that could be of use to housing communities.

1.1 Our Contribution

This paper contributes to the state of the art as follows:

- We study the problem of utilizing the given housing resources by maximizing the "value" of assigned homeless youth to a finite number of housing programs, each with their corresponding resource capacities. To study this problem from a computational perspective, we provide the first analytical model for the homeless youth housing assignment problem (Section 2).

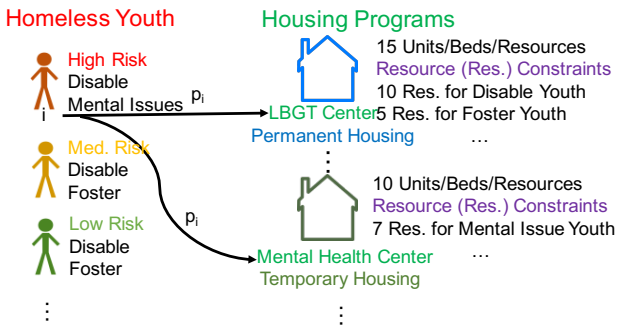

Figure 1: **Assigning Homeless Youth to Housing Programs**. The figure depicts the setting of assigning homeless youth to some housing programs. Homeless youth have different types. Each housing program has some resource capacity on the number and the types of homeless youth that they can accept.

- We show that this problem is APX-hard (Section 3). Therefore, no PTAS solution for this problem exists.
- To (approximately) solve our assignment problem, we reformulate it as a multiple multi-dimensional knapsack problem (MMDKP) nontrivially. While the knapsack literature has been widely studied for a long time, it is quite surprising that the MMDKP does not have approximation algorithms with provable performance guarantees. To address this issue, we provide the first greedy approximation algorithm for MMDKP based on the work of (Dobson 1982) (Section 4).
- We then evaluate our proposed method on several random generated assignment instances (up to 1 million youth) and demonstrate that our algorithm is efficient both in running time and maximizing the number of assigned youth in practice (Section 5).

2 Model of Housing Assignment Under Resource Capacity Constraints

In our setting, we have a set N of homeless youth and a set H of housing programs. Figure 1 depicts a simple abstract scenario of assigning homeless youth to housing programs.

Description of Homeless Youth Each homeless youth $i \in N$ has a subset of types $t_i \subseteq T$ where T denotes the possible types of the homeless youth (e.g., $T = \{registered\ disable, registered\ mental\ health, LGBT, ..., etc\}$). Each youth i is characterized by a set of conditions/types t_i.

Description of the Housing Programs There are certain housing programs available to the homeless youth. We let H to denote the set of housing programs. There are different types of housing programs such as the LBGT permanent, mental health temporary, and LBGT rapidly housing programs. The permanent housing program is designed for long-term staying ($>$ 1 year) while the temporary (6-12 months) and the rapidly (1-6 months) (rental) housing programs are short-term solutions for homeless youth that lost their jobs or stable homes recently. The goal of the program is to house homeless youth so that they can find stable solutions afterward. Each housing program $h \in H$ has c_h housing units available to homeless youth.

However, not all of the homeless youth can be assigned to the available programs/units. As limited resources are provided by some funding agencies such as the U.S. Department of Mental Health, only some homeless youth with certain types can be placed into the units. As an example, homeless youth with registered mental health, disability, and foster issues can be in at most 80%, 0%, and 50% of the available housing units, respectively. Thus, there are some constraints and requirements on how many youth of certain types can be assigned to each of the programs in H.

For $h \in H$, h can provide support to youth containing some types in $t_h \subseteq 2^T$. Due to the resource capacity constraint, for each $t \in t_h$, h can only support at most u_h^t ($\leq c_h$) youth that have type containing t. In other words, youth consume resources of their types in the housing programs.

