Online Capacitated General Matching with Knapsack

Ruoyu Wu[1], **Wei Bao**[1], **Ben Liang**[2], **Hequn Wang**[1]

[1]The University of Sydney
[2]University of Toronto
ruoyu.wu@sydney.edu.au, wei.bao@sydney.edu.au, liang@ece.utoronto.ca, hwan0565@uni.sydney.edu.au

Abstract

We study a new online matching problem termed Online Capacitated General Matching with Knapsack (OCGMK), which generalizes the Online General Matching (OGM) problem. In the original OGM, vertices arrive sequentially and need to be paired with other vertices to maximize the total reward of pairing. Our study is the first to consider capacitated vertices in OGM: we allow each vertex to be assigned to multiple vertices up to a capacity limit. We also consider a previously unexamined knapsack constraint in OGM: assigning a pair of vertices has a cost, but the total cost is budgeted. To solve the OCGMK problem, we propose the Online Capacity-Knapsack Assignment (OCKA) algorithm, which constructs capacity-friendly sets and knapsack-friendly sets to simultaneously and effectively address both constraints. OCKA achieves a competitive ratio of $\alpha = \frac{\gamma}{2\beta}$, where $\gamma = 1/(3+e^{-2})$ and β is the ratio between the overall cost of all edges and the cost budget. When the knapsack constraint is not imposed but the capacitated vertices remain, the competitive ratio of OCKA is $\alpha' = 1/2$, recovering the previous best result for single-capacity OGM. We implement trace-driven experiments to evaluate the practical performance of OCKA on a real-world dating dataset, demonstrating the superior performance of OCKA in online dating applications.

1 Introduction

In this paper, we introduce Online Capacitated General Matching with Knapsack (OCGMK), which is a natural generalization of the Online General Matching (OGM) problem. In the original OGM (Ezra et al. 2020), the vertices of an undirected graph arrive sequentially in an online manner. When a vertex arrives, the rewards for edges incident on the vertex are revealed. The decision maker makes an irrevocable decision to either assign (match) an arriving vertex to a previously arrived but unmatched vertex, provided that there is an edge between them, or leave the vertex unmatched. Each vertex has a single-vertex matching capacity so that once two vertices are matched, neither can be matched to another vertex. The decision maker's objective is to maximize the overall reward without knowledge of future vertex arrivals or the rewards of the edges incident on future vertices. OGM is a generalization of the Online Bipartite Matching (OBM) problem (Karp, Vazirani, and Vazirani 1990; Gamlath et al. 2019; Lowalekar, Varakantham, and Jaillet 2020; Wu, Bao, and Ge 2023), which has a wide range of applications, including ride-sharing, Internet advertising, and crowd-sourcing (Dickerson et al. 2021; Mehta et al. 2013).

OCGMK further generalizes OGM to accommodate the following two important features in many practical applications. We consider the ***multi-capacity constraint***: each vertex is capacitated, with a certain capacity to be matched to multiple other vertices. We also consider the ***knapsack constraint*** (Zhou, Chakrabarty, and Lukose 2008; Zhang, Li, and Wu 2017; Sun et al. 2022): each edge has a cost, and the total cost of accepted edges in vertex matching is limited by a budget. With these generalized features, which have not been studied in prior work as far as we are aware, OCGMK can be applied to model a substantially larger number of real-world problems, such as user pairing in dating applications, device connection in wireless sensor networks (Yang et al. 2013), and file sharing in P2P networks. Further details of these applications can be found in our tech report (Wu et al. 2025).

The multi-capacity constraint and the knapsack constraint together make OCGMK more realistic but substantially more challenging: (i) First, the ***multi-capacity constraint*** introduces another dimension in optimizing the matching among vertices, which has never been considered in the context of OGM (Goyal 2022; Ezra et al. 2024; Ma, MacRury, and Nuti 2024). In OCGMK, each vertex may exist in more than just the binary states of "matched" and "unmatched". It may be tempting to decompose multi-capacity vertices into multiple single-capacity vertices and then solve the resultant OGM problem. However, this approach can result in deteriorated performance, since it matches two vertices multiple times, leading to invalid matching. (ii) Second, the additional ***knapsack constraint*** is inherently challenging, which requires careful cost management to avoid prematurely depleting the budget on low-reward items (edges), which is a predicament often encountered in worst-case scenarios (Dean, Goemans, and Vondrák 2008). None of the studies on the online knapsack problem (OKP) addressed the knapsack constraint in the context of OCGMK, and applying these solutions directly to OCGMK would exhaust a vertex's matching capacity too early, missing out on potentially higher rewards in the future, even if the remaining

cost budget is still sufficient. (iii) Third, the combination of multi-capacity and knapsack constraints introduces a ***coupled evolution*** between capacity utilization and cost budget consumption, which requires a delicate matching strategy to counteract such coupled effects. In a worst-case scenario, an adversary could manipulate the graph and the arrival of vertices to exhaust the budget early, leaving vertex capacities underutilized, or vice versa.

The original OGM can be solved with an Online Contention Resolution Scheme (OCRS) (Ezra et al. 2020; Feldman, Svensson, and Zenklusen 2021). Separately, it has been shown that the OCRS is also effective in the OKP (Jiang, Ma, and Zhang 2022). However, none of the existing OCRS frameworks can simultaneously address the multi-capacity constraint and the knapsack constraint in OCGMK. Therefore, a new OCRS solution is needed to strike the right balance between capacity usage and budget consumption. To the best of our knowledge, we are the first to consider either multi-capacity vertices or a possible knapsack constraint in the OGM. We provide detailed discussions in our tech report (Wu et al. 2025) to compare our work with related literature.

Our Contribution To address the new challenges in OCGMK, we develop the Online Capacity-Knapsack Assignment (OCKA) algorithm.

(i) We design a novel probabilistic matching approach to manage the usage of vertices' capacities. During the online matching process, for vertices with different remaining capacities, we design different probabilities to match them with other vertices. Furthermore, these probabilities are delicately tuned to achieve a subtle balance between consuming a capacity for instant reward and reserving a capacity for higher future reward, so that the vertex capacities are well utilized even without knowledge of the future vertex arrivals.

(ii) Additionally, we design a matching acceptance strategy that refines the probabilistic matching approach, ensuring our decisions align with the cost budget. We track the consumption of the cost budget during the online process and decide whether to consume the budget for an instant reward or to save it for a higher future reward. This matching acceptance approach is tuned to avoid both early depletion and wastage of the cost budget.

