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), 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 Appendix A.

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 Appendix B 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 Appendix C 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 ALG(i) and the optimal overall reward obtained by the offline optimal algorithm OPT as 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 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).

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 Appendix D.

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:

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: $q_t^{\mathrm{c}} \leftarrow \texttt{Assign}(t, v, E_\sigma^-(v), \{r(e)\}_{e \in E_\sigma^-(v)})$
9: *# Assignment Acceptance Phase*
10: $Q_t \leftarrow \texttt{Accept}(t, v, E_\sigma^-(v), q_t^{\mathrm{c}})$
11: $Q \leftarrow Q \cup Q_t$, $t \leftarrow t + 1$.
12: **end for**

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

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

$$x_e \in [0, 1], \forall e \in E. \tag{8}$$

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

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

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 $M \leq |E(v)| + 1$. 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 $\bar{q}_0^{\mathrm{c}} = \emptyset$,

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.

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

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

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

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.

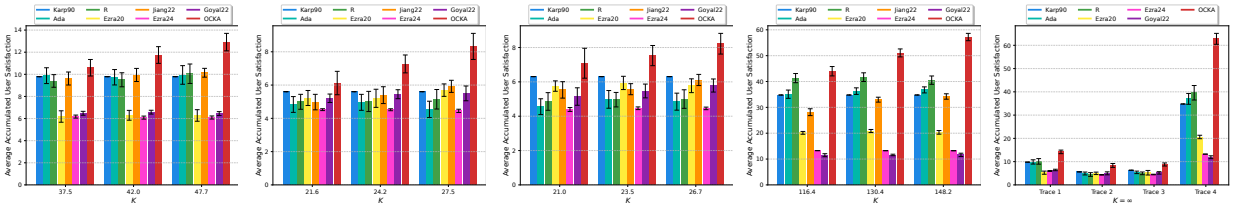
consumption and capacity usage. These new probability rules also result in a different update mechanism for the distribution of budget consumption.

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.

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^+



(a) Trace 1. (b) Trace 2. (c) Trace 3. (d) Trace 4. (e) $K = \infty$ (w/o knapsack).

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

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

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 $\theta_j^- + d(e_j)$.

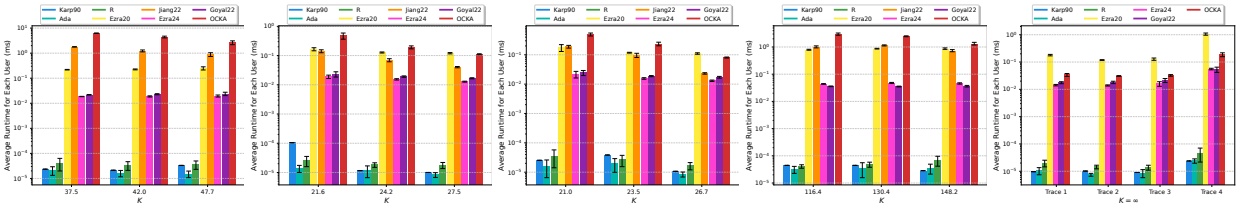
Then the j-th edge iteration is completed (Line 24 of Alg. 3). In time slot t, the `Accept` function will complete such an iteration for each edge $e \in E_\sigma^-(\sigma(t))$ to make a decision for each of these edges whether to include it in the knapsack-friendly set q_t^{k}.

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 Appendix E.

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 $\beta = \sum_{e \in E} d(e)/K$.*



(a) Trace 1. (b) Trace 2. (c) Trace 3. (d) Trace 4. (e) $K = \infty$ (w/o knapsack).

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

Theorem 1 characterizes the effectiveness of our new OCRS framework, which is designed to counteract the coupled effect of multi-capacity vertices and the knapsack constraint. 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 Constraint) When the knapsack constraint is not imposed, the competitive ratio that OCKA achieves on OCGMK is $\alpha' = \frac{1}{2}$.*

We note that this competitive ratio recovers the previous best result achieved for the original single-capacity OGM (Ezra et al. 2020). This improvement is enabled by considering the Carathéodory set in OCKA and by the novel design of the probability $\phi_u(\sigma(t))$ to balance between the remaining capacity and the expected capacity usage of each vertex. In Appendix F, 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 beyond the theoretical performance guarantees.

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 dataset (Fisman et al. 2006). The SpeedDating 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 restructured into the HFM dataset for training recommendation models. The HFM dataset contains 123 users and over 1000 interaction records, along with critical features (e.g., age, gender, religion, user biographies, and dating preferences) that significantly influence user decisions in dating contexts.

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 by pairing users 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 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. For example, dating platforms such as Tinder frequently launch time-limited promotional events to engage existing users (Dellinger 2019), where the platform needs to access relevant user information in advance. 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 (all with certain adaptation for OCGMK), 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 modified for the OCGMK. Please refer to Appendix G 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 and 2, we compare the average accumulated user satisfaction (Fig. 1) and runtime (Figs. 2) of OCKA against benchmarks on each trace. In Figs. 1a–1d and Figs. 2a–2d, we apply OCKA and benchmarks to each trace with different risk appetites. In Figs. 1e and 2e, 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 compared with all benchmarks are given in Appendix G. 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.). These results are also discussed in Appendix G.

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.

A Real-World Applications of OCGMK

OCGMK can be applied to model real-world problems in various areas. We provide some typical examples below:

User Pairing in Dating Application. In a social platform, the system receives pairing requests from users. Each user (vertex) wishes to be paired with other users with similar interests (edges). Each user has limited time to engage in dating activities, which limits the number of users they can be paired with (vertex capacity). Pairing users increases ad revenue (edge reward) by boosting user engagement, but it also incurs legal risks (edge cost) due to potential user misconduct. The legal risk is constrained by a fixed budget (cost budget). The platform must strategically decide which users to pair in order to maximize overall ad revenue. In Section 5 and Appendix G, we evaluate our algorithm with trace-driven experiments based on this scenario.

Device Connection in Wireless Sensor Networks (Yang et al. 2013). In a wireless sensor network, the system receives registration requests from wireless sensor devices. Each device (vertex) can be connected to other devices within a specific physical distance (edges) to analyze the nearby environment. Each device can connect to a limited number of other devices (vertex capacity). Connecting the devices enhances the accuracy of environment analysis (edge reward) but also increases resource consumption (edge cost), which is constrained by a budget (cost budget). The system must determine which devices to connect in order to maximize the overall accuracy of the environment analysis.

File Sharing in P2P Network (a special case in bipartite graphs). In a P2P network, the system receives file transfer requests from users. Each user (vertex) must connect with other users to either upload or download a file (bipartite graph). If a user wants to upload a file, it must connect to users who want to download it; if a user wants to download a file, it must connect to users who have the file (edges). Due to limited network bandwidth (vertex capacity), each user can only connect to a limited number of other users. Connecting users increases the overall throughput of the P2P network (edge reward), but also increases the long-term average power consumption (edge cost), which is constrained (cost budget). The P2P network must strategically decide which users to connect to maximize the overall network throughput.

B Related Work

In this section, we provide a detailed discussion on the existing works related to the OCGMK, which was briefly summarized in Section 1 of the main paper. The OCGMK problem is connected to the topics of online matching, the online stochastic knapsack problem, and OCRS. However, none of the existing work within these frameworks addresses the OCGMK problem.

B.1 Online Prophet Matching

One category of work related to this paper is the online prophet matching problem, a major branch of the broader Online Matching Problem(OMP). OMP was first studied by Karp, Vazirani, and Vazirani (1990). They considered online bipartite matching (OBM), where there is one set of offline vertices and one set of online vertices, with the online vertices arriving in an unknown order. Then Mehta et al. (2013) introduced online stochastic matching to OBM, where the offline vertices accept online matches with a certain probability.

Under the prophet matching branch, the OBM was extended to consider that edge rewards are drawn from known distributions and the vertex arrival order is unknown (Feldman, Gravin, and Lucier 2015). Within this branch, further research has extended OBM to settings like the known identical independent distribution (KIID) (Brubach et al. 2016) and known adversarial distribution (KAD) (Alaei, Hajiaghayi, and Liaghat 2012; Lowalekar, Varakantham, and Jaillet 2020). Recent works within the prophet matching branch of OBM study reusable offline vertices (Dickerson et al. 2021), offline vertices that can reject online matches (Andani, Puello, and Geurs 2021; Sumita et al. 2022; Xu 2024), capacitated vertices (Lowalekar, Varakantham, and Jaillet 2020; Wang, Yan, and Bei 2022; Naor, Srinivasan, and Wajc 2025), and edges to be probed (Borodin and MacRury 2025). However, these works only consider bipartite graphs and only one set of vertices arrives online.

Wang and Wong (2015) and Gamlath et al. (2019) investigated OBM with two-sided arrivals, where both sets of vertices in the bipartite graph arrive in an online manner. However, since these works still focus on the bipartite graph, the matching of vertices is limited to being across the two different groups of vertices. Moreover, these studies only considered an unweighted graph, i.e., the reward of matching any pair of vertices is identical, thereby they do not fall under the prophet matching branch.

The online general matching (OGM) problem was studied by (Ezra et al. 2020; Ma, MacRury, and Nuti 2024) under the prophet matching branch, where the graph is not necessarily bipartite, the edges are weighted, and every vertex arrives in an online manner. Ezra et al. (2020) studied OGM with stochastic edge weights, and Ma, MacRury, and Nuti (2024) studied OGM in an edge-arrival setting with vanishing probabilities. However, they only considered single-capacity vertices. To the best of our knowledge, we are the first to consider multi-capacity vertices in the OGM and to account for a possible knapsack cost constraint.

B.2 Other Online Matching Methods

Other important branches of the OMP include secretary matching and prophet secretary matching. The secretary matching branch considers the unknown edge reward with random vertex arrival in the OBM (Goyal 2022; Ezra et al. 2024). The prophet

secretary matching considers the prophet matching with vertices arriving in uniform random order (Chen et al. 2025). However, to the best of our knowledge, none of the existing works in any branch of the OMP have considered OGM with multi-capacity vertices or a knapsack constraint.