Assigning Homeless Youth to Housing Programs A housing program can only accommodate youth with certain conditions. A youth $i \in N$ can be assigned to housing program $h \in H$ if there is $t \in t_h$ such that $t \subseteq t_i$. Moreover, there is a value p_{ih} denoting the utility of assigning i to h. In our setting, the value $p_{ih} = p_i$ is independent of h for all $h \in H$. In general, the value p_i can be arbitrary. or based on the probability of success (Chan et al. 2017).

Objective and Goal: Our goal is to assign homeless youth to housing programs that maximizes the overall value of the assignment subject to the housing program resource capacity constraints. To present our optimization objective more formally, we define the following terms. We let $\mathcal{C} = \{(i,h) \in N \times H \ : \ \exists t \in t_h \text{ such that } t \subseteq t_i\}$ to be the set of feasible assignment pairs. We let $x_{(i,h)} \in \{0,1\}$ to denote whether $i \in N$ is assigned to housing program $h \in H$ for all $(i,h) \in \mathcal{C}$. Our problem of assigning homeless youth to housing programs can be formulated as the following optimization problem:

$$\max_{\mathbf{x}} \sum_{(i,h)\in\mathcal{C}} p_i x_{(i,h)} \tag{1}$$

$$\text{subject to} \sum_{i:(i,h)\in\mathcal{C}} x_{(i,h)} \leq c_h \qquad \forall h \in H \tag{2}$$

$$\sum_{i,t\subseteq t_i:(i,h)\in\mathcal{C}} x_{(i,h)} \leq u_h^t \qquad \forall t \in t_h, h \in H \tag{3}$$

$$\sum_{h:(i,h)\in\mathcal{C}} x_{(i,h)} \leq 1 \qquad \forall i \in N \tag{4}$$

$$x_{(i,h)} \in \{0,1\} \qquad \forall (i,h) \in \mathcal{C}. \tag{5}$$

Notice that the above optimization problem is a natural instance of integer programming (IP), and we solve it using techniques for solving IP. However, solving IP in general is not scalable for large problem instances ($\geq$ 50k youth) as we show in the experimental section. Thus, we need an efficient approximation algorithm that works well in practice.

Definition 1. *An assignment $\{x_{(i,h)}\}_{(i,h)\in\mathcal{C}}$ is feasible if and only if it satisfies Constraints (2) - (5). It is optimal if and only it is a feasible assignment obtaining the maximum value of the objective function (1).*

3 Computing an Optimal Assignment

It turns out that, even if the value $p_i = 1$ for all $i \in N$, it is unlikely that we will be able to solve our problem in an efficient manner. In fact, it is hard to get within some constant factor of the optimal solution.

Theorem 1. *It is APX-hard to compute an optimal assignment even with one program h and $u_h^t = 1$ for $t \in t_h$.*

Proof (Sketch). We reduce from the 3-dimensional matching (3DM) problem, which is known to be APX-hard (Ausiello et al. 1999). In an instance of the 3DM, we are given sets A, B, and C ($A \cap B \cap C = \emptyset$) of the same size and $D \subseteq A \times B \times C$, and we want to find a set $M \subseteq D$ of maximum size such that $\forall\{a_1, b_1, c_1\}, \{a_2, b_2, c_2\} \in M$, $a_1 \neq a_2, b_1 \neq b_2$, and $c_1 \neq c_2$.

Given an instance of the 3DM, we reduce it to our problem. We let $T = A \cup B \cup C$. For each $(a_i, b_i, c_i) \in D$, we create a youth $i \in N$ such that $t_i = \{a_i, b_i, c_i\}$. Thus, $|N| = |D|$. For $i \in N$, $p_{ih} = 1$ and $p_{i\perp} = 0$. There is one (non-dummy) program h with $c_h = |N|$, $t_\perp = t_h = A \cup B \cup C$ such that, for each $t \in t_h$, $u_h^t = 1$ and $u_\perp^t = c_h$. We can show that M is maximum size if and only if the assignment is optimal since their objective values are the same. □

Despite this hardness result, it is still worth to investigate whether we have an efficient approximation algorithm with good performance guarantees. In Section 4, we present an efficient approximation algorithm for solving the problem.