(iii) We develop a new OCRS framework to integrate the probabilistic matching approach and the matching acceptance approach and to counteract the coupled evolution of capacity usage and cost consumption. Overall, OCKA achieves a competitive ratio of $\alpha = \frac{\gamma}{2\beta}$ for OCGMK, where $\gamma = \frac{1}{3+e^{-2}} \approx 0.319$ and β is the ratio between the expected overall cost of all edges and the cost budget.

In addition, when the knapsack constraint is not imposed on OCGMK but the capacitated vertices remain, OCKA achieves a competitive ratio of $1/2$. This matches the current best result obtained for the single-capacity OGM (Ezra et al. 2020), demonstrating the superior performance of OCKA despite the increased challenge of capacitated vertices.

2 The OCGMK Problem

In this section, we present a formal description of the OCGMK problem. We also provide a summary of the notations introduced in this section in our tech report (Wu et al. 2025) for reference, while the main paper is self-contained, with all symbols clearly defined within the text.

2.1 Problem Formulation

OCGMK is based on an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, |V|\}$ denotes the vertex set and E denotes the edge set. We use $e \in E$ and $(u, v) \in E$ interchangeably to denote an edge in E, where u and v are two vertices in V. For a vertex $v \in V$, we define $N(v)$ as the set of all neighbors of vertex v, and $E(v)$ as the set of all edges incident to vertex v.

In graph G, each vertex v has an integer capacity c_v, such that $c_v \geq 1, \forall v \in V$. Each edge $e \in E$ has a deterministic positive cost $d(e)$ and a random positive reward $r(e)$. The reward $r(e)$ of each edge follows an independent discrete distribution $\mathcal{R}(e)$. The decision maker has a cost budget K, and it has to select a subset of E, denoted as $\hat{E}$, to maximize the overall reward $\sum_{e \in \hat{E}} r(e)$ under the following new constraints introduced in this paper: (i) *Multi-capacity constraint.* In set $\hat{E}$, each vertex v can only connect to up to c_v edges, i.e., $|\hat{E} \cap E(v)| \leq c_v, \forall v \in V$. (ii) *Knapsack constraint.* The overall cost of edges in $\hat{E}$ does not exceed the budget K, i.e., $\sum_{e \in \hat{E}} d(e) \leq K$. We assume that the cost budget K is no less than the cost $d(e)$ of each edge. Otherwise, if there exists an edge e' whose cost $d(e')$ is larger than the cost budget K, we simply remove the edge e', since the decision maker will never select this edge e' in the subset $\hat{E}$. We further assume that the cost budget K is less than the overall cost of all edges, i.e., $K < \sum_{e \in E} d(e)$. Otherwise, the knapsack constraint becomes trivial.

For each edge e, the decision maker either accepts this edge by setting $x_e = 1$ or discards this edge by setting $x_e = 0$ (detailed in Section 2.2). The decision maker's objective is to maximize the cumulative reward. We formulate this optimization problem as problem **P**:

$$\mathbf{P}: \max_{x_e, \forall e \in E} \sum_{e \in E} x_e r(e) \tag{1}$$

$$\text{s.t.} \sum_{e \in E(v)} x_e \leq c_v, \forall v \in V, \tag{2}$$

$$\sum_{e \in E} x_e d(e) \leq K, \tag{3}$$

$$x_e \in \{0, 1\}, \forall e \in E, \tag{4}$$

where $r(e) \sim \mathcal{R}(e), \forall e \in E$. Constraint (2) is the capacity constraint, constraint (3) is the knapsack constraint, and constraint (4) defines the (binary) decision space.

2.2 Online Environment and Competitive Ratio

We consider an online setting with $|V|$ time slots. The vertices in V arrive sequentially, one at a time. Let the permutation (arriving order) σ of vertices be a mapping from and to

$\{1, 2, \ldots, |V|\}$, such that in time slot $t \in [|V|]$, the vertex arriving in time slot t is $\sigma(t)$. We use σ^{-1} to denote the inverse function of σ, such that for each vertex $v \in V$, $\sigma^{-1}(v)$ is the time slot that v arrives. For each vertex v, we use $N_\sigma^-(v)$ to denote the neighbors of v that have arrived before v, i.e., $N_\sigma^-(v) = \{\sigma(t) \in N(v) \mid t \in [1, \sigma^{-1}(v) - 1]\}$. We use $N_\sigma^+(v)$ to denote the neighbors of v that arrive after v, i.e., $N_\sigma^+(v) = \{\sigma(t) \in N(v) \mid t \in [\sigma^{-1}(v) + 1, |V|]\}$. Correspondingly, we define $E_\sigma^-(v) = \{(u, v) \in E(v) \mid u \in N_\sigma^-(v)\}$ and $E_\sigma^+(v) = \{(u, v) \in E(v) \mid u \in N_\sigma^+(v)\}$.

Before the first time slot $t = 1$, the decision maker knows the topology of the undirected graph $G = (V, E)$, the vertex capacities $\{c_v\}$, the distributions $\{\mathcal{R}(e)\}$ of edge rewards, the costs $\{d(e)\}$ of all edges, and the cost budget K. The decision maker does not know the arriving order σ of vertices or the exact reward $r(e)$. When vertex $v = \sigma(t)$ arrives in time slot t. Then, the reward $r(e)$ of each edge e in set $E_\sigma^-(v)$ is realized from the distribution $\mathcal{R}(e)$ and becomes known to the decision maker. For each edge $(u, v) \in E_\sigma^-(v)$, the decision maker must make an irrevocable decision $x_{(u,v)}$ either to accept this edge ($x_{(u,v)} = 1$) or to discard it ($x_{(u,v)} = 0$). If the decision maker accepts edge (u, v), it will get the reward $r((u, v))$ of this edge. If the decision maker discards an edge, it cannot accept that edge in future time slots. The decision maker will immediately discard an edge (u, v) if accepting it would violate either (i) the capacity constraint of any vertex or (ii) the knapsack constraint.

We use I to denote a problem instance, defined by the given graph $G = (V, E)$, distributions of edge rewards $\{\mathcal{R}(e)\}_{e \in E}$, edge costs $\{d(e)\}_{e \in E}$, capacities of vertices $\{c_v\}_{v \in V}$, cost budget K, and permutation σ of vertices. For a problem instance I, the mapping σ^{-1} and the sets $N_\sigma^-(v)$, $N_\sigma^+(v)$, $E_\sigma^-(v)$, and $E_\sigma^+(v)$ are also specified. A problem instance I is randomized by the distribution $\{\mathcal{R}(e)\}$. For a problem instance I, a realization i is determined when the reward $r(e)$ of each edge e is realized from its distribution $\mathcal{R}(e)$.