B.3 Online Knapsack Problem

Another category of work related to this paper is the online knapsack problem (OKP). In OKP (Zhou, Chakrabarty, and Lukose 2008; Zhang, Li, and Wu 2017), the decision maker selects from a sequence of items to maximize a total reward without exceeding the cost budget. The reward and cost of each item are revealed to the decision-maker when the item arrives. Based on OKP, a wide variety of generalizations have been studied: Zhou, Chakrabarty, and Lukose (2008) and Sun et al. (2020) considered the multiple-knapsack variant; Zhang, Li, and Wu (2017) and Yang et al. (2021) considered the multiple-resource variant; and Zhou, Chakrabarty, and Lukose (2008) and Sun et al. (2022) considered the departure of items. A stochastic variant of OKP is the Online Stochastic Knapsack Problem (OSKP), where either task costs or rewards are stochastic. In the deterministic cost setting (Jiang, Ma, and Zhang 2022), costs are known by the decision maker at the beginning, while in the stochastic cost setting (Papastavrou, Rajagopalan, and Kleywegt 1996; Dean, Goemans, and Vondrák 2008), the cost of each task follows a known distribution, with its exact value revealed upon task arrival. Recently, OSKP has been studied in the context of OCRS by (Jiang, Ma, and Zhang 2022), achieving a tight competitive ratio of 0.319. However, none of these studies addressed the knapsack problem in the context of OGM. Furthermore, 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.

B.4 Online Contention Resolution Scheme

The solution that we propose to tackle the OCGMK problem falls within the framework of OCRS. It is the online version of the contention resolution scheme (CRS), which is a rounding framework for solving submodular function optimization problems (Chekuri, Vondrák, and Zenklusen 2011). OCRS was first proposed by (Feldman, Svensson, and Zenklusen 2016) to solve Bayesian and stochastic online optimization problems. Since then, OCRS has been used to solve many online optimization problems, including Bayesian selection problems (Feldman, Svensson, and Zenklusen 2021), OGM (Ezra et al. 2020), and OSKP (Jiang, Ma, and Zhang 2022). Recent work has also studied various extensions of OCRS, including multidimensional OCRS (Chawla et al. 2025), batched OCRS (Ezra et al. 2020), and oblivious OCRS (Fu et al. 2022). However, none of these frameworks can simultaneously address the multi-capacity constraint and the knapsack constraint in OCGMK. In our work, we develop a new framework to integrate two different contention resolution components to tackle the combined challenge of the online matching problem and the online knapsack problem, which forms a main step of our OCKA algorithm.

C Summary of Notations

In this section, we summarize the notations introduced in Section 2 of the main paper in Table 1.

D Algorithm Design

In this section, we provide additional details and intuition for OCKA.

D.1 Offline Preparation Phase

For the offline preparation phase, we use the Monte Carlo method to calculate each y_e. Note that since solving the optimization problem $\mathbf{P}^*(\{r_i(e)\})$ does not require knowledge of the arriving order σ of vertices or the edge costs $\{d(e)\}$, we can calculate y_e by only sampling the edge rewards. We sample the edge rewards for N realizations independently from the reward distributions $\{\mathcal{R}(e)\}_{e\in E}$. We denote the edge rewards of realization n as $\{\bar{r}_n(e)\}_{\forall e\in E}$ for each $n \in [N]$.[1] For each realization n, we solve the offline LP $\mathbf{P}^*(\{\bar{r}_n(e)\})$ to obtain the optimal solution $\mathbf{x}^*(\{\bar{r}_n(e)\})$ by standard approaches, such as the interior point method (Boyd and Vandenberghe 2004) or the simplex method (Bartels and Golub 1969). Then the average preference y_e for edge e is set by $y_e \leftarrow \frac{1}{N}\sum_{n=1}^{N} x_e^*(\{\bar{r}_n(e')\}_{\forall e'\in E}), \forall e \in E$.

The Monte Carlo simulation is commonly used in other online matching problems (Dickerson et al. 2021; Ezra et al. 2020). We use ϵ_{Mon} to denote the sampling error of the Monte Carlo method. Then the number of simulation rounds N required to achieve this sampling error is bounded by $\mathcal{O}(\frac{1}{\epsilon_{\text{Mon}}})$, and this sampling error can be folded into a multiplicative factor of $(1-\epsilon_{\text{Mon}})$ in the competitive ratio by standard Chernoff bounds (Dickerson et al. 2021; Ezra et al. 2020). Therefore, for cleaner presentation, in the later performance analyses, we will simply assume that we know the exact value of y_e defined in Eq. (9).

D.2 Online Assignment Phase

The reference preference, intuitively, evaluates the reward $r(e)$ of the edges in $E_\sigma^-(v)$ against the potential reward of other edges over the whole problem instance, so if we can accept each edge with a probability equal to $\hat{y}_\sigma^t(e)$, the expected overall

[1] We denote the integer set $\{1, 2 \ldots, N\}$ by $[N]$.

Symbols	Explanation
$G = (V, E)$	The graph with vertex set V and edge set E.
e	An edge.
(u, v)	An edge with vertices u and v being its endpoints.
$N(v)$	The set of neighbors of vertex v.
$E(v)$	The set of edges incident to vertex v.
c_v	The capacity of vertex v.
$r(e)$	The reward of edge e.
$\mathcal{R}(e)$	The discrete and independent distribution of the reward of edge e.
$d(e)$	The cost of edge e.
K	The cost budget of the decision maker.
$\hat{E}$	The set of edges selected (accepted) by the decision maker.
t	A time slot.
σ	The mapping from time slots to vertices. $\sigma(t)$ is the vertex arriving in time slot t.
σ^{-1}	The inverse function of σ. $\sigma^{-1}(v)$ is the time slot that vertex v arrives in.
$N_\sigma^-(v)$	The neighbors of vertex v that arrive before v.
$N_\sigma^+(v)$	The neighbors of vertex v that arrive after v.
$E_\sigma^-(v)$	The edges incident to vertex v and the neighbors of v that arrive before v.
$E_\sigma^+(v)$	The edges incident to vertex v and the neighbors of v that arrive after v.
x_e	The decision of whether the decision maker selects (accepts) edge e into $\hat{E}$ or discards edge e. $x_e = 0$ if the decision maker discards e. $x_e = 1$ if the decision maker accepts edge e.
I	A problem instance, which is specified by G, $\{\mathcal{R}(e)\}$, $\{d(e)\}$, $\{c_v\}$, K, and σ.
i	A realization i of a problem instance I has the edge rewards $\{r(e)\}$ realized from their distributions $\{\mathcal{R}(e)\}$ specified by I.
ALG(i)	The overall reward achieved by algorithm ALG on the realization i.
OPT(i)	The overall reward achieved by OPT on the realization i.
α	The competitive ratio.

Table 1: Commonly used symbols.

reward could be guaranteed (see Theorem 3 in Appendix E). However, accepting each edge independently with $\hat{y}_\sigma^t(e)$ may violate the capacity constraint of vertex v. To guarantee the expected reward without violating the capacity constraint for vertex v, we need to obtain the Carathéodory set.

The Carathéodory set U_t^* is obtained as follows. For each edge $e \in E(v)$, we initialize $z_\sigma^t(e, 1)$ as $z_\sigma^t(e, 1) = \hat{y}_\sigma^t(e)$. In the m-th iteration of the loop in Lines 11–20 of Alg. 2, we first construct the base set U_m^v: If the number of positive $z_\sigma^t(e, m)$ is not larger than c_v, U_m^v includes all edges e such that $z_\sigma^t(e, m) > 0$ (Line 12 of Alg. 2); If the number of positive $z_\sigma^t(e, m)$ is larger than c_v, U_m^v includes every edge e such that $z_\sigma^t(e, m)$ is one of the c_v largest values in $\{z_\sigma^t(e', m)\}_{e' \in E(v)}$, with ties broken uniformly (Lines 13–15 of Alg. 2). Then, we initialize the weight λ_m^v to the smallest $z_\sigma^t(e, m)$ among all edges $e \in U_m^v$ (Line 16 of Alg. 2). Finally, we calculate $z_\sigma^t(e, m+1)$ from $z_\sigma^t(e, m)$ in Lines 17–18 of Alg. 2. The loop in Lines 11–20 of Alg. 2 ends when we have $z_\sigma^t(e, m) = 0$ for all edges $e \in U_m^v$. Let M be the value of m when this loop finishes. Then, we set U_M^v as an empty set, and set $\lambda_M^v = 1 - \sum_{m'=1}^{M-1} \lambda_{m'}^v$ (Line 21 of Alg. 2). Finally, we sample the Carathéodory set U_t^* from base sets, with each U_m^v being picked with probability λ_m^v (Line 22 of Alg. 2). 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, which will be discussed in detail in Lemma 1 in Appendix E.

For **the seesaw probability** defined in Eq. (10), the denominator $2 - \frac{1}{c_u} \sum_{\tau < t} y_{(u, \sigma(\tau))}$ captures the significance of the future reward for vertex u. Note that here we use the average preference y_e calculated in the Offline Preparation phase. Thus, $\phi_u(\sigma(t))$ indicates how much we should prioritize consuming u's capacity for an immediate reward, considering both the remaining capacity of u and the potential for its future rewards. The seesaw probability $\phi_u(\sigma(t))$ constructed in Line 24 of Alg. 2 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. It efficiently manages the vertex capacity in the presence of the extra decision dimension introduced by the multi-capacity vertices. This reasoning will be elaborated in the discussion of Theorem 3 in Appendix E, and we will show that the seesaw probability $\phi_u(\sigma(t))$ is a valid probability in Lemma 3 in Appendix E.

D.3 Assignment Acceptance Phase

Motivation of the knapsack-friendly set $q_t^{\mathbf{k}}$ and the assignment acceptance phase. The knapsack-friendly set $q_t^{\mathbf{k}}$ is constructed through a matching acceptance approach, which efficiently manages the consumption of the knapsack budget. The Online Assignment and Assignment Acceptance phases together form our new OCRS framework. They are designed to effectively counteract the coupled effect of multi-capacity vertices and the knapsack constraint. 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.