4 Approximation Algorithm for Computing an Optimal Assignment

To provide an approximate algorithm for computing an optimal assignment, we first reformulate it as a multiple multi-dimensional knapsack problem (MMDKP). We then derive an approximation algorithm for our problem by introducing a new approximation algorithm to solve any MMDKP. The approximation algorithm for the MMDKP is based on the basic greedy principle by selecting a single MDKP at a time and packs the single MDKP using existing (approximation) algorithm for a single MDKP. The approximation ratio largely depends on the approximation ratio of the algorithm for the single MDKP. The best known ratio for MDKP that runs in polynomial time with respect to the input size is the algorithm of (Dobson 1982) that depends on the logarithmic of the sum of the "weights" of the items.

We begin by first discussing the single MDKP and then present our approximation algorithm for MMDKP.

4.1 Multi-Dimensional Knapsack Problem

In the multi-dimensional knapsack problem (MDKP), we are given a set of items $L = \{1, ..., l\}$ and a d-dimensional knapsack with non-negative integer capacities $c_1, ..., c_d$ for the d dimensions. For each item $i \in L$, there a value v_i for packing the item into the knapsack and weights $w_{i1}, ..., w_{id}$ for the d dimensions. The goal is to find a subset of items $S \subseteq L$ such that the overall value of the items is maximized without exceeding the capacities of the d dimensions. More specifically, $S \in \arg\max_{S' \in F(L)} \sum_{i \in S'} v_i$ where $F(L) = \{S \subseteq L \mid \sum_{i \in S} w_{ij} \leq c_j,\ \forall j = 1, ..., d\}$ consists of the feasible subsets of L.

The best known approximation algorithm for MDKP (that runs in polynomial time) is to greedily add items with the best bang-for-bucks without exceeding the capacity (Dobson 1982; Kellerer, Pferschy, and Pisinger 2004). Hereafter we refer to this algorithm as BfB (bang-for-bucks), and it can be described as follows: Defining the bang-for-buck of item $i \in L$ to be $e_i = \frac{v_i}{\sum_{j=1}^{d} w_{ij}}$, the greedy algorithm that sorts items according to the $e_i's$ and packs the current highest bang-for-bucks items, one by one, into the knapsack without violating the capacity constraints yields the following approximation.

Theorem 2 ((Dobson 1982))**.** *The* BfB *algorithm gives $\frac{1}{O(H(\max_{i\in L}\sum_{j=1}^{d} w_{ij}))}$-approximation to the MDKP where $H(n)$ is the first n terms of the harmonic series.*

The above theorem states that the approximation ratio depends on the logarithmic of the weights of the items.

4.2 Multiple Multi-Dimensional Knapsack Problem

In the typical multiple knapsack setting, each knapsack is only one dimension. We can further extend this notion to multiple multi-dimensional knapsack where each knapsack is now multi-dimension. More formally, we are given a set of items L with value v_i for each $i \in L$ and a set of d-dimensional knapsacks K where for each $k \in K$, there is a capacity c_{jk} for each dimension $j = 1, ..., d$. The goal is to find $(S_1, ..., S_k)$ such that $(S_1, ..., S_k) \in \arg\max_{(S_1', ..., S_k') \in F^k(L)} \sum_{i \in \bigcup_{k\in K} S_k'} v_i$ where $F^k(L) = \{(S_1, ..., S_k) \subseteq L^k \mid (a) \bigcap_{k\in K} S_k = \emptyset$ and $(b) \sum_{i\in S_k} w_{ij} \leq c_{jk},\ \forall j = 1, ..., d, k \in K\}$ and S_k is the set of items packed to knapsack $k \in K$.