For a realization i, we denote the average reward obtained by an online algorithm ALG over algorithm randomness as $\text{ALG}(i)$ and the optimal overall reward obtained by the offline optimal algorithm OPT as $\text{OPT}(i)$, where i is fully known to OPT in advance, including vertices' permutation σ and the realized edge rewards $\{r(e)\}$. We denote the average performance of the online algorithm ALG over all realizations i of the problem instance I as $\mathbb{E}_{i \sim I}[\text{ALG}(i)]$. Similarly, the average performance of the offline optimal algorithm OPT over all realizations i of the problem instance I is denoted by $\mathbb{E}_{i \sim I}[\text{OPT}(i)]$.

To evaluate the performance of an online algorithm ALG, we use the competitive ratio and say that ALG is α-competitive if

$$\mathbb{E}_{i \sim I}[\text{ALG}(i)] / \mathbb{E}_{i \sim I}[\text{OPT}(i)] \geq \alpha, \forall I. \tag{5}$$

This competitive ratio is the ratio between the average online performance of $\text{ALG}(i)$ against the average performance of the offline optimal algorithm OPT in the worst-case scenario, where the problem instance I is arranged by an adversary. Eq. (5) is also adopted in previous works on OGM (Ezra et al. 2020) and the Online Stochastic Knapsack Problem (OSKP) (Jiang, Ma, and Zhang 2022).

Algorithm 1: OCKA Algorithm

1: **(GLOBAL) INPUT:** G, $\{\mathcal{R}(e)\}$, $\{c_v\}$, $\{d(e)\}$, K.
2: *# Offline Preparation Phase*
3: Calculate $\{y_e\}$ by Eq. (9).
4: **for** $t = 1, 2, \ldots, |V|$ **do**
5: Vertex $\sigma(t)$ arrives. Reward $r(e)$ of every edge $e \in E_\sigma^-(\sigma(t))$ becomes known.
6: *# Online Assignment Phase*
7: $v \leftarrow \sigma(t)$.
8: $q_t^{\text{c}} \leftarrow \texttt{Assign}(t, v, E_\sigma^-(v), \{r(e)\}_{e \in E_\sigma^-(v)})$
9: *# Assignment Acceptance Phase*
10: $Q_t \leftarrow \texttt{Accept}(t, v, E_\sigma^-(v), q_t^{\text{c}})$
11: $Q \leftarrow Q \cup Q_t$, $t \leftarrow t + 1$.
12: **end for**

3 Algorithm Design

In this section, we present the OCKA algorithm (Alg. 1), which consists of three phases: (i) Before the first vertex arrives, OCKA uses an ***Offline Preparation*** phase (Lines 2–3) to compute an *average preference* y_e for each edge $e \in E$, which will be used later to help determine the likelihood of selecting e (Section 3.1). (ii) When vertex $v = \sigma(t)$ arrives in each time slot t, OCKA first enters the ***Online Assignment*** phase (Lines 6–8) and calls the `Assign` function (Alg. 2) to determine the *capacity-friendly set* q_t^{c} through a probabilistic matching approach. The capacity-friendly set q_t^{c} is a subset of edges in $E_\sigma^-(v)$ that maximizes reward accumulation while respecting the capacity constraint of each vertex (Section 3.2). (iii) OCKA then enters the ***Assignment Acceptance*** phase (Lines 9–11) for time slot t and calls the `Accept` function (Alg. 3) to determine the *knapsack-friendly set* q_t^{k}, which is a subset of edges in $E_\sigma^-(v)$ that maximizes reward accumulation while preserving sufficient cost budget for future assignments. Then, for each edge e that appears in both q_t^{c} and q_t^{k}, OCKA accepts this edge (Section 3.3). We mainly focus on the algorithm design in this section, while the **underlying intuition** and **computational complexity analysis** are provided in our tech report (Wu et al. 2025).

3.1 Offline Preparation Phase

Before the first vertex arrives, OCKA starts in the Offline Preparation phase. We calculate y_e for each edge $e \in E$ (Line 3 of Alg. 1), which will be utilized in the Online Assignment phase to determine the probability of including edge e in the capacity-friendly set (Section 3.2).

We consider a relaxation of the problem **P** on the realization i, by removing the knapsack constraint (3) and the binary constraint (4), and denote it as $\mathbf{P}^*(\{r_i(e)\})$, where we use $r_i(e)$ to denote the realized edge reward of edge e in realization i:

$$\mathbf{P}^*(\{r_i(e)\}): \max_{x_e, \forall e \in E} \sum_{e \in E} x_e r_i(e) \tag{6}$$

Algorithm 2: Online Assignment Phase (`Assign`)

1: **Function** `Assign`$(t, v, E_\sigma^-(v), \{r(e)\}_{e\in E_\sigma^-(v)})$:
2: **if** $t = 1$ **then** Initialize $\bar{q}_0^{\mathrm{c}} \leftarrow \emptyset$.
3: $v \leftarrow \sigma(t)$, $q_t^{\mathrm{c}} \leftarrow \emptyset$.
4: **for** every edge $e \in E$ **do**
5: If $e \notin E_\sigma^-(v)$, sample $\hat{r}(e,t)$ by $\mathcal{R}(e)$.
6: If $e \in E_\sigma^-(v)$, $\hat{r}(e,t) \leftarrow r(e)$.
7: **end for**
8: Calculate $\mathbf{x}^*(\{\hat{r}(e,t)\}_{e\in E})$ as the optimal solution to the optimization problem in Eqs. (6)–(8).
9: $\hat{y}_\sigma^t(e) \leftarrow x_e^*(\{\hat{r}(e',t)\}_{e'\in E}), \forall e \in E(v)$.
10: $m \leftarrow 1$. $z_\sigma^t(e,m) \leftarrow \hat{y}_\sigma^t(e), \forall e \in E(v)$.
11: **while** $\exists e \in E(v)$ such that $z_\sigma^t(e,m) \neq 0$ **do**
12: $U_m^v \leftarrow \{e \in E(v) : z_\sigma^t(e,m) > 0\}$.
13: **if** $|U_m^v| > c_v$ **then**
14: $U_m^v \leftarrow \{e \in E(v) : z_\sigma^t(e,m)$ is one of the c_v largest values in $\{z_\sigma^t(e',m)\}_{e'\in E(v)}\}$.
15: **end if**
16: $\lambda_m^v \leftarrow \min_{e\in U_m^v} z_\sigma^t(e,m)$.
17: $z_\sigma^t(e,m+1) \leftarrow z_\sigma^t(e,m) - \lambda_m^v, \forall e \in U_m^v$.
18: $z_\sigma^t(e,m+1) \leftarrow z_\sigma^t(e,m), \forall e \in E(v) \setminus U_m^v$.
19: $m \leftarrow m + 1$.
20: **end while**
21: $M \leftarrow m$. $U_M^v \leftarrow \emptyset$ and $\lambda_M^v \leftarrow 1 - \sum_{m'=1}^{M-1} \lambda_{m'}^v$.
22: Sample a set U_t^* from $\{U_m^v\}_{m\in[M]}$ with probabilities $\{\lambda_m^v\}_{m\in[M]}$.
23: **for** every vertex u that $(u,v) \in U_t^* \cap E_\sigma^-(v)$ **do**
24: Calculate $\phi_u(v)$ by Eq. (10).
25: Include edge (u,v) in set q_t^c with probability $\phi_u(v)$.
26: **end for**
27: $\bar{q}_t^{\mathrm{c}} \leftarrow \bar{q}_{t-1}^{\mathrm{c}} \cup q_t^{\mathrm{c}}$.
28: Memorize $\bar{q}_t^{\mathrm{c}}$ for next call.
29: **Return:** q_t^c.