Detailed steps to update the reference knapsack consumption. The reference knapsack consumption X_{j-1} and the reference consumption distribution $\tilde{X}_{j-1}$ are updated to X_j and $\tilde{X}_j$ at the end of the j-th edge iteration (Lines 18–23 of Alg. 3): 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} and e_j is then accepted with a probability of $1/\beta$. As a result, 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 update. Specifically, in the second step, we perform the following operations for each $l \in [l_2+1, l_1]$:

$$\begin{cases} \Pr\{\tilde{X}_j = b_l\} \leftarrow \Pr\{\tilde{X}_j = b_l\} - (1/\beta)\Pr\{\tilde{X}_{j-1} = b_l\}, \\ \Pr\{\tilde{X}_j = b_l + d(e_j)\} \leftarrow \Pr\{\tilde{X}_j = b_l + d(e_j)\} + (1/\beta)\Pr\{\tilde{X}_{j-1} = b_l\}. \end{cases} \tag{13}$$

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. As a result, we move a ψ_j fraction of the probability mass from θ_j^- of the distribution $\tilde{X}_{j-1}$ to $\theta_j^- + d(e_j)$ for the update. Specifically, we perform the following operation in the third step:

$$\begin{cases} \Pr\{\tilde{X}_j = \theta_j^-\} \leftarrow \Pr\{\tilde{X}_j = \theta_j^-\} - \psi_j \Pr\{\tilde{X}_{j-1} = \theta_j^-\}, \\ \Pr\{\tilde{X}_j = \theta_j^- + d(e_j)\} \leftarrow \Pr\{\tilde{X}_j = \theta_j^- + d(e_j)\} + \psi_j \Pr\{\tilde{X}_{j-1} = \theta_j^-\}. \end{cases} \tag{14}$$

D.4 OCKA Algorithm Complexity Analysis

Before the first vertex arrives, we calculate the average preference y_e for each edge by N rounds of Monte Carlo simulation. The complexity for solving the LP $\mathbf{P}^*(\{r_n(e)\})$ can be bounded by the complexity of the interior point method, which is $O(|E|^{3.5}\log(\frac{1}{\epsilon_{\text{New}}}))$, with ϵ_{New} denoting the accuracy of each Newton step (Boyd and Vandenberghe 2004; Nesterov and Nemirovski 1994).[2] For a desired sampling error ϵ_{Mon} for the Monte Carlo method, the required simulation rounds N is upper bounded by $\mathcal{O}(\frac{1}{\epsilon_{\text{Mon}}})$. As a result, the overall complexity of the Offline Preparation phase with N rounds of Monte Carlo simulations is $O(\frac{1}{\epsilon_{\text{Mon}}}|E|^{3.5}\log(\frac{1}{\epsilon_{\text{New}}}))$.

For the Online Assignment phase in each time slot, we solve the LP $\mathbf{P}^*(\{\hat{r}(e,t)\})$ in Line 8 of Alg. 2 by the interior point method with a complexity of $O(|E|^{3.5}\log(\frac{1}{\epsilon_{\text{New}}}))$. In Lines 11–21 of Alg. 2, we find each λ_m^v and U_m^v. By the Carathéodory theorem, this loop terminates after no more than $|E(\sigma(t))|+1$ iterations, leading to a complexity of $O(|E|)$. In Lines 23–26 of Alg. 2, we calculate the seesaw probability $\phi_u(\sigma(t))$ for at most $|E|$ edges. The computation of $\phi_u(\sigma(t))$ for one edge $(u, \sigma(t))$ requires at most $|V|$ summation steps, leading to a complexity of $O(|V||E|)$ to compute all $\phi_u(\sigma(t))$. Consequently, the overall complexity of the Online Assignment phase in each time slot is $O(|E|^{3.5}\log(\frac{1}{\epsilon_{\text{New}}}))$, dominated by solving the LP in Line 8 of Alg. 2.

For the Assignment Acceptance phase in each time slot, we first calculate the computational complexity required by each edge iteration. In Line 11 of Alg. 3, we calculate l_1 and θ_j^+ with at most L steps, and then we calculate l_2 and θ_j^- with at most L steps, resulting in a complexity of $O(L)$. In Line 12 of Alg. 3, we calculate ψ_j by Eq. (12) with a complexity of $O(L)$. In Line 23 of Alg. 3, we update the distribution $\tilde{X}_j$ by Eqs. (13)–(14) with at most $2L$ summation steps, which leads to a complexity of $O(L)$. As a result, the computational complexity for each edge iteration is $O(L)$, where L is the number of possible values for the distribution $\tilde{X}_j$. In each time slot, OCKA iterates over at most $|E|$ edges during the Assignment Acceptance phase, resulting in a complexity of $O(|E|L)$. In practice, we can normalize the edge cost with respect to the total budget K. Then, L is bounded by the number of decimal places used to represent each normalized edge cost $d(e)$. Let dec be the number of decimal places used to represent each $d(e)$. Then the granularity of representing each $d(e)$ is $\text{gra} = 10^{-\text{dec}}$, and L is upper bounded by $\frac{1}{\text{gra}}$ so that the computational complexity of the Assignment Acceptance phase is $O(|E|\frac{1}{\text{gra}})$.

In conclusion, the computational complexity of OCKA in each time slot is $O(|E|^{3.5}\log(\frac{1}{\epsilon_{\text{New}}}) + |E|\frac{1}{\text{gra}})$. We note that the total complexity of our algorithm in each time slot is polynomial with respect to the number of edges and pseudo-polynomial with respect to the cost granularity. In real-world applications, cost values typically have coarse granularity. For example, monetary costs are typically measured in whole dollars, so that the number of possible costs is not large. In Appendix F, we also conduct numerical experiments to compare the average runtime of OCKA with several benchmark algorithms, demonstrating that it has an acceptable runtime, especially considering its substantial performance benefits.

[2] We follow the tradition in (Boyd and Vandenberghe 2004) to bound the complexity of solving the LP by the interior point method.

E Competitive Ratio Analysis

In this section, we analyze the competitive ratio of the OCKA algorithm. The road map of our analysis is shown in Figure 3. We will first analyze the probability that an edge is in the capacity-friendly set q_t^{c} in Appendix E.1 (Lemmas 1–3 and Theorem 3), and then analyze the probability that an edge is in the knapsack-friendly set q_t^{k} in Appendix E.2 (Lemmas 4–5 and Theorem 4). Then, we will give the competitive ratios of OCKA in Appendix E.3 (Theorems 5–6).

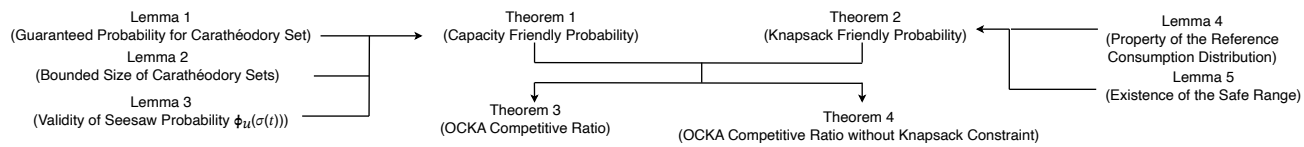


Figure 3: Road map of proofs.

E.1 Analysis of the Capacity-Friendly Set

The capacity-friendly set q_t^{c} is constructed in the Online Assignment phase. Recall that in this phase, OCKA first sets the reference reward $\hat{r}(e,t)$ for each edge and calculates the optimal solution $\mathbf{x}^*(\{\hat{r}(e,t)\}_{e\in E})$ to the LP $\mathbf{P}^*(\{\hat{r}(e,t)\}_{e\in E})$. Then OCKA samples the Carathéodory set U_t^* and includes each edge $(u,\sigma(t)) \in U_t^*$ into q_t^{c} with probability $\phi_u(\sigma(t))$. We design our Carathéodory set based on the Carathéodory Theorem (Leonard and Lewis 2015), so we will first provide a formal description of Carathéodory Theorem. Then, we (i) give the probability that an edge is in the Carathéodory set U_t^* in Lemma 1, (ii) provide an upper bound on the number of edges in U_t^* in Lemma 2, (iii) show that every $\phi_u(\sigma(t))$ is a valid probability in Lemma 3, and finally (iv) give the probability that an edge e is included in q_t^{c} in Theorem 3.

Carathéodory Theorem (Theorem 3.3.10 in (Leonard and Lewis 2015)) Let C be a subset of $\mathbb{R}^n$, $\mathrm{conv}(C)$ be the Convex Hull of C, and $x \in \mathrm{conv}(C)$, then there exists a set of at most $n+1$ points $\{x_1, x_2, \ldots, x_{n+1}\} \subset C$ such that

$$x = \lambda_1 x_1 + \lambda_2 + x_2 + \cdots + \lambda_{n+1} x_{n+1},$$

where $0 \le \lambda_i \le 1$ for $i = 1, 2, \ldots, n+1$ and $\lambda_1 + \lambda_2 + \cdots + \lambda_{n+1} = 1$.

Lemma 1. *(Guaranteed Probability for Carathéodory Sets) For an arbitrary t, let $e \in E_\sigma^-(\sigma(t))$ be an edge whose reward is revealed in time slot t and $\tilde{r} \in$ supp$(\mathcal{R}(e))$ be a possible reward of e, we have $\Pr_{i\sim I}\{e \in U_t^* \mid r_i(e) = \tilde{r}\} = \mathbb{E}_{i\sim I}[x_e^*(\{r_i(e')\}_{e'\in E}) \mid r_i(e) = \tilde{r}]$ for every $t \in [|V|]$, $e \in E_\sigma^-(\sigma(t))$, and $\tilde{r}$.*

Proof. **Overview:** To prove this lema, we find that over all sample paths and all algorithm runs, for each edge $e \in E_\sigma^-(\sigma(t))$, the average value of $\hat{y}_\sigma^t(e)$ is the same as the average value of $x_e^*(\{r(e')\}_{e'\in E})$. Then, by the Carathéodory theorem, we show that the probability that the average value of $\hat{y}_\sigma^t(e)$ is the same as the probability that this edge is included in the Carathéodory set U_t^*, which will complete the proof.