Given an instance of MMDKP, it turns out that a simple approach that greedily select a single multi-dimensional knapsack to pack until all of the knapsacks are filled works very well. In particular, suppose that we a $f(W)$-approximation for the MDKP (i.e., the multi-dimensional knapsack problem), where f is some function parametrized by W with $f(W) > 1$. Then by using this approximation algorithm to fill each multi-dimensional knapsack, we can provide a $O(\frac{1}{2f(W)})$-approximation in total to the MMDKP.

Theorem 3. *Given any $\frac{1}{O(f(W))}$-approximation to the MDKP, the greedy algorithm that sequentially packs each multi-dimensional knapsack in the MMDKP yields $\frac{1}{O(2f(W))}$-approximation.*

The above proof follows from (Kellerer, Pferschy, and Pisinger 2004; Chekuri and Khanna 2000). Combining the above and Theorem 2, we have the following result.

Theorem 4. *By applying* BfB *to the abovementioned greedy approach we achieve $\frac{1}{O(2H(\max_{i\in L}\sum_{j=1}^{d} w_{ij}))}$-approximation to the MMDKP where $H(n)$ is the first n terms of the harmonic series.*

Algorithm 1 The greedyMMD Algorithm

Inputs:
MDKPs - $\forall k \in K, j = 1 \dots d$: $c_{jk} \geq 0$;
List of items L - $\forall i \in L$: value v_i, weight $w_i =< w_{i1}, \dots, w_{id} >$
Main part:
$\forall i \in L'$: calculate $e_i = \frac{v_i}{\sum_{j=1}^{d} w_{ij}}$
$K' = K$ //list of remaining unused knapsacks
$L' = L$ //set of remaining unpacked items
while unused knapsack exists **do**
  choose arbitrary $k \in K'$
  while packing k is feasible **do**
    Pack item $I := \arg\max_{i \in L'} e_i$ into k
    Remove I from L'
  end while
  Remove k from K'
end while

For the sake of simplicity, hereafter we refer to the approach (used and described in Theorem 5) as greedyMMD (for greedy multiple multi-dimensional knapsack). The pseudo code of this algorithm is depicted in Algorithm 1.

Finally, notice that Theorem 2 is the best known approximation algorithm based on the weights of the items, and we use it to establish the main result of Theorem 5. In general, we can plug-in the fixed dimension PTAS result of (Caprara et al. 2000) and obtain approximation ratio of $\frac{(1-\epsilon)}{2}$ for some $\epsilon > 0$ with running time exponential in $O\left(\frac{d}{\epsilon}\right)$. Such algorithms might not be feasible for large problem instances and not as easy to use as BfB.

4.3 Assignment Problem to MMDKP Reduction

Given an instance (N, H, T) of our homeless youth assignment problem, we can reduce it to an instance of the MMDKP. A straight forward reduction is to represent the homeless youth as items and create a MDKP for each program with appropriate capacity constraints and (binary) weights for the homeless youth. However, this direct approach has a main drawback: the overall dimension of the MDKP can be exponential in $|T|$. Beyond the representation and computational issues, the approximation guarantee in Theorem 5 becomes quite useless (i.e., also exponential in $|T|$ in the worst case)!

It turns out that we can have a more compact MDKP for each program $h \in H$ by creating one with dimension $|t_h|+1$ instead of the overall dimension $2^{|T|} + 1$ and considering the items/homes youth that can be assigned to h. Previously, we would have to explicitly represent each homeless youth type regardless of whether it can be assigned to a particular housing under the direct reduction. This formulation makes the problem much more compact.

More precisely, for $h \in H$, we create a MDKP with $L_h = \{i \in N \mid \exists t \in t_h, t \subseteq t_i\}$ and a $d = |t_h| + 1$ dimensional knapsack where each dimension corresponds to a subset of type in t_h and the additional dimension corresponds to the overall capacity constraint. Furthermore, let $c_t = u_h^t$ for each $t \in t_h$ and $c_d = c_h$. For $i \in L_h$, $v_i = p_i$ and $w_{it} = 1$ if $t \subseteq t_i$ and $w_{it} = 0$ otherwise for $t \in t_h$ and $w_{id} = 1$. Applying greedyMMD (Algorithm 1) to the size-variant MMDKP, we can obtain a better approximation.