$$\text{s.t.} \sum_{e\in E(v)} x_e \le c_v, \forall v \in V, \quad (7)$$

$$x_e \in [0,1], \forall e \in E. \quad (8)$$

The optimal solution to $\mathbf{P}^*(\{r_i(e)\})$ is denoted as $\mathbf{x}^*(\{r_i(e)\}) = \{x_{e'}^*(\{r_i(e)\})\}_{e'\in E}$. The average preference y_e for each edge e is the expectation of $x^*(\{r_i(e')\})$ over the edge reward distributions $\{\mathcal{R}(e')\}$, such that

$$y_e := \mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{\forall e'\in E})\right], \forall e \in E. \quad (9)$$

We can use the Monte Carlo method (Dickerson et al. 2021; Ezra et al. 2020) to calculate each y_e by sampling the edge rewards, and we can solve the offline LP $\mathbf{P}^*(\{r_i(e)\})$ by standard approaches, such as the interior point method (Boyd and Vandenberghe 2004) or the simplex method (Bartels and Golub 1969).

3.2 Online Assignment Phase

When vertex $\sigma(t)$ arrives in each time slot t, OCKA first enters the Online Assignment phase (Lines 6–8 of Alg. 1) and calls the `Assign` function (Alg. 2) to determine the capacity-friendly set. For convenience of notation, we set $v = \sigma(t)$ (Line 7 of Alg. 1) and use v and $\sigma(t)$ interchangeably in Alg. 2 and the rest of this subsection. Intuitively, the capacity-friendly set q_t^{c} indicates which edges should be accepted if we are only restricted by the capacity constraint of each vertex.

To construct q_t^{c}, we implement a three-step probabilistic matching approach: (i) We determine the ***reference preference*** $\hat{y}_\sigma^t(e)$ by obtaining an optimal solution to the LP $\mathbf{P}^*$, which indicates how much we should prefer to accept or discard an edge $e \in E_\sigma^-(v)$, given its reward $r(e)$. (ii) We use the reference preferences to determine a ***Carathéodory set***, which includes each edge $e \in E_\sigma^-(v)$ with a probability of $\hat{y}_\sigma^t(e)$ while ensuring that the capacity constraint of vertex v is not violated. (iii) We calculate a ***seesaw probability*** $\phi_u(v)$ for each edge in the Carathéodory set to include it into the capacity-friendly set q_t^{c}, which is a crucial element of our probabilistic matching approach, dedicatedly designed to handle multi-capacity vertices.

Reference Preference The first step is to determine the ***reference preference*** $\hat{y}_\sigma^t(e)$ for each edge $e \in E_\sigma^-(v)$. We set a reference reward $\hat{r}(e,t)$ for each edge e by the following rules: If $e \notin E_\sigma^-(v)$, we sample $\hat{r}(e,t)$ from the distribution $\mathcal{R}(e)$; If $e \in E_\sigma^-(v)$, then $r(e)$ is already known and we set $\hat{r}(e,t) = r(e)$. We use the reference rewards $\hat{r}(e,t)$ to construct the LP $\mathbf{P}^*(\{\hat{r}(e,t)\})$ and obtain its optimal solution $\mathbf{x}^*(\{\hat{r}(e,t)\}_{e\in E})$ in Line 8 of Alg. 2. For each edge $e \in E_\sigma^-(v)$, we set the reference preference $\hat{y}_\sigma^t(e) = x_e^*(\{\hat{r}(e',t)\}_{e'\in E})$.

Carathéodory Set Next, we use the reference preference $\hat{y}_\sigma^t$ to obtain the Carathéodory set U_t^*, which is a subset of $E(v)$ of cardinality at most c_v (Lines 10–22 of Alg. 2). This is based on an implementation of the Carathéodory theorem (Leonard and Lewis 2015), which implies that there exists a convex decomposition of $\{\hat{y}_\sigma^t(e)\}$, i.e., $\hat{y}_\sigma^t(e) = \sum_{m=1}^{M} \lambda_m^v \mathbf{1}_{\{e\in U_m^v\}}$, where each U_m^v is a subset of $E(v)$ associated with a positive weight λ_m^v, $\mathbf{1}_{\{\cdot\}}$ is the indicator function, and $M \le |E(v)| + 1$. We then sample the Carathéodory set U_t^* from $\{U_m^v\}_{m\in[M]}$ with each U_m^v being picked with probability λ_m^v. By this design, each edge $e \in E(v)$ is included in the Carathéodory set U_t^* with probability equal to $\hat{y}_\sigma^t(e)$, and U_t^* contains no more than c_v edges.