Detailed Proof: Let supp$(\mathcal{R}(e))$ denote the support of distribution $\mathcal{R}(e)$, given by

$$\mathrm{supp}(\mathcal{R}(e)) = \{r \in \mathbb{R} \mid \Pr\{r(e) = r\} > 0\}, \forall e \in E. \tag{15}$$

Since U_t^* is sampled from $\{U_m^v\}_{m\in[M]}$ and $\{U_m^v\}_{m\in[M]}$ is constructed from $\{\hat{y}_\sigma^t(e)\}$, we first analyze $\{\hat{y}_\sigma^t(e)\}$. In particular, we first prove that for all $t \in [|V|]$ and $e \in E_\sigma^-(\sigma(t))$, we have $\mathbb{E}[\hat{y}_\sigma^t(e) \mid r(e) = \tilde{r}] = \mathbb{E}[x_e^*(\{r(e')\}_{e'\in E}) \mid r(e) = \tilde{r}]$ over all realizations. By the construction of each $\hat{y}_\sigma^t(e)$, we have $\hat{y}_\sigma^t(e) = x_e^*(\{\hat{r}(e',t)\}_{e'\in E})$. As a result, we have

$$\mathbb{E}\left[\hat{y}_\sigma^t(e) \mid r(e) = \tilde{r}\right] = \mathbb{E}\left[x_e^*(\{\hat{r}(e',t)\}_{e'\in E}) \mid r(e) = \tilde{r}\right], \ \forall \tilde{r} \in \mathrm{supp}(\mathcal{R}(e)). \tag{16}$$

By the construction of each $\hat{r}(e,t)$, the sets $\{\hat{r}(e',t)\}_{e'\in E}$ and $\{r(e')\}_{e'\in E}$ follow the same distribution over the realizations where the reward of any edge $e \in E_\sigma^-(\sigma(t))$ is realized as $r(e) = \tilde{r}$. As a result, we have

$$\mathbb{E}[x_e^*(\{r(e')\}_{e'\in E}) \mid r(e) = \tilde{r}] = \mathbb{E}[x_e^*(\{\hat{r}(e',t)\}_{e'\in E}) \mid r(e) = \tilde{r}] \tag{17}$$

$$= \mathbb{E}\left[\hat{y}_\sigma^t(e) \mid r(e) = \tilde{r}\right], \tag{18}$$

for all $t \in [|V|]$, $e \in E_\sigma^-(\sigma(t))$, and $\tilde{r} \in \mathrm{supp}(\mathcal{R}(e))$. Next, we show that, for every t and $e \in E(\sigma(t))$, the probability that an edge e is included in set U_t^* equals $\hat{y}_\sigma^t(e)$ on each fixed realization. We first define a polytope $\mathcal{P}_{\sigma(t)}$ by

$$\mathcal{P}_{\sigma(t)} = \left\{ \mathbf{x} \in [0,1]^{E(\sigma(t))} \mid \sum_{e\in E(\sigma(t))} x_e \le c_{\sigma(t)} \right\}, \tag{19}$$

with the set of vertices in $\mathcal{P}_{\sigma(t)}$ denoted by

$$\mathcal{P}'_{\sigma(t)} = \left\{ \mathbf{x} \in \{0,1\}^{E(\sigma(t))} \mid \sum_{e \in E(\sigma(t))} x_e \leq c_{\sigma(t)} \right\}. \tag{20}$$

Since $\mathbf{x}^*(\{\hat{r}(e',t)\}_{e' \in E})$ is the optimal solution to the optimization problem $\mathrm{P}^*(\{\hat{r}(e',t)\}_{e' \in E})$, we have $\sum_{e \in E(\sigma(t))} x_e^*(\{\hat{r}(e',t)\}_{e' \in E}) \leq c_\sigma$. Since we have $\hat{y}_\sigma^t(e) = x_e^*(\{\hat{r}(e',t)\}_{e' \in E})$ for every e, we then have $\sum_{e \in E(\sigma(t))} \hat{y}_\sigma^t(e) \leq c_{\sigma(t)}$. As a result, the set of reference preference $\{\hat{y}_\sigma^t(e)\}_{e \in E(\sigma(t))}$ is inside the polytope $\mathcal{P}_{\sigma(t)}$.[3] Furthermore, define $\mathcal{U}_{\sigma(t)} = \{U \subseteq E(\sigma(t)) \mid |U| \leq c_{\sigma(t)}\}$ and let $\mathbf{1}_U$ be the indicator vector of set U. Equivalently, $\mathcal{P}'_{\sigma(t)}$ can also be expressed as

$$\mathcal{P}'_{\sigma(t)} = \{\mathbf{1}_U \mid U \in \mathcal{U}_{\sigma(t)}\}. \tag{21}$$

By the Carathéodory theorem (Leonard and Lewis 2015), $\{\hat{y}_\sigma^t(e)\}_{e \in E(\sigma(t))}$ can be decomposed as a convex combination of at most $|E(\sigma(t))| + 1$ sets in $\mathcal{P}'_{\sigma(t)}$. In particular, there exist $U_m \in \mathcal{U}_{\sigma(t)}$ and λ_m ($m = 1, \ldots, |E(\sigma(t))| + 1$) such that

$$\{\hat{y}_\sigma^t(e)\}_{e \in E(\sigma(t))} = \sum_{m=1}^{|E(\sigma(t))|+1} \lambda_m \mathbf{1}_{U_m}, \tag{22}$$

which can also be written as

$$\hat{y}_\sigma^t(e) = \sum_{m=1}^{|E(\sigma(t))|+1} \lambda_m \mathbf{1}_{\{e \in U_m\}}, \forall e \in E(\sigma(t)), \tag{23}$$

where $\sum_m \lambda_m = 1$ and $\lambda_m \in [0,1]$ for every $m \in [|E(\sigma(t))| + 1]$. In Alg. 2, we compute $\{\lambda_m^v\}$ and $\{U_m^v\}$ in Lines 11–21 of Alg. 2, which is a standard approach to obtain such a convex decomposition from the Carathéodory theorem and was also used in (Sumita et al. 2022). Recall that $v = \sigma(t)$. For brevity, we use v to denote $\sigma(t)$ in the superscript of λ_m^v and U_m^v.

To show that the weights $\{\lambda_m^v\}$ and the sets $\{U_m^v\}$ computed in Lines 11–21 of Alg. 2 satisfy Eq. (22), we show that: (i) When Alg 2 finishes Line 21, we have $M \leq |E(\sigma(t))| + 1$. This is because there is at most $|E(\sigma(t))|$ edges e with $\hat{y}_\sigma^t(e) > 0$, so after at most $|E(\sigma(t))|$ iterations of the loop between Lines 11–21 will end. (ii) For each $m \in [M]$, we have $\lambda_m^v \in [0,1]$. This is because for each $e \in E(\sigma(t))$ we have $\hat{y}_\sigma^t(e) \in [0,1]$ and, by the construction of each λ_m^v and $z_\sigma^t(e,m)$, we have $z_\sigma^t(e,m) \in [0,1]$ for each m, so we have $\lambda_m^v \in [0,1]$ by Line 16 of Alg. 2. (iii) For each $m \in [M]$, we have $U_m^v \in \mathcal{U}_{\sigma(t)}$ and $\mathbf{1}_{U_m^v} \in \mathcal{P}'_{\sigma(t)}$. This directly follows Lines 12–14 of Alg. 2. (iv) For each $e \in E(\sigma(t))$, we have

$$\hat{y}_\sigma^t(e) = \sum_{m=1}^{M} \lambda_m^v \mathbf{1}_{\{} e \in U_m^v\}. \tag{24}$$

This comes from the termination condition of the loop in Lines 11–20 of Alg. 2. In conclusion, the weights $\{\lambda_m^v\}$ and the sets $\{U_m^v\}$ computed in Lines 11–21 of Alg. 2 satisfy

$$\hat{y}_\sigma^t(e) = \sum_{m=1}^{M} \lambda_m^v \mathbf{1}_{\{e \in U_m^v\}}, \forall e \in E(\sigma(t)). \tag{25}$$

where $M \leq |E(\sigma(t))| + 1$.

Since for each $m \in [M]$, the Carathéodory set U_t^* is sampled as U_m^v with probability λ_m^v independently, we have the following equation on each single realization:

$$\Pr\{e \in U_t^*\} = \sum_{l=1}^{k} \lambda_m^v \mathbf{1}_{e \in U_m^v} = \hat{y}_\sigma^t(e), \forall e \in E(\sigma(t)). \tag{26}$$

As a result, over all realizations with the reward of edge $e \in E_\sigma^-(\sigma(t))$ that is realized as $r(e) = \tilde{r}$, we have

$$\Pr\left\{e \in U_t^* \mid r(e) = \tilde{r}\right\} = \mathbb{E}\left[\hat{y}_\sigma^t(e) \mid r(e) = \tilde{r}\right] = \mathbb{E}\left[x_e^*(\{r(e')\}_{e' \in E}) \mid r(e) = \tilde{r}\right], \forall e \in E_\sigma^-(\sigma(t)), \tag{27}$$

where the last equality comes from Eq. (18). Thus, Lemma 1 is proved. □

[3] Here we refer to the fact that the vector converted from $\{\hat{y}_\sigma^t(e)\}_{e \in E(\sigma(t))}$ is inside the polytope $\mathcal{P}_{\sigma(t)}$. Note that the set $\{\hat{y}_\sigma^t(e)\}_{e \in E(\sigma(t))}$ can be easily converted to a vector with $|E(\sigma(t))|$ dimensions, each corresponding to an edge $e \in E(\sigma(t))$ with value $\hat{y}_\sigma^t(e)$. For a cleaner presentation, we use set $\{\hat{y}_\sigma^t(e)\}_{e \in E(\sigma(t))}$ to refer to the vector converted from it. Throughout this appendix, we also use this approach to refer to the vectors converted from other sets.