Theorem 5. *The greedy algorithm* greedyMMD *computes an assignment that gives* $\frac{1}{O(2\log(\max_{h \in H} |t_h|+1))}$*-approximation to the optimal assignment social welfare.*

Proof (sketch). The key to prove this theorem is to show that the optimal solution values opt^{sv}, opt, and opt^a of the size-variant MMDKP, the original MMDKP, and the optimal assignment social welfare are the same, respectively.

We begin by showing the equivalence of the first two. Let $(S_h)_{h \in H}$ and $(S_h^{sv})_{h \in H}$ be any feasible sets of of items packed into the MMDKP under the original and size-variant settings, respectively. We need to show that $(S_h)_{h \in H}$ ($(S_h^{sv})_{h \in H}$) is also a feasible solution of the size-variant setting (original setting). For any $h \in H$, $S_h \subseteq L_h$ and items in S_h must satisfy the capacity constraints of the size-variant MMDKP (since they are the same as the original MDKP minus the items that cannot be packed to h). Since also $\bigcap_{h \in H} S_h = \emptyset$, $(S_h)_{h \in H}$ is feasible solution of the size-variant setting. Similar argument holds for showing $(S_h^{sv})_{h \in H}$ is a feasible solution of the original setting. Thus, their feasible solution sets are equivalent and $opt^{sv} = opt$.

Applying BfB for the single MDKP to the original MDKP and the size-variant MDKP instances for a program h, we obtain the same feasible solution (as the bang-per-buck of the items that can be packed to the knapsack are the same for the two instances) where the latter case has a better approximation bound due to a more compact representation. Thus, greedyMMD for our size-variant instance yields the approximation that depends on the maximum size of the t_h.

Finally, let $(S_h^{sv})_{h \in H}$ be a feasible solution of the size-variant MMDKP. We can construct a feasible assignment by setting $x_{ih} = 1$ for $i \in S_h^{sv}$ for all $i \in N$ and $h \in H$ and $x_{ih} = 0$ for $i \in N \setminus \bigcup_{h \in H} S_h^{sv}$. Indeed, any feasible assignment is also a feasible solution of the size-variant MMDKP. Hence, $opt^{sv} = opt^a$ and we obtain our claimed result. □

4.4 Practical Implication of Our Approach

A strong advantage of our algorithm greedyMMD is that it is very simple and intuitive, even for non-technical audiences in the housing community. It is also fairly straight forward to explain and implement (i.e., assign youth that consume the least amount of resources).

Another advantage of greedyMMD is that it is invariant to the order of the knapsacks. As such, when it comes to the application to the homeless youth housing assignment domain, the algorithm, as well as the management of each housing program, only needs to care about the assignment of their own program, independently from what have happened with the other programs (or will happen with the future ones). This significantly simplifies the administrative overhead work needs to be done between different housing programs. In addition, this advantage allows the algorithm to be deployed in a distributed manner in each program, and thus, can act as an *autonomous intelligent agent* on the behalf of the management of each particular housing program.

Finally, our algorithm can also work in online settings where new knapsacks sequentially arrive to the system (i.e.,

we do not have access to all the knapsacks at the beginning). This directly follows from the proof of Theorem 3. This property is especially important for application domains such as the homeless youth housing assignment problem, where many new programs are launched time-by-time.

5 Experiment

In this section, our main goal is to evaluate the running time and performance of greedyMMD for various instances of our homeless youth assignment problems. The IP program presented earlier would be used as a benchmark to compare to the solution generated by greedyMMD.