Seesaw Probability Finally, we calculate a seesaw probability $\phi_u(\sigma(t))$ for each edge in the Carathéodory set in Line 24 of Alg. 2. For time slot t, we define the *union capacity-friendly set* $\bar{q}_{t-1}^{\mathrm{c}}$ as the union set of all capacity-friendly sets before time slot t. By the definition of $\bar{q}_{t-1}^{\mathrm{c}}$, it is calculated by $\bar{q}_{t-1}^{\mathrm{c}} = \bigcup_{\tau\in[t-1]} q_\tau^{\mathrm{c}}$. The union capacity-friendly set is initialized to $\bar{q}_0^{\mathrm{c}} = \emptyset$, and is updated to $\bar{q}_t^{\mathrm{c}}$ at the end of the `Assign` function in time slot t (Line 27 of Alg. 2). By the calculation of $\bar{q}_{t-1}^{\mathrm{c}}$, the union capacity-friendly set $\bar{q}_{t-1}^{\mathrm{c}}$ contains all edges that are included in each capacity-friendly set before time slot $t-1$. We further define the *reference remaining capacity* $n_u(\sigma(t))$ as the remaining capacity of vertex u when vertex $\sigma(t)$ arrives at t, in light of having accepted every edge in $\bar{q}_{t-1}^{\mathrm{c}}$, i.e., $n_u(\sigma(t)) = c_u - |\bar{q}_{t-1}^{\mathrm{c}} \cap E(u)|$. Finally, for each edge $(u,\sigma(t)) \in U_t^* \cap E(\sigma(t))$, we calculate the seesaw proba-

bility $\phi_u(\sigma(t))$ in Line 24 of Alg. 2 by the following rule:

$$\phi_u(\sigma(t)) = \frac{n_u(\sigma(t))}{c_u} \frac{1}{2 - \frac{1}{c_u}\sum_{\tau<t} y_{(u,\sigma(\tau))}}. \quad (10)$$

The seesaw probability is a critical component of our probabilistic matching approach, which is dedicatedly designed for the multiple vertex matching capacities and balances between consuming a capacity for instant reward and reserving a capacity for higher future reward.

We include every edge $(u, \sigma(t)) \in U_t^*$ with probability $\phi_u(\sigma(t))$ in the capacity-friendly set q_t^{c} (Line 25 of Alg. 2). We calculate $\bar{q}_t^{\mathrm{c}}$ by $\bar{q}_t^{\mathrm{c}} = \bar{q}_{t-1}^{\mathrm{c}} \cup q_t^{\mathrm{c}}$ (Line 27 of Alg. 2), where $\bar{q}_t^{\mathrm{c}}$ will be memorized in function `Assign` and q_t^{c} will be returned to OCKA (Lines 28–29 of Alg. 2). Finally, OCKA obtains q_t^{c} (Line 8 of Alg. 1) and enters the Assignment Acceptance phase (Line 9 of Alg. 1).

3.3 Assignment Acceptance Phase

In time slot t, after we obtain the capacity-friendly set $\bar{q}_t^{\mathrm{c}}$ and finish the Online Assignment phase, OCKA enters the Assignment Acceptance phase (Lines 9–11 of Alg. 1), where it calls the `Accept` function (Alg. 3) to construct the knapsack-friendly set q_t^{k} (Lines 2–28 of Alg. 3). We first explain the construction of the knapsack-friendly set and then conclude the decision of OCKA in time slot t.

The design of the Assignment Acceptance phase is inspired by (Jiang, Ma, and Zhang 2022). However, in OCGMK, the consumption of the cost budget is closely linked with the capacity usage of vertices. Directly applying the solution from (Jiang, Ma, and Zhang 2022) would disrupt the delicate dynamic between budget consumption and capacity usage. Therefore, we develop new probability rules for budget consumption when accepting an edge, ensuring a robust balance between budget consumption and capacity usage. These new probability rules also result in a different update mechanism for the distribution of budget consumption.

Constructing the Knapsack-friendly Set To construct the knapsack-friendly set q_t^{k} in time slot t, we implement a matching acceptance approach, where the `Accept` function evaluates each edge $(u, \sigma(t)) \in E_\sigma^-(\sigma(t))$ to determine whether to include it into q_t^{k}. We will introduce the ***reference knapsack consumption*** and the ***reference consumption distribution***, which together reflect how tight the current remaining budget consumption is. Then, we will introduce the critical design of the ***safe range***, which judges whether we should consume the remaining budget to accept edge e or to reserve the budget for the future.

Edge Iteration For each edge $e \in E_\sigma^-(\sigma(t))$ in time slot t, the `Accept` function will execute the loop between Lines 8–25 of Alg. 3 once to decide whether to include this edge in q_t^{k}. We call such a decision process for an edge e an edge iteration. We use $j = 1, 2, \ldots, |E|$ to index these edge iterations, and use e_j to denote the edge evaluated in edge iteration j. In the following discussion, we will focus only on a fixed edge iteration j that happens during some given time slot t, and we will omit the subscript t.

Algorithm 3: Assignment Acceptance Phase (`Accept`)

1: **Function** `Accept`$(t, v, E_\sigma^-(v), q_t^{\mathrm{c}})$:
2: **if** $t = 1$ **then**
3: Initialize $\beta \leftarrow \frac{\sum_e d(e)}{K}$, $\gamma = \frac{1}{3+e^{-2}}$.
4: Initialize $j \leftarrow 1$, $X_0 \leftarrow 0$, $\bar{q}_0^{\mathrm{k}} \leftarrow \emptyset$.
5: Initialize $\tilde{X}_0$ as a distribution with $\Pr\{\tilde{X}_0 = 0\} = 1$.
6: **end if**
7: $q_t^{\mathrm{k}} \leftarrow \emptyset$.
8: **for** each $e \in E_\sigma^-(\sigma(t))$ **do**
9: $e_j \leftarrow e$.
10: Denote $\{b_1, b_2, \ldots, b_L\}$ as the supports of distribution $\tilde{X}_{j-1}$, in an increasing order.
11: Calculate θ_j^+ and θ_j^- by Eq. (11).
12: Calculate ψ_j by Eq. (12).
13: **if** $X_{j-1} \in (\theta_j^-, \theta_j^+]$ **then**
14: Include e_j in q_t^{k} with a probability of $1/\beta$.
15: **else if** $X_{j-1} = \theta_j^-$ **then**
16: Include e_j in q_t^{k} with a probability of ψ_j.
17: **end if**
18: **if** $e_j \in q_t^{\mathrm{k}}$ **then**
19: $X_j \leftarrow X_{j-1} + d(e_j)$, $\bar{q}_j^{\mathrm{k}} \leftarrow \bar{q}_{j-1}^{\mathrm{k}} \cup e_j$.
20: **else**
21: $X_j \leftarrow X_{j-1}$, $\bar{q}_j^{\mathrm{k}} \leftarrow \bar{q}_{j-1}^{\mathrm{k}}$.
22: **end if**
23: Update $\tilde{X}_j$ from $\tilde{X}_{j-1}$.
24: $j \leftarrow j + 1$.
25: **end for**
26: $Q_t \leftarrow q_t^{\mathrm{c}} \cap q_t^{\mathrm{k}}$. For every edge $(u, v) \in Q_t$, assign u and v to each other.
27: Memorize j, X_{j-1}, $\tilde{X}_{j-1}$, and $\bar{q}_{j-1}^{\mathrm{k}}$ for next call.
28: **Return:** Q_t.