Lemma 2. *(Bounded Size of Carathéodory Sets) For each $t \in [|V|]$, we have $|U_t^*| \le c_{\sigma(t)}$.*

Proof. Let $v = \sigma(t)$. For each $m \in [M]$, we have $|U_m^v| \le c_v$ by the construction of each U_m^v in Lines 12–14 of Alg. 2. Since $U_t^* \in \{U_m^v\}_{m\in[M]}$, we also have $|U_t^*| \le c_{\sigma(t)}$. Thus, Lemma 2 is proved. □

Lemma 3. *(Validity of Seesaw Probability $\phi_u(\sigma(t))$) For every t and for every edge $(u, \sigma(t)) \in U_t^*$, we have $\phi_u(\sigma(t)) \in [0, 1]$.*

Proof. **Overview:** This lemma will be proved once we show that $\sum_{e\in E(v)} y_e \le c_v$ for all vertices v. We will give this property by analyzing the solution of the optimization problem $\mathbf{P}^*$.

Detailed Proof: Recall that $\phi_u(\sigma(t))$ is defined as

$$\phi_u(\sigma(t)) = \frac{n_u(\sigma(t))}{c_u} \frac{1}{2 - \frac{1}{c_u}\sum_{\tau<t} y_{(u,\sigma(\tau))}}. \tag{28}$$

There are two multiplication terms in Eq. (28), and we first discuss the second term. By the definition of $y_{(u,\sigma(\tau))}$, we have $y_{(u,\sigma(\tau))} = \mathbb{E}_{i\sim I}[x^*_{(u,\sigma(\tau))}(\{r_i(e')\}_{\forall e'\in E})]$. By the construction of LP $\mathbf{P}^*$, we have $\sum_{e\in E(u)} x_e^*(\{r_i(e')\}_{\forall e'\in E}) \le c_u$ for each $u \in V$. As a result, we have

$$\begin{aligned}
\sum_{\tau<t} y_{(u,\sigma(\tau))} \le& \sum_{\tau=1}^{|V|} y_{(u,\sigma(\tau))} && (29)\\
=& \sum_{e\in E(u)} y_e && (30)\\
=& \sum_{e\in E(u)} \mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{\forall e'\in E})\right] && (31)\\
=& \mathbb{E}_{i\sim I}\left[\sum_{e\in E(u)} x_e^*(\{r_i(e')\}_{\forall e'\in E})\right] && (32)\\
\le& c_u. && (33)
\end{aligned}$$

Therefore, for the second term in Eq. (28) we have

$$\frac{1}{2 - \frac{1}{c_u}\sum_{\tau<t} y_{(u,\sigma(\tau))}} \in [\frac{1}{2}, 1]. \tag{34}$$

Next, we discuss the first term of $\phi_u(\sigma(t))$. Recall that by definition

$$n_u(\sigma(t)) = c_u - |\bar{q}^{\mathrm{c}}_{t-1} \cap E(u)|, \tag{35}$$

where $\bar{q}^{\mathrm{c}}_{t-1} = \bigcup_{\tau\in[t-1]} q^{\mathrm{c}}_\tau$ is the edges that are included in each capacity-friendly set before time slot t. By Eq. (35), we have $n_u(\sigma(t)) \le c_u$, which further implies that

$$\phi_u(\sigma(t)) \le 1. \tag{36}$$

Then we show that $n_u(\sigma(t)) \ge 0$ by contradiction. Suppose that there exists a node u that arrives in the time slot $\sigma^-(u) < t$ such that $(u, \sigma(t)) \in E$ and $n_u(\sigma(t)) < 0$. Note that in this case, since in time slot $\sigma^-(u)$ we have $U^*_{\sigma^-(u)} \le c_u$, at most c_u edges incident to u can be included in the capacity-friendly set in time slot $\sigma^-(u)$. As a result, there must exist another time slot $\tau \in (\sigma^-(u), t)$ such that $n_u(v_{\sigma(\tau)}) = 0$ but edge $(u, v_{\sigma(\tau)})$ is included in the capacity-friendly set q^{c}_τ. However, under this case, $n_u(v_{\sigma(\tau)}) = 0$ leads to $\phi_u(v_{\sigma(\tau)}) = 0$ and edge $(u, v_{\sigma(\tau)})$ cannot be included in the capacity-friendly set, resulting in a contradiction. Thus, we have proved that $n_u(\sigma(t)) \ge 0$.

In conclusion, for the first term in Eq. (28), we have $n_u(\sigma(t)) \in [0, c_u]$, leading to $n_u(\sigma(t))/c_u \in [0, 1]$. Combined with Eq. (34), we have $\phi_u(\sigma(t)) \in [0, 1]$. Thus, Lemma 3 is proved. □

Theorem 3. *(Capacity Friendly Probability) For each t and each edge $e \in E^-_\sigma(\sigma(t))$, let $\tilde{r}$ be a possible reward of e, we have $\Pr_{i\sim I}\{e \in q^c_t \mid r_i(e) = \tilde{r}\} = \frac{1}{2}\mathbb{E}_{i\sim I}[x^*_e(\{r_i(e')\}_{e'\in E}) \mid r_i(e) = \tilde{r}]$.*

Proof. **Overview:** We will prove this theorem by induction. We will first use Lemmas 1 and 3 to obtain the probability that each edge is included in the capacity-friendly set, and then we will use Lemma 2 to show that the capacity-friendly set does not violate the capacity constraint of each vertex. The key to this proof is that the probability $\phi_u(\sigma(t))$ is elaborated to balance between the expected capacity usage of vertex u and the remaining capacity of vertex u when vertex $\sigma(t)$ arrives.

Detailed Proof: We prove this theorem by induction. The base case is the first time slot $t \in [|V|]$ with $|E_\sigma^-(\sigma(t))| > 0$. We denote this time slot by $\tau = \min_{t\in[|V|]}\{|E_\sigma^-(\sigma(t))| > 0\}$. By the definition of τ, no edge has been included in the capacity-friendly set before the time slot τ. As a result, for each node u such that $(u, \sigma(\tau)) \in E_\sigma^-(\sigma(\tau))$, we have $n_u(\sigma(\tau)) = c_u$. For each node u such that $(u, \sigma(\tau)) \in E_\sigma^-(\sigma(\tau))$ and for each $\tau' < \tau$, we have $y_{(u,\sigma(\tau'))} = 0$, since $(u, \sigma(\tau')) \notin E$. Therefore, we have $\phi_u(\sigma(\tau)) = \frac{1}{2}$. By the construction of the capacity-friendly set, an edge $(u, \sigma(\tau'))$ is included in q_τ^{c} only if it is included in U_τ^* and then sampled into q_τ^{c} with probability $\phi_u(\sigma(\tau))$. We note that the probability $\phi_u(\sigma(\tau))$ is independent of the edge reward revealed in time slot τ. Therefore, we have the following equations for every $e = (u, \sigma(\tau')) \in E_\sigma^-(\sigma(\tau))$ in the base case:

$$\Pr_{i\sim I}\{e \in q_\tau^{\mathrm{c}} \mid r_i(e) = \tilde{r}\} = \Pr_{i\sim I}\{e \in U_\tau^* \mid r_i(e) = \tilde{r}\} \times \phi_u(\sigma(\tau)) \tag{37}$$

$$= \frac{1}{2}\mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{\forall e'\in E}) \mid r_i(e) = \tilde{r}\right], \ \forall \tilde{r} \in \mathrm{supp}(\mathcal{R}(e)). \tag{38}$$

Thus, the base case is proved.

For the induction step, we assume that it holds for every $t' < t$ that

$$\Pr_{i\sim I}\{e \in q_{t'}^{\mathrm{c}} \mid r_i(e) = \tilde{r}\} = \frac{1}{2}\mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{\forall e'\in E}) \mid r_i(e) = \tilde{r}\right], \forall e \in E_\sigma^-(\sigma(t')), \tilde{r} \in \mathrm{supp}(\mathcal{R}(e)). \tag{39}$$

Then, for each edge $e' = (u, \sigma(t)) \in E_\sigma^-(\sigma(t))$ we have

$$\begin{aligned}
\Pr_{i\sim I}\{e' \in q_t^{\mathrm{c}} \mid r_i(e') = \tilde{r}\} =& \Pr_{i\sim I}\{e' \in U_t^* \mid r_i(e') = \tilde{r}\} \times \sum_{l\in[c_u]} \Pr_{i\sim I}\{n_u(\sigma(t)) = l | r_i(e') = \tilde{r}\} \\
&\times \frac{l}{c_u} \times \frac{1}{2 - \frac{1}{c_u}\sum_{t'<t} y_{(u,\sigma(t'))}} && (40)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \sum_{l\in[c_u]} \left(\Pr_{i\sim I}\{n_u(\sigma(t)) = l\} \cdot l\right) \\
&\times \frac{1}{c_u} \times \frac{1}{2 - \frac{1}{c_u}\sum_{t'<t} y_{(u,\sigma(t'))}} && (41)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \mathbb{E}_{i\sim I}\left[n_u(\sigma(t))\right] \\
&\times \frac{1}{2c_u - \sum_{t'<t} y_{(u,\sigma(t'))}} && (42)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \frac{1}{2c_u - \sum_{t'<t} y_{(u,\sigma(t'))}} \\
&\times \mathbb{E}_{i\sim I}\left[c_u - |\bar{q}_{t-1}^{\mathrm{c}} \cap E(u)|\right] && (43)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \frac{1}{2c_u - \sum_{t'<t} y_{(u,\sigma(t'))}} \\
&\times \left(c_u - \mathbb{E}_{i\sim I}\left[|\bar{q}_{t-1}^{\mathrm{c}} \cap E(u)|\right]\right) && (44)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \frac{1}{2c_u - \sum_{t'<t} y_{(u,\sigma(t'))}} \\
&\times \left(c_u - \sum_{t'<t} \Pr_{i\sim I}\{(u, \sigma(t')) \in q_{t'}^{\mathrm{c}}\}\right) && (45)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \frac{1}{2c_u - \sum_{t'<t} y_{(u,\sigma(t'))}} \\
&\times \left(c_u - \frac{1}{2}\sum_{t'<t} \mathbb{E}_{i\sim I}\left[x_{(u,\sigma(t'))}^*(\{r_i(e'')\}_{\forall e''\in E})\right]\right) && (46)\\
=& \mathbb{E}_{i\sim I}\left[x_{e'}^*(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e') = \tilde{r}\right] \times \frac{1}{2c_u - \sum_{t'<t} y_{(u,\sigma(t'))}} \\
&\times \left(c_u - \frac{1}{2}\sum_{t'<t} y_{(u,\sigma(t'))}\right) && (47)
\end{aligned}$$

$$=\frac{1}{2}\mathbb{E}_{i\sim I}\left[x^*_{e'}(\{r_i(e'')\}_{\forall e''\in E}) \mid r_i(e')=\tilde{r}\right]. \tag{48}$$

where (i) Eq. (40) comes from Lemma 1 and the fact that $n_u(\sigma(t))$ is independent from $r_i(e')$, $\forall e' \in E^-_\sigma(\sigma(t))$; (ii) Eq. (43) comes from the definition of $n_u(\cdot)$; (iii) Eq. (45) comes from the definition of $\bar{q}^{\mathfrak{c}}_{t-1}$ and the fact that every edge in $\bar{q}^{\mathfrak{c}}_{t-1} \cap E(u)$ connects node u to a node $\sigma(t')$ that arrives in time slot t' before t; (iv) Eq. (46) comes from the induction assumption; and (v) Eq. (48) comes from the definition of $y_{(u,\sigma(t'))}$.