For this purpose, we consider two main instances of housing assignment problems based on (1) "hard" and (2) "realistic" instances of the problems. The "hard" instances are the 3-dimensional matching (3DM) instances in our APX-hard proof. We believe these instances will most likely the hardest instances we will encounters and would be a good test of concept to show the efficiency and effectiveness of greedyMMD. The "realistic" instances of the problems are based on the current housing matching criteria and the approximate number of programs/capacities available in a single snapshot of a 3-month period. In both cases, we generate them randomly according to some parameter settings.

We will begin by presenting our experimental setups and results on the 3DM instances and then we will move on to discuss the realistic instances.

5.1 Experiments on Solving Hard 3DM Instances

Recall that in a 3DM housing assignment instance, there is only one program h with $t_h = T = A \cup B \cup C$ and $u_h^t = 1$ for $t \in T$ where A, B, and C are disjoint sets of types of equal sizes (i.e., $A \cap B \cap C = \emptyset$ and $|A| = |B| = |C|$). Each youth $i \in N$ has types $(a_i, b_i, c_i) \in A \times B \times C$ and $p_i = 1$. The goal is to find a maximum set of youth of disjoint types without violating the resource capacity constraints.

As such, in our experiments, we consider different numbers of youth in $\{100, 500, 1000, 5000, 10000\}$ and different sizes of the set of types in $\{5, 10, 15, 20, 25, 30, 35, 40\}$ (i.e., the size of A, B, and C and the number of types is three times the set size). The youth's type is generated randomly by selecting an element from each set (out of the three) of a given set size. For each possible combination of number of youth and set size, we generate 100 different instances and report the running time of the IP and greedyMMD measured in seconds. The performance of greedyMMD is computed as the ratio of the solution generated by greedyMMD divided by the optimal solution (OPT) generated by the IP. We report the average and the minimum ratio values of the 100 instances for different parameter settings.

Figure 2 shows the running time of the IP and greedyMMD, and greedyMMD's performance ratio for a fix number of youth (top plot) and a fix number of set size (bottom plot). From the top plot, we observe that the running time of the IP grows exponentially as the number of set size (types) increase (as indicated by the linear growth in the logarithmic y-axis scale) while the running time of our greedyMMD grows linearly. The performance ratio is quite high ($\geq$ .90) but it seems to decrease as the number of set size increases. However, the bottom plot suggests

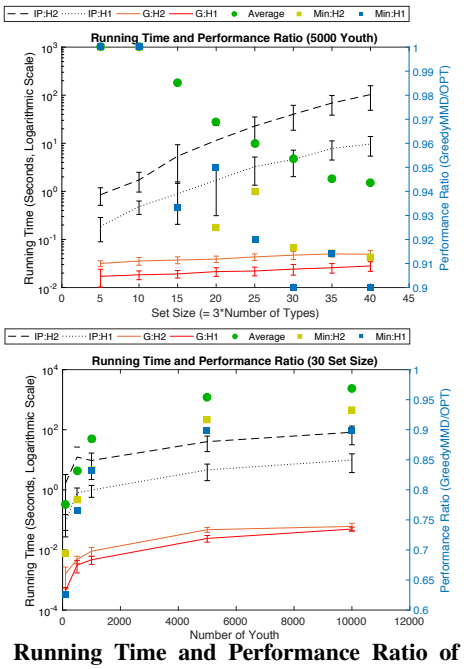

Figure 2: **Running Time and Performance Ratio of IP and** greedyMMD**.** The top plot shows the running time (left y-axis) of IP and greedyMMDwith one (H1) and two (H2) programs along with the performance ratio (right y-axis) for 5000 youth and various set size (x-axis). The bottom plot shows the same except the number of youth varies (x-axis) and 30 set size. Other plots of different settings are similar.

that as the number of youth increases, the performance ratio increases as well. Finally, the running times of the IP and greedyMMD grow linearly with the number of youth for a fixed set size. These suggests that our greedyMMD is efficient (much more than the IP) and obtains good performance when we have a large number of youth, which is generally the case in the homeless youth setting.