For edge iteration j, we define the *union knapsack-friendly set* $\bar{q}_{j-1}^{\mathrm{k}}$ as the set of edges that are included in all knapsack-friendly sets before the j-th edge iteration. We use *reference knapsack consumption* X_{j-1} to denote the total cost required by all edges contained in $\bar{q}_{j-1}^{\mathrm{k}}$, and use *reference consumption distribution* $\tilde{X}_{j-1}$ to denote the distribution of X_{j-1} over all sample paths and all algorithm run. The reference knapsack consumption and the reference consumption distribution together capture the long-term availability of the remaining knapsack budget, allowing us to balance consuming the budget for immediate rewards with reserving it for future vertices. At the beginning of the first call of the `Accept` function, we initialize $\bar{q}_0^{\mathrm{k}} = \emptyset$, $X_0 = 0$, and initialize $\tilde{X}_0$ with $\Pr\{\tilde{X}_0 = 0\} = 1$ (Lines 3–5 of Alg. 3). We calculate $\bar{q}_j^{\mathrm{k}}$, X_j, and $\tilde{X}_j$ at the end of the j-th edge iteration (Lines 18–23 of Alg. 3), which will be detailed below after we explain the Safe Range.

The Safe Range for Edge Iteration j During the j-th edge iteration, we denote the supports of the distribution $\tilde{X}_{j-1}$ by $\{b_1, b_2, \ldots, b_L\}$, where $b_1 < b_2 < \cdots < b_L$. the `Accept` function first evaluates the reference consumption distribution $\tilde{X}_{j-1}$ to determine a *safe range* of the reference

knapsack consumption X_{j-1}. Let θ_j^- be the left boundary of this safe range, and θ_j^+ be the right boundary. We calculate θ_j^- and θ_j^+ by

$$\theta_j^+ = b_{l_1} \text{and } \theta_j^- = b_{l_2}, \tag{11}$$

where $l_1 = \arg\max_{l\in[L]}\{l : \ b_l + d(e_j) \le K\}$ and $l_2 = \arg\min_{l\in[L]}\{\Pr\{\tilde{X}_{j-1} \in (b_l, K - d(e_j)]\} \le \gamma\}$.

The decision to include e_j in q_t^{k} accounts for the cumulative impact of all prior decisions and is randomized, depending on the value of X_{j-1}: (i) If $X_{j-1} \in (\theta_j^-, \theta_j^+]$, the `Accept` function will include e_j in q_t^{k} with probability $1/\beta$ (Lines 13–14 of Alg. 3). (ii) If X_{j-1} is equal to θ_j^-, the `Accept` function will include e_j in q_t^{k} with probability ψ_j (Lines 15–16 of Alg. 3). The probability ψ_j is calculated as

$$\psi_j = \frac{\gamma - \sum_{l=l_2+1}^{l_1} \Pr\{\tilde{X}_{j-1} = b_l\}}{\beta \times \Pr\{\tilde{X}_{j-1} = \theta_j^-\}}. \tag{12}$$

(iii) If X_{j-1} is outside the safe range $[\theta_j^-, \theta_j^+]$, the `Accept` function will not include e_j in q_t^{k}.

The safe range and the probability ψ_j reshape the probabilistic matching approach by deciding whether the current knapsack budget consumption is safe to accept an edge in the capacity-friendly set, so that our decisions align with the cost budget.

Updating Reference Consumption Distribution in Edge Iteration j Next, we update the reference knapsack consumption from X_{j-1} to X_j and update $\bar{q}_{j-1}^{\mathrm{k}}$ to $\bar{q}_j^{\mathrm{k}}$ (Lines 18–22 of Alg. 3). Updating X_j and $\bar{q}_j^{\mathrm{k}}$ is straightforward: If e_j is included in q_t^{k}, we set $X_j = X_{j-1} + d(e_j)$ and set $\bar{q}_j^{\mathrm{k}} = \bar{q}_{j-1}^{\mathrm{k}} \cup \{e_j\}$; otherwise we set $X_j = X_{j-1}$ and $\bar{q}_j^{\mathrm{k}} = \bar{q}_{j-1}^{\mathrm{k}}$.

More importantly, we also update the reference consumption distribution from $\tilde{X}_{j-1}$ to $\tilde{X}_j$, in Line 23 of Alg. 3, to reflect the change due to our safe-range based decision in edge iteration j. First, we make a copy of $\tilde{X}_{j-1}$, such that $\tilde{X}_j \leftarrow \tilde{X}_{j-1}$. Second, for the sample paths with $X_{j-1} \in [b_{l_2+1}, b_{l_1}]$, a cost of $d(e_j)$ is added to X_{j-1} with a probability of $1/\beta$, so we move a $1/\beta$ fraction of the probability mass from each $b_l \in [b_{l_2+1}, b_{l_1}]$ of distribution $\tilde{X}_{j-1}$ to $b_l + d(e_j)$. For the sample paths with $X_{j-1} = b_{l_2} = \theta_j^-$, a cost of $d(e_j)$ is added to X_{j-1} with a probability of ψ_j, so we move a ψ_j fraction of the probability mass from θ_j^- of the distribution $\tilde{X}_{j-1}$ to $\theta_j^- + d(e_j)$.

Then the j-th edge iteration is completed (Line 24 of Alg. 3). In time slot t, the `Accept` function will complete such an iteration for each edge $e \in E_\sigma^-(\sigma(t))$ to make a decision for each of these edges whether to include it in the knapsack-friendly set q_t^{k}.

Decisions in Time slot t**.** After the `Accept` function finishes the edge iterations above, it calculates the intersection $Q_t = q_t^{\mathrm{c}} \cap q_t^{\mathrm{k}}$ and accepts every edge $e \in Q_t$ (Line 26 of Alg. 3). The `Accept` function memorizes j, X_{j-1}, $\tilde{X}_{j-1}$, and $\bar{q}_{j-1}^{\mathrm{k}}$ for next call and returns Q_t to OCKA (Lines 27–28 of Alg. 3), then OCKA will wait for the next vertex to arrive. When every vertex has arrived, the union set $Q = \cup_{t\in[|V|]} Q_t$ contains all accepted edges. The Online Assignment and Assignment Acceptance phases together form our new OCRS framework. Through this new OCRS framework, OCKA strikes the right balance between vertex capacity usage and cost budget consumption, preventing the adversary from exhausting the vertex capacity before the cost budget is fully utilized, or vice versa.

4 Competitive Ratio Analysis

We summarize the main result of the competitive ratio achieved by OCKA on OCGMK in Theorems 1–2, and provide the detailed analysis in our tech report (Wu et al. 2025).