By Eq. 48, Theorem 3 is proved. Note that Eqs. (40)–(48) attribute to the dedicated design of the seesaw probability $\phi_u(\sigma(t))$ for the multi-capacity vertices in OCGMK, which serves as a critical component of our probabilistic matching approach. Since $\phi_u(\sigma(t))$ is tuned with the remaining capacity and the average preference. The seesaw probability in all time slots together perfectly matches the average preference, so they effectively strike a balance between consuming u's capacity and conserving it for a potentially better future reward, which finally leads to Eq. 48.

□

E.2 Analysis of the Knapsack-Friendly Set

The analysis of the assignment acceptance phase is inspired by (Jiang, Ma, and Zhang 2022), following a similar road map. However, since we design a new probability rule to select edges into the knapsack-friendly set, we require different characterizations of the reference consumption distribution in Lemmas 4–5.

In the assignment acceptance phase, OCKA constructs the knapsack-friendly set by calculating a safe range of the reference knapsack consumption X_j. We need to prove the existence of θ^-_j, which is the left boundary of the safe range, so that the reference consumption distribution constructed by Eqs. (13)–(14) is a valid probability distribution. The existence of θ^-_j is proved by induction on j. In each induction step, we need an important property of the reference consumption distribution, as stated in Lemma 4.

Lemma 4. *(Property of the Reference Consumption Distribution) Let $\mu_j(a,b] = \Pr\{\tilde{X}_j \in (a,b]\}$. Suppose that $\tilde{X}_j$ is a valid probability distribution. For any $b \in (0, \frac{K}{2}]$, the following inequality holds for all $j = 0, 1, \ldots, |E|$:*

$$\frac{1}{\gamma}\mu_j(0,b] \le \exp\left(-\frac{1}{\gamma}\mu_j(b, K-b]\right). \tag{49}$$

Proof. **Overview:** We will prove this lemma by induction and divide and conquer. With the inductive hypothesis that this inequality holds for $j'-1$, we will categorize the evolution of μ_j into four cases based on the relationships among b, $\theta^-_{j'}$ and $d(e_{j'})$ and prove the above inequality for them one by one.

Detailed Proof: We prove this lemma by induction. The base case is at $j=0$. By the construction of $\tilde{X}_0$, we have $\Pr\{\tilde{X}_0 = 0\} = 1$, and

$$\mu(0,b] = \mu(b, K-b] = 0,\ \forall b \in (0, \frac{K}{2}]. \tag{50}$$

As a result, Eq. (49) holds for $j=0$.

For the induction step, we assume that Eq. (49) holds for all j' such that $j' < j$ and we prove Eq. (49) for j. We achieve this by discussing four cases based on the different values of θ^-_j and $d(e_j)$.

Case 1: $\theta^-_j > 0$. Since we only move the probability mass in the interval $[\theta^-_j, K-d(e_j)]$ towards distribution $\tilde{X}_{j-1}$ to the interval $[\theta^-_j + d(e_j), K]$ of distribution $\tilde{X}_j$, there is no probability mass moved from point 0 of distribution $\tilde{X}_{j-1}$ to the interval $(0, \theta^-_j]$. Since we may have $b \le \theta^-_j$ or $b > \theta^-_j$, we have

$$\mu_j(0,b] \le \mu_{j-1}(0,b], \tag{51}$$

and

$$\exists z \ge 0, \text{ such that } \mu_j(0,b] = \mu_{j-1}(0,b] - z. \tag{52}$$

Since the probability mass z moved from the interval $(0,b]$ of distribution $\tilde{X}_{j-1}$ may be moved to either the intervals $(b, K-b]$ or $(K-b, K]$ of $\tilde{X}_j$, and since the probability mass in the interval $(b, K-b]$ may also be moved to the interval $(K-b, K]$ of $\tilde{X}_j$, we have

$$\mu_j(b, K-b] \le \mu_{j-1}(b, K-b] + z. \tag{53}$$

By Eq. (53), we have

$$\exp(-\frac{1}{\gamma}\mu_j(b, K-b]) \ge \exp(-\frac{1}{\gamma}(\mu_{j-1}(b, K-b] + z)) \tag{54}$$

$$
\begin{aligned}
&= \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b]) \cdot \exp(-\frac{z}{\gamma}) && (55)\\
&\geq \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b]) \cdot (1-\frac{z}{\gamma}) && (56)\\
&= \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b]) - \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b])\frac{z}{\gamma} && (57)\\
&\geq \frac{1}{\gamma}\mu_{j-1}(0, b] - \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b])\frac{z}{\gamma} && (58)\\
&\geq \frac{1}{\gamma}\mu_{j-1}(0, b] - \frac{z}{\gamma} && (59)\\
&= \frac{1}{\gamma}\mu_{j}(0, b], && (60)
\end{aligned}
$$

where (i) inequality (56) comes from the fact that $e^x \geq 1-x$ when $x \geq 0$; (ii) inequality (58) comes from the inductive hypothesis; (iii) inequality (59) comes from the fact that $e^x \leq 1-x$ when $x \leq 0$, and (iv) inequality (60) comes from Eq. (52). Thus, inequality (49) holds for the induction step in case 1.

Case 2: $\theta_j^- = 0$ and $d(e_j) \leq b$. In this case, by the construction of OCKA, a probability mass of $\psi_j \Pr\{\tilde{X}_{j-1} = 0\}$ is moved from point 0 of distribution $\tilde{X}_{j-1}$ to interval $(0, b]$ of distribution $\tilde{X}_j$. Since the probability mass in the interval $(0, b]$ of distribution $\tilde{X}_{j-1}$ may also be moved to the interval $(b, K-b]$ of distribution $\tilde{X}_j$, we have

$$
\mu_j(0, K-b] \leq \mu_{j-1}(0, K-b] + \psi_j \Pr\{\tilde{X}_{j-1} = 0\}. \tag{61}
$$

By $\theta_j^- = 0$ and the definition of θ_j^-, we have

$$
\mu(0, K-d(e_j)] = \Pr\{\tilde{X}_{j-1} \in (0, K-d(e_j)]\} \leq \gamma. \tag{62}
$$

By $d(e_j) \leq b$, we have $K - d(e_j) \geq K - b$ and

$$
\mu_{j-1}(0, K-b] \leq \mu_{j-1}(0, K-d(e_j)]. \tag{63}
$$

By Eqs. (61)–(63), we find a lower bound on $\mu_j(0, K-b]$ by

$$
\begin{aligned}
\mu_j(0, K-b] \leq& \mu_{j-1}(0, K-b] + \psi_j \Pr\{\tilde{X}_{j-1} = 0\} && (64)\\
\leq& \mu_{j-1}(0, K-d(e_j)] + \psi_j \Pr\{\tilde{X}_{j-1} = 0\} && (65)\\
=& \mu_{j-1}(0, K-d(e_j)] + \frac{\Pr\{\tilde{X}_{j-1} = 0\}(\gamma - \sum_{l=l_2+1}^{l_1} \Pr\{\tilde{X}_{j-1} = b_l\})}{\beta \times \Pr\{\tilde{X}_{j-1} = \theta_j^-\}} && (66)\\
=& \mu_{j-1}(0, K-d(e_j)] + \frac{\Pr\{\tilde{X}_{j-1} = 0\}(\gamma - \mu_{j-1}(0, K-d(e_j)])}{\beta \times \Pr\{\tilde{X}_{j-1} = 0\}} && (67)\\
=& \mu_{j-1}(0, K-d(e_j)] \left(1 - \frac{1}{\beta}\right) + \frac{\gamma}{\beta} && (68)\\
\leq& \gamma \left(1 - \frac{1}{\beta}\right) + \frac{\gamma}{\beta} && (69)\\
=& \gamma, && (70)
\end{aligned}
$$

where (i) inequality (65) comes from inequality (63); (ii) inequality (67) comes from the fact that in the j-th edge iteration we have $b_{l_1} = 1 - d_{e_j}$ and, in case 2, we have $\theta_j^- = 0$; (iii) and inequality (69) comes from inequality (62).

Since $e^{-x} \geq 1 - x$ when $x \geq 0$, we have

$$
\begin{aligned}
\exp(-\frac{1}{\gamma}\mu_j(b, K-b]) \geq& 1 - \frac{1}{\gamma}\mu_j(b, K-b] && (71)\\
\geq& 1 - \frac{1}{\gamma}\mu_j(0, K-b] + \frac{1}{\gamma}\mu_j(0, b] && (72)\\
\geq& \frac{1}{\gamma}\mu_j(0, b], && (73)
\end{aligned}
$$

where inequality (73) comes from inequality (70). Thus, inequality (49) holds for the induction step in case 2.