5.2 Experiments on Solving Realistic Instances

In the realistic instance setting, we consider the current housing matching criteria as types such as the one below.

Subsidy Eligibility Criteria						*Match Ready Criteria*	
Acuity?	Chronic?	Mental Health?	Substance Abuse?	"Special Needs"?	Criminal Background?	CA ID and SSCard IN HAND ?	Housing Navigator?

Except the first question for acuity, which has three possible answers, all of the other questions are binary. Roughly speaking, acuity is a measure of risk level (low, medium, and high) (Chan et al. 2017). As a result, the type set $T = \{LowAcuity, HighAcuity, ..., NoMentalHealth, YesMentalHealth, ...\}$ and the youth's types are the subsets of T of size 8 and the answers to these questions. Each program h in this setting has $t_h \subseteq T$, resource capacity of $u_h^t \sim uniform[1, 25]$, and the overall capacity of $c_h \sim uniform[1, 50]$ drawn from the uniform distribution of the given supports. We initial the $p_i \in [0, 1]$ for all i to be some random value in between 0 and 1. For the running time and performance ratio com-

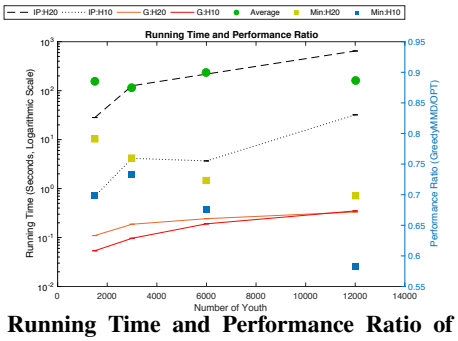

Figure 3: **Running Time and Performance Ratio of IP and** greedyMMD**.** The plot shows the running time (left y-axis) of IP and greedyMMDwith 10 (H10) and 20 (H20) programs along with the performance ratio (right y-axis) for different number of youth.

parisons with the IP, we consider smaller instances setting of the number of youth in $\{1500, 300, 600, 12000\}$ and the number of programs in $\{10, 20\}$ due to feasibility. For the in-house comparison without the IP, we consider much larger instances with the number of youth in $\{5000, 10000, 50000, 100000, 500000, 1000000\}$ and the number of programs in {10, 20, 30}. We consider 50 randomly generated instances for each possible combination.

Benchmarking with IP As in earlier experiments, we consider the running time and performance ratio of the greedyMMD. In this setting, the running time of the IP grows drastically as the number of youth and the number of program increase (Figure 3). As a result, it is infeasible for us to run the IP on the larger instances. Figure 3 shows that the average performance ratio stays around .9 with the worst performance ratio (of the single instance) at around .7 and .5 for the 20-program and 10-program settings, respectively.

Interestingly enough, as the number of youth (number of programs) increase, the worst case performance decreases (increases) while maintaining similar average performance ratio. While the worst case performance does not look too pessimistic, we expect our greedyMMD to perform better on average ($\sim$.9).

Solving Large Instances with greedyMMD We consider the running time of greedyMMD and its performance relative to other simple heuristics on large instances up to one million youth. We consider random (Rand) and prioritize (Prio) heuristics where random youth and prioritized youth (by various factors such as low, medium, high acuity) are added into the programs, respectively, under the same greedy method (instead of the bang-for-bucks).

Figure 4 shows the running time and performance of greedyMMD on large instances. It is no surprise that the running time of greedyMMD grows linearly in the number of youth even with different numbers of programs (top of Figure 4). Moreover, as observe in Figure 4 (bottom plot), the random and prioritization heuristics obtain only 50% of the social welfare generated by greedyMMD. These provide additional evidence that our greedyMMD is effective and efficiently in solving large instances of our homeless youth housing assignment problems.