Theorem 1. *(OCKA Competitive Ratio) The competitive ratio of the OCKA algorithm on OCGMK is* $\alpha = \frac{\gamma}{2\beta}$*, where* $\gamma = \frac{1}{3+e^{-2}}$ *is a constant and* $\beta = \sum_{e\in E} d(e)/K$.

Theorem 1 characterizes the effectiveness of our new OCRS framework. Our algorithm also works when the knapsack constraint is not imposed, in which case we simply accept the edges in the capacity-friendly set q_t^{c} in time slot t. We give its competitive ratio for this case in Theorem 2.

Theorem 2. *(OCKA Competitive Ratio without Knapsack) When the knapsack constraint is not imposed, the competitive ratio that OCKA achieves on OCGMK is* $\alpha' = \frac{1}{2}$.

This competitive ratio recovers the previous best result achieved for the original single-capacity OGM (Ezra et al. 2020). In our tech report (Wu et al. 2025), we further provide numerical results to study the impact of various system parameters on the competitive ratio of OCKA.

5 Trace-Driven Evaluation

We further evaluate the performance of OCKA through trace-driven experiments on a real-world dating application dataset, in order to demonstrate its superior capability in real-world dating scenarios.

Dating Dataset We consider a real-world dating scenario based on the Hugging Face Matchmaking (HFM) dataset (dstam 2024), a curated version of the original SpeedDating (SD) dataset (Fisman et al. 2006). The SD dataset consists of 402 participants and their interaction data, which has been widely used to study dating and mating behaviours (Bjerk 2009; Finkel et al. 2012; Schwartz 2013), and later curated into the HFM dataset for model training.

In this dating scenario, users (vertices) arrive one by one and need to be paired with other eligible (edges) users. Pairing two users yields user satisfaction (edge reward) and incurs a legal risk for the service provider (edge cost). The accumulated risk incurred is budgeted as the risk appetite (Rittenberg and Martens 2012) of the service provider (edge cost budget). Each user's total number of allowed pairings (vertex capacity) is limited. The goal is to decide which users to pair, in order to maximize the overall user satisfaction within the risk appetite and users' allowed pairings. Before user arrivals, their satisfaction of being paired with others can be

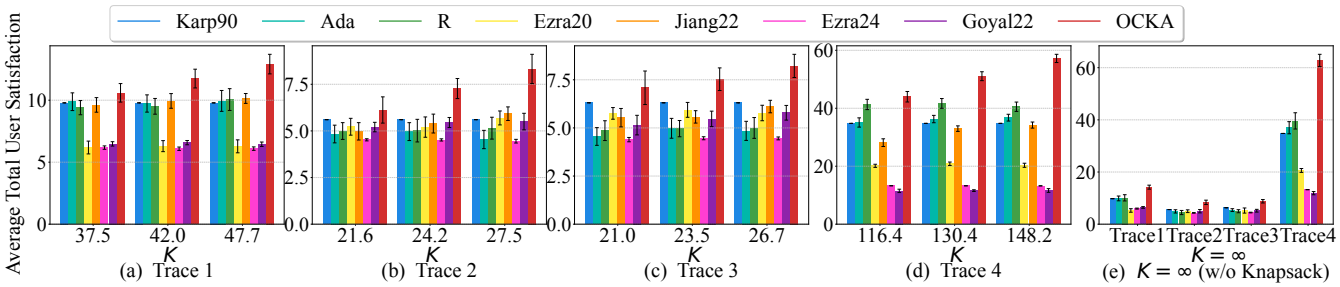

Figure 1: Performance comparison on traces with varying risk appetite (i.e., edge cost budget) K.

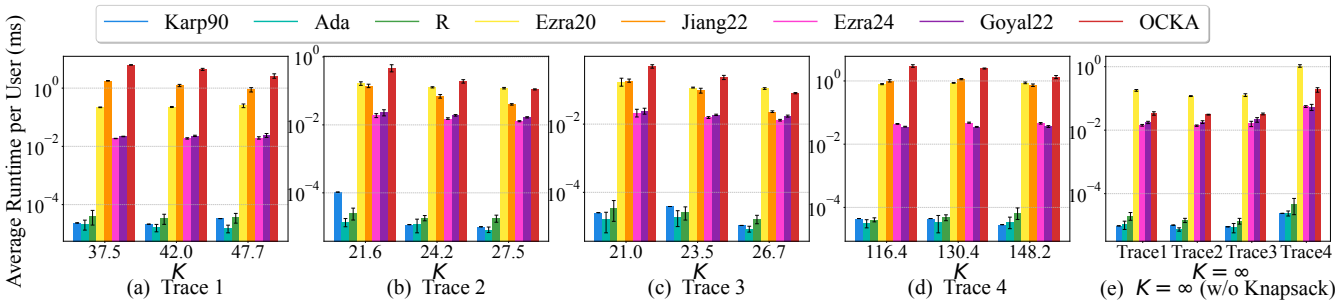

Figure 2: Runtime comparison on traces with varying risk appetite (i.e., edge cost budget) K.

estimated by a Multi-Layer Perceptron (MLP) as a probability distribution. Meanwhile, the perceived legal risk of pairing two users is estimated by another MLP but treated as the true value (Liberti and Petersen 2019). The legal risk and potential satisfaction can be inferred from user profiles, which are typically available in real-world online dating applications (Dellinger 2019). The exact satisfaction from pairing two users is obtained via ChatGPT-4 API (OpenAI 2023) inference once both have arrived.

Benchmarks and Experiment Results We compare the performance and runtime of OCKA against benchmarks, including Karp90 (Karp, Vazirani, and Vazirani 1990), Adaptive (Ada), Random (R), Ezra20 (Ezra et al. 2020), Jiang22 (Jiang, Ma, and Zhang 2022), Ezra24 (Ezra et al. 2024), and Goyal22 (Goyal 2022), all of which are adapted for the OCGMK. Please refer to our tech report (Wu et al. 2025) for details on the benchmarks.

We collect four traces from the HFM dataset and deploy OCKA along with benchmark algorithms on a MacBook Air with an M2 chip to process the user pairing requests contained in these traces. In Figs. 1–2, we compare the average accumulated user satisfaction (Fig. 1) and runtime (Figs. 2) of OCKA against benchmarks on each trace. In (a)–(d) of Fig. 1 and Fig. 2, we apply OCKA and benchmarks to each trace with different risk appetites. In Figs. 1(e) and 2(e), we also study the case where the knapsack constraint is not imposed (i.e., $K = \infty$). In most settings, OCKA substantially outperforms all benchmarks with minimal runtime overhead. We observe that the advantage of OCKA increases when the risk appetite increases. This is because when the knapsack budget is very limited, all algorithms face a highly constrained decision space, leaving little room for OCKA to gain a substantial reward advantage through optimizing decisions. In contrast, when the knapsack budget is large or unlimited, the decision space expands significantly, giving OCKA greater flexibility to make more effective decisions and thereby achieving more significant performance gains.