Case 3**:** $\theta_j^- = 0$ **and** $d(e_j) \in (b, K-b]$**.** In this case, the total amount of probability mass moved from the distribution $\tilde{X}_{j-1}$, when it is updated to $\tilde{X}_j$, is γ/β. This amount of probability mass is moved to intervals $(b, K-b]$ and $(K-b, K]$ in distribution $\tilde{X}_j$. As a result, we have

$$\mu_j(b, K-b] \leq \mu_{j-1}(b, K-b] + \gamma/\beta. \tag{74}$$

Since $d(e_j) \in (b, K-b]$, we have $K - d(e_j) \geq b$. Therefore, $1/\beta$ fraction of the probability mass in interval $(0, b]$ of $\tilde{X}_{j-1}$ is moved to interval $(b, K]$ towards $\tilde{X}_j$. As a result, we have

$$\mu_j(0, b] = (1 - \frac{1}{\beta})\mu_{j-1}(0, b]. \tag{75}$$

Combining inequality (74) and Eq. (75), we have

$$\begin{aligned}
\exp(-\frac{1}{\gamma}\mu_j(b, K-b]) \geq& \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b] - \frac{1}{\beta}) && (76)\\
=& \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b]) \cdot \exp(-\frac{1}{\beta}) && (77)\\
\geq& \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b]) \cdot (1 - \frac{1}{\beta}) && (78)\\
\geq& \frac{1}{\gamma}\mu_{j-1}(0, b] \cdot (1 - \frac{1}{\beta}) && (79)\\
=& \frac{1}{\gamma}\mu_j(0, b], && (80)
\end{aligned}$$

where (i) inequality (76) comes from inequality (74); (ii) inequality (78) comes from the fact that $e^{-x} \geq 1 - x$ when $x \geq 0$; (iii) inequality (79) comes from the inductive hypothesis; and (iv) Eq. (80) comes from Eq. (75). Thus, inequality (49) holds for the induction step in case 3.

Case 4**:** $\theta_j^- = 0$ **and** $d(e_j) > K - b$**.** In this case, we have $K - d(e_j) < b$. A $1/\beta$ fraction of the probability mass in interval $[0, K - d(e_j)]$ of $\tilde{X}_{j-1}$ is moved to interval $[d(e_j), K]$ in distribution $\tilde{X}_j$. The probability mass in interval $(b, K-b]$ of $\tilde{X}_{j-1}$ is unchanged in $\tilde{X}_j$. Since $[0, K - d(e_j)] \subset [0, b]$ and $[d(e_j), K] \subset [K-b, K]$, we have

$$\mu_j(0, b] \leq \mu_{j-1}(0, b], \tag{81}$$

and

$$\mu_j(b, K-b] = \mu_{j-1}(b, K-b]. \tag{82}$$

As a result, we have

$$\begin{aligned}
\exp(-\frac{1}{\gamma}\mu_j(b, K-b]) =& \exp(-\frac{1}{\gamma}\mu_{j-1}(b, K-b] && (83)\\
\geq& \frac{1}{\gamma}\mu_{j-1}(0, b] && (84)\\
\geq& \frac{1}{\gamma}\mu_j(0, b], && (85)
\end{aligned}$$

where (i) Eq. (83) comes from Eq. (82); (ii) inequality (84) comes from the inductive hypothesis; and (iii) inequality (85) comes from inequality (81). Thus, inequality (49) holds for the induction step in case 4.

From the above discussion of cases 1–4, we have proved that inequality (49) holds for the induction step. Thus, this lemma is proved. □

Then, Lemma 5 asserts the existence of θ_j^-.

Lemma 5. *(Existence of the Safe Range) With* $\gamma = \frac{1}{3+e^{-2}}$*, for every* $j \in [|E|]$*, there exists* θ_j^- *and* l_2*, such that* $\theta_j^- = b_{l_2}$ *and* $l_2 = \arg\min_{l \in [L]}\{\Pr\{\tilde{X}_{j-1} \in (b_l, K - d(e_j)]\} \leq \gamma\}$.

Proof. **Overview:** We will prove this lemma by showing that for every j the distribution $\tilde{X}_{j-1}$ has at least a probability of γ to be realized in 0, and we show this through induction. For the induction step, we will give the lower bound for $\Pr\{\tilde{X}_{j-1} = 0\}$ by Lemma 4 and by the construction of Alg. 3.

Detailed Proof: We prove this lemma by induction. The base case is at $j = 0$. By the construction of $\tilde{X}_0$, we have $\Pr\{\tilde{X}_0 = 0\} = 1$, leading to $l_2 = 1$ and $\theta_0^- = 0$.

For the induction step, we assume that $\theta_{j'}^-$ exists for every j' such that $0 < j' \leq j$, thus each $\tilde{X}_{j'}$, including $\tilde{X}_j$, is well constructed. To show the existence of l_2 and θ_{j+1}^- for iteration $j+1$, we only need to show that $\Pr\{\tilde{X}_j = 0\} \geq \gamma$. Equivalently, we define $U_j(s) = \mu_j(0, s]$ and will prove $U_j(K) \leq 1 - \gamma$ for the induction step.

For each j' such that $0 < j' \leq j$, by the construction of $\tilde{X}_{j'}$ in Eqs. (13)–(14), we have $\mathbb{E}[\tilde{X}_{j'}] = \mathbb{E}[\tilde{X}_{j'-1}] + (\gamma d(e_{j'}))/\beta$, so we have

$$\begin{aligned}
\mathbb{E}[\tilde{X}_j] =& \frac{\gamma}{\beta} \sum_{j'=1}^{j} d(e_{j'}) && (86)\\
\leq& \frac{\gamma}{\beta} \sum_{j'=1}^{|E|} d(e_{j'}) && (87)\\
=& \frac{\gamma}{\beta} \sum_{e \in E} d(e) && (88)\\
=& \gamma K, && (89)
\end{aligned}$$

where Eq. (89) comes from the definition $\beta = \sum_{e \in E} d(e)/K$. As a result, we have

$$\begin{aligned}
\gamma \geq& \frac{1}{K} \mathbb{E}[\tilde{X}_j] && (90)\\
=& \frac{1}{K} \sum_{s \in \text{supp}(\tilde{X}_j)} s \cdot \Pr\{\tilde{X}_j = s\} && (91)\\
=& \frac{1}{K} \int_{s=0}^{K} s \cdot \Pr\{\tilde{X}_j = s\} ds && (92)\\
=& \frac{1}{K} \int_{s=0}^{K} s \cdot d \Pr\{\tilde{X}_j \in [0, s]\} && (93)\\
=& \frac{1}{K} \int_{s=0}^{K} s \cdot d \left(U_j(s) + \Pr\{\tilde{X}_j = 0\} \right). && (94)
\end{aligned}$$

Since there is

$$d \left(s \cdot \left(U_j(s) + \Pr\{\tilde{X}_j = 0\} \right) \right) = s \cdot dU_j(s) + \left(U_j(s) + \Pr\{\tilde{X}_j = 0\} \right) \cdot ds, \tag{95}$$

by Eq. (94) we have

$$\begin{aligned}
\gamma \geq& \frac{1}{K} \left[\int_{s=0}^{K} d \left(s \cdot \left(U_j(s) + \Pr\{\tilde{X}_j = 0\} \right) \right) - \int_{s=0}^{K} \left(U_j(s) + \Pr\{\tilde{X}_j = 0\} \right) \cdot ds \right] && (96)\\
=& \frac{1}{K} \left[K \cdot \left(U_j(K) + \Pr\{\tilde{X}_j = 0\} \right) - \int_{s=0}^{K} U_j(s) ds - K \cdot \Pr\{\tilde{X}_j = 0\} \right] && (97)\\
=& U_j(K) - \frac{1}{K} \left[\int_{s=0}^{K} U_j(s) ds \right]. && (98)
\end{aligned}$$

Inequality (98) can also be expressed as

$$U_j(K) \leq \frac{1}{K} \left[\int_{s=0}^{K} U_j(s) ds \right] + \gamma. \tag{99}$$

To bound $U_j(K)$, we first bound $\int_{s=0}^{K} U_j(s) ds$. When $U_j(K) \leq \gamma = \frac{1}{3+e^{-2}}$, it is straightforward that $U_j(K) \leq 1 - \gamma$, and the induction step is proved. We then focus on the case that $U_j(K) \geq \gamma$. In this case, we have $U_j(K)/\gamma \geq 1$. Note that the value of function $f(x) = x - \ln(x)$ monotonically decreases from ∞ to 1 when x increases from 0 (exclusive) to 1 (inclusive). As a result, these exists u^* such that $u^* \in (0, 1]$ and satisfies

$$u^* - \ln(u^*) = U_j(K)/\gamma \geq 1. \tag{100}$$

We then define s^* by

$$s^* = \begin{cases} \arg\min_{s\in(0,\frac{K}{2}]}\{s : U_j(s) \geq \gamma u^*\}, & \text{if } U_j(\frac{K}{2}) \geq \gamma u^*, \\ \frac{K}{2}, \text{ if } U_j(\frac{K}{2}) < \gamma u^*. \end{cases} \tag{101}$$

Since $U_j(\cdot)$ has discontinuities in the interval $(0, K]$, we also define $U_j^-(s^*) = \lim_{s\to s^*} U_j(s)$ as the left-hand limit of $U_j(s)$ at s^*. Since $U_j(s)$ is non-decreasing when s increases from 0 (exclusive) to K (inclusive), we can bound $\int_{s=0}^{K} U_j(s)ds$ by

$$\begin{aligned}
\int_{s=0}^{K} U_j(s)ds =& \left(\int_{s=0}^{s^*} + \int_{s=s^*}^{\frac{K}{2}} + \int_{s=\frac{K}{2}}^{K-s^*} + \int_{s=K-s^*}^{K}\right) U_j(s)ds && (102)\\
\leq& s^* \cdot U_j^-(s^*) + \left(\int_{s=s^*}^{\frac{K}{2}} + \int_{s=\frac{K}{2}}^{K-s^*}\right) U_j(s)ds + s^* \cdot U_j(K) && (103)\\
=& s^* \left(U_j^-(s^*) + U_j(K)\right) + \left(\int_{s=s^*}^{\frac{K}{2}} + \int_{s=\frac{K}{2}}^{K-s^*}\right) U_j(s)ds && (104)\\
\leq& s^* \left(2\gamma u^* - \gamma \ln(u^*)\right) + \left(\int_{s=s^*}^{\frac{K}{2}} + \int_{s=\frac{K}{2}}^{K-s^*}\right) U_j(s)ds && (105)\\
=& s^* \left(2\gamma u^* - \gamma \ln(u^*)\right) + \int_{s^*}^{\frac{K}{2}} \left(U_j(s) + U_j(K-s)\right) ds, && (106)
\end{aligned}$$

where inequality (105) comes from Eqs. (100)–(101).