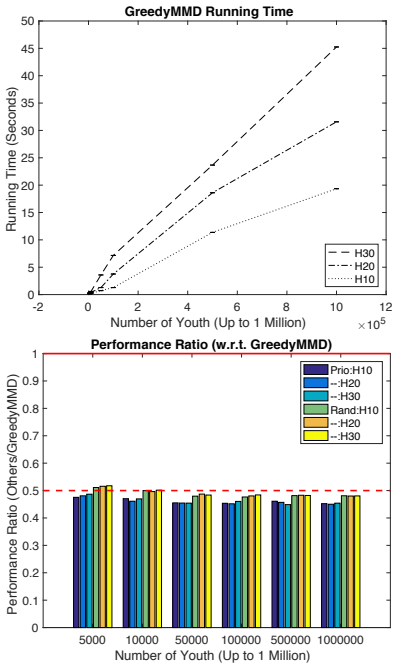

Figure 4: greedyMMD **Running Time and Performance Ratio.** The top plot shows the running time (y-axis) of greedyMMD with ten (H10), twenty (H20), and thirty (H30) programs for different number of youth up to one million. The bottom plot shows the relatively performance ratio of random and prioritization heuristics w.r.t. greedyMMD.

6 Related Work

A related problem is the display ads setting (Mehta 2013). The display ads setting is essentially the bipartite matching problem in which there is a packing capacity constraint for each node in a bipartition specifying the number of times it can be matched.

Our problem is an instance of the widely studied knapsack literature (Kellerer, Pferschy, and Pisinger 2004). While there are many existing heuristics for these knapsacks, surprisingly (to our best knowledge) there are no heuristics with theoretical guarantees for the MMDKP. Notice that (Song, Zhang, and Fang 2008) claim that they have a greedy based solution with $\frac{1}{2}$ ratio. However, their work does not contain any detailed proof. In addition, their result contradicts to many of the other well known theoretical results in the literature (e.g., the work of (Dobson 1982; Kellerer, Pferschy, and Pisinger 2004)).

7 Conclusions

In this paper we investigated the problem of housing assignment for homeless youth. In particular, we have proved that it is APX-hard. We have also proposed greedyMMD, and efficient greedy algorithm with provable performance guarantees, which makes it first of its kind in both the multiple multidimensional knapsack and homeless youth housing assignment literature. This algorithm has several advantagous properties which together make it practical, and thus, deployable as a core algorithm for an intelligent agent to autonomously and efficiently manage the homeless youth housing assignment programs.

References

Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; and Protasi, M. 1999. *Complexity and Approximation*. Springer-Verlag Berlin Heidelberg.

Caprara, A.; Kellerer, H.; Pferschy, U.; and Pisinger, D. 2000. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research* 123(2):333 – 345.

Chan, H.; Rice, E.; Vayanos, P.; Tambe, M.; and Morton, M. 2017. Evidence from the past: Ai decision aids to improve housing systems for homeless youth. In *AAAI Fall Symposium Series*.

Chekuri, C., and Khanna, S. 2000. A ptas for the multiple knapsack problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, 213–222. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Dobson, G. 1982. Worst-case analysis of greedy heuristics for integer programming with nonnegative data. *Mathematics of Operations Research* 7(4):515–531.

Housing and Urban Development (HUD). 2015. Coordinated Entry Policy Brief. https://www.hudexchange.info/resources/documents/Coordinated-Entry-Policy-Brief.pdf.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Knapsack Problems*. Springer, Berlin, Heidelberg.

Mehta, A. 2013. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.* 8(4):265–368.

Song, Y.; Zhang, C.; and Fang, Y. 2008. Multiple multidimensional knapsack problem and its applications in cognitive radio networks. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, 1–7.

Toro, P. A.; Lesperance, T. M.; and Braciszewski, J. M. 2011. *The heterogeneity of homeless youth in America: Examining typologies*. National Alliance to End Homelessness: Washington, DC.