Further observations and the rationale behind OCKA's benefits over benchmarks are given in our tech report (Wu et al. 2025). To demonstrate OCKA's capability across broader settings, we also conduct simulation experiments using structural features extracted from the dataset (including graph topology, vertex capacities, edge costs, etc.), which are also discussed in our tech report (Wu et al. 2025).

6 Conclusion

In this paper, we study the OCGMK problem. We address the complicated trade-off between reward accumulation, vertex capacity utilization, and cost budget consumption, as a result of the multi-capacity vertices and the knapsack constraint. We propose the OCKA algorithm, a novel OCRS framework that decouples the joint evolution of capacity and budget usage through a two-phase design, achieving an effective balance between vertex capacity utilization and knapsack budget consumption. OCKA achieves a competitive ratio of $\alpha = \frac{\gamma}{2\beta}$ on OCGMK. When the knapsack constraint is not present, it achieves a competitive ratio of $\alpha' = 1/2$, which recovers the previous best result on the single-capacity OGM. We perform trace-driven experiments to demonstrate the excellent capability of the OCKA algorithm to accommodate multi-capacity vertices and the knapsack constraint in OCGMK.

References

Bartels, R. H.; and Golub, G. H. 1969. The simplex method of linear programming using LU decomposition. *Communications of the ACM*, 12(5): 266–268.

Bjerk, D. 2009. Beauty vs. earnings: Gender differences in earnings and priorities over spousal characteristics in a matching model. *Journal of Economic Behavior & Organization*, 69(3): 248–259.

Boyd, S.; and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Dean, B. C.; Goemans, M. X.; and Vondrák, J. 2008. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4): 945–964.

Dellinger, A. 2019. Tinder adds a 'Spring Break' mode for college spring flings. https://www.engadget.com/2019-02-26-tinder-spring-break-mode.html. Accessed: 2025-07-12.

Dickerson, J. P.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2021. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation*, 9(3): 1–17.

dstam. 2024. Matchmaking 1.0: An open-source starter dataset for training dating app and matchmaking recommendation models. https://huggingface.co/datasets/dstam/matchmaking. Accessed: 2025-07-18.

Ezra, T.; Feldman, M.; Gravin, N.; and Tang, Z. 2024. Tight bounds for secretary matching in general graphs. *Mathematics of Operations Research*.

Ezra, T.; Feldman, M.; Gravin, N.; and Tang, Z. G. 2020. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In *Proceedings of the ACM Conference on Economics and Computation*, 769–787.

Feldman, M.; Svensson, O.; and Zenklusen, R. 2021. Online contention resolution schemes with applications to Bayesian selection problems. *SIAM Journal on Computing*, 50(2): 255–300.

Finkel, E. J.; Eastwick, P. W.; Karney, B. R.; Reis, H. T.; and Sprecher, S. 2012. Online dating: A critical analysis from the perspective of psychological science. *Psychological Science in the Public Interest*, 13(1): 3–66.

Fisman, R.; Iyengar, S. S.; Kamenica, E.; and Simonson, I. 2006. Gender differences in mate selection: Evidence from a speed dating experiment. *The Quarterly Journal of Economics*, 121(2): 673–697.

Gamlath, B.; Kapralov, M.; Maggiori, A.; Svensson, O.; and Wajc, D. 2019. Online matching with general arrivals. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 26–37.

Goyal, M. 2022. Secretary matching with vertex arrivals and no rejections. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5051–5058.

Jiang, J.; Ma, W.; and Zhang, J. 2022. Tight guarantees for multi-unit prophet inequalities and online stochastic knapsack. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 1221–1246.

Karp, R. M.; Vazirani, U. V.; and Vazirani, V. V. 1990. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 352–358.

Leonard, I. E.; and Lewis, J. E. 2015. *Geometry of Convex Sets*. John Wiley & Sons.

Liberti, J. M.; and Petersen, M. A. 2019. Information: Hard and soft. *Review of Corporate Finance Studies*, 8(1): 1–41.

Lowalekar, M.; Varakantham, P.; and Jaillet, P. 2020. Competitive ratios for online multi-capacity ridesharing. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 771–779.

Ma, W.; MacRury, C.; and Nuti, P. 2024. Online matching and contention resolution for edge arrivals with vanishing probabilities. In *Proceedings of the ACM Conference on Economics and Computation*, 159.

Mehta, A.; et al. 2013. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4): 265–368.

OpenAI. 2023. GPT-4: An older high-intelligence GPT model. https://platform.openai.com/docs/models/gpt-4. Accessed: 2025-07-11.

Rittenberg, L.; and Martens, F. 2012. Enterprise risk management: Understanding and communicating risk appetite. https://www.coso.org/_files/ugd/3059fc_b0013c9344764b0b8c30a7eb7e5c27c9.pdf. Accessed: 2025-07-18.

Schwartz, C. R. 2013. Trends and variation in assortative mating: Causes and consequences. *Annual Review of Sociology*, 39(1): 451–470.

Sun, B.; Yang, L.; Hajiesmaili, M.; Wierman, A.; Lui, J. C.; Towsley, D.; and Tsang, D. H. 2022. The online knapsack problem with departures. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, volume 6, 1–32.

Wu, R.; Bao, W.; and Ge, L. 2023. Online task assignment with controllable processing time. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 5466–5474.

Wu, R.; Wei, B.; Ben, L.; and Wang, H. 2025. TechReport: Online Capacitated Matching with Knapsack. https://github.com/RuoyuWu6940/online-capacitated-matching-with-knapsack/blob/main/Online-Capacitated-Matching-with-Knapsack.pdf. Accessed: 2025-11-17.

Yang, S.; Yang, X.; McCann, J. A.; Zhang, T.; Liu, G.; and Liu, Z. 2013. Distributed networking in autonomic solar powered wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 31(12): 750–761.

Zhang, Z.; Li, Z.; and Wu, C. 2017. Optimal posted prices for online cloud resource allocation. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, volume 1, 1–26.

Zhou, Y.; Chakrabarty, D.; and Lukose, R. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *Proceedings of the International Conference on World Wide Web*, 1243–1244.