By Lemma 4, for every $s \in [s^*, \frac{K}{2}]$, we have

$$\begin{aligned}
\frac{1}{\gamma} U_j(s) =& \frac{1}{\gamma}\mu_j(0, s] && (107)\\
\leq& \exp\left(-\frac{1}{\gamma}\mu_j(s, K-s]\right) && (108)\\
=& \exp\left(-\frac{1}{\gamma}\left(U_j(K-s) - U_j(s)\right)\right). && (109)
\end{aligned}$$

As a result, we have

$$\ln\left(\frac{U_j(s)}{\gamma}\right) \leq \frac{U_j(s)}{\gamma} - \frac{U_j(K-s)}{\gamma}, \tag{110}$$

and

$$\frac{U_j(K-s)}{\gamma} \leq \frac{U_j(s)}{\gamma} - \ln\left(\frac{U_j(s)}{\gamma}\right). \tag{111}$$

By Eqs. (106) and (111), we have

$$\begin{aligned}
\int_{s=0}^{K} U_j(s)ds \leq& s^* \left(2\gamma u^* - \gamma \ln(u^*)\right) + \int_{s^*}^{\frac{K}{2}} \left(U_j(s) + U_j(K-s)\right) ds && (112)\\
\leq& s^* \left(2\gamma u^* - \gamma \ln(u^*)\right) + \int_{s^*}^{\frac{K}{2}} \left(2U_j(s) - \gamma \ln\left(\frac{U_j(s)}{\gamma}\right)\right) ds. && (113)
\end{aligned}$$

Considering Eq. (113), we discuss the upper bound on $\int_{s=0}^{K} U_j(s)ds$ in two cases: (i) $s^* = \frac{K}{2}$ and (ii) $s^* < \frac{K}{2}$.

Case 1: $s^* = \frac{K}{2}$. In this case, Eq. (113) leads to

$$\begin{aligned}
\int_{s=0}^{K} U_j(s)ds \leq& s^*(2\gamma u^* - \gamma \ln(u^*)) && (114)\\
=& K\left(\gamma u^* - \frac{\gamma \ln(u^*)}{2}\right). && (115)
\end{aligned}$$

By Eq. (99), the definition of u^* in Eq. (100), and Eq. (115), we have

$$U_j(K) = \gamma(u^* - \ln(u^*)) \leq \gamma + \gamma u^* - \frac{\gamma \ln(u^*)}{2}, \tag{116}$$

and

$$u^* \geq e^{-2}. \tag{117}$$

Since $x - \ln(x)$ is non-increasing on $(0, 1)$, we have

$$\begin{aligned} U_j(K) =& \gamma(u^* - \ln(u^*)) & (118)\\ \leq& \gamma(e^{-2} - \ln(e^{-2}) & (119)\\ =& \gamma e^{-2} + 2\gamma & (120)\\ =& \frac{2 + e^{-2}}{3 + e^{-2}} & (121)\\ =& 1 - \frac{1}{3 + e^{-2}} & (122)\\ =& 1 - \gamma. & (123) \end{aligned}$$

Thus, $1 - U_j(K) \geq \gamma$ and the induction step in case 1 is proved.

Case 2: $s^* < \frac{K}{2}$. In this case, we have

$$\gamma u^* \leq U_j(s^*) \leq U_j(s) \leq U_j(\frac{1}{2}), \forall s \in [s^*, \frac{K}{2}], \tag{124}$$

where the first inequality comes from the definition of s^* in Eq. (101), and the last two inequalities come from the fact that $U_j(\cdot)$ is non-decreasing on the interval $(0, \frac{K}{2}]$. By Lemma 4, we have

$$\begin{aligned} \frac{U_j\left(\frac{K}{2}\right)}{\gamma} =& \frac{1}{\gamma}\mu_j(0, \frac{K}{2}] & (125)\\ \leq& \exp\left(-\frac{1}{\gamma}\mu_j(\frac{K}{2}, \frac{K}{2}]\right) & (126)\\ =& 1. & (127) \end{aligned}$$

By Eqs. (124) and (127), we have

$$\gamma u^* \leq U_j(s) \leq \gamma. \tag{128}$$

Let us denote $g(x) = 2x - \gamma \ln(x/\gamma)$. Since $g(x)$ is convex, it is also quasi-convex. As a result, by Eq. (128), we have

$$g(U_j(s)) \leq \max\{g(\gamma u^*), g(\gamma)\} \tag{129}$$

and

$$2U_j(s) - \gamma \ln\left(\frac{U_j(s)}{\gamma}\right) \leq \max\{2\gamma u^* - \gamma \ln(u^*), 2\gamma\}. \tag{130}$$

By Eqs. (113) and (130), we have

$$\begin{aligned} \int_{s=0}^{K} U_j(s)ds \leq& s^*\left(2\gamma u^* - \gamma \ln(u^*)\right) + \int_{s^*}^{\frac{K}{2}} \left(2U_j(s) - \gamma \ln\left(\frac{U_j(s)}{\gamma}\right)\right) ds & (131)\\ \leq& s^*\left(2\gamma u^* - \gamma \ln(u^*)\right) + \left(\frac{K}{2} - s^*\right) \max\{2\gamma u^* - \gamma \ln(u^*), 2\gamma\}. & (132) \end{aligned}$$

Next, we consider the two possible outcomes in the maximization operation in Eq. (132). On the one hand, if $2\gamma u^* - \gamma \ln(u^*) \leq 2\gamma$, Eq. (132) leads to

$$\begin{aligned} \int_{s=0}^{K} U_j(s)ds \leq& s^*\left(2\gamma u^* - \gamma \ln(u^*)\right) + \left(\frac{K}{2} - s^*\right) \cdot 2\gamma & (133)\\ \leq& s^* \cdot 2\gamma + \left(\frac{K}{2} - s^*\right) \cdot 2\gamma & (134) \end{aligned}$$

$$=\gamma K. \tag{135}$$

By Eq. (99), we have

$$\begin{aligned}
U_j(K) \leq & \frac{1}{K}\int_{s=0}^{K} U_j(s)ds + \gamma && (136)\\
\leq & 2\gamma && (137)\\
= & \frac{2}{3+e^{-2}} && (138)\\
< & \frac{2+e^{-2}}{3+e^{-2}} && (139)\\
= & 1-\gamma, && (140)
\end{aligned}$$

which proves the induction step.

On the other hand, if $2\gamma u^* - \gamma \ln(u^*) > 2\gamma$, Eq. (132) leads to

$$\begin{aligned}
\int_{s=0}^{K} U_j(s)ds \leq & \frac{K}{2}\left(2\gamma u^* - \gamma \ln(u^*)\right) && (141)\\
= & K \cdot \left(\gamma u^* - \frac{\gamma \ln(u^*)}{2}\right). && (142)
\end{aligned}$$

Since inequality (142) is identical to inequality (115), the remainder of the proof for the condition $2\gamma u^* - \gamma \ln(u^*) > 2\gamma$ in case 2 is identical to the proof between Eq. (115) and Eq. (123), which eventually leads to $1 - U_j(K) \geq \gamma$. Thus, the induction step in case 2 is also proved.

Cases 1 and 2 together prove the induction step. Thus, Lemma 5 is proved. Lemma 5 is achieved by the design of both the safe range and the probability ψ_j, which serve as critical components for our matching acceptance approach. They effectively balance between the budget consumption and reserving cost budget, so that they always leave enough chance for current and future edge rewards. □

Finally, Theorem 4 gives the probability that an edge is included in q_t^{k}.

Theorem 4. *(Knapsack Friendly Probability) For every t and $e \in E_\sigma^-(\sigma(t))$, the probability that an edge e is included in the knapsack-friendly set q_t^k is γ/β.*

Proof. **Overview:** Given Lemma 5, we can show that Theorem 4 follows directly from the construction of the knapsack-friendly set. The term γ comes from the safe range, and the term $1/\beta$ comes from the probability that we include an edge in q_t^{k} if the reference knapsack consumption is in the safe range.

Detailed Proof: The key to proving Theorem 4 is to show the existence of θ_j^- and l_2, which is proved in Lemma 5. In each edge iteration j, we focus on Lines 13–17 of Alg. 3. Recall that, because of the randomness of each knapsack-friendly set q_t^{k}, the reference knapsack consumption X_{j-1} is different over different realizations and over different algorithm runs on one realization. As a result, for a fixed problem instance, X_j is a random variable. By the construction of the reference consumption distribution $\tilde{X}_j$, it is the *ex-ante* distribution of X_j. In other words, over all algorithm runs and over all realizations, X_j follows the probability distribution $\tilde{X}_j$. For each t and $e_j \in E_\sigma^-(\sigma(t))$, by the construction of θ_j^- and θ_j^+ and our decision for e_j, we have

$$\begin{aligned}
\Pr\{e_j \in q_t^{\mathrm{k}}\} = & \frac{1}{\beta}\Pr\{X_{j-1} \in (\theta_j^-, \theta_j^+]\} + \frac{1}{\psi}\Pr\{X_{j-1} = \theta_j^-\} && (143)\\
= & \frac{\gamma}{\beta}. && (144)
\end{aligned}$$

The equation (143) comes from the different probability rules that we use to include edge e_j when X_{j-1} has different values. By Eq. (144), this theorem is proved. □

E.3 Competitive Ratio of the OCKA Algorithm

Before we show the competitive ratio of OCKA, we still need one more lemma to give an upper bound of $\mathbb{E}_{i\sim I}[\mathrm{OPT}(i)]$, which is stated in Lemma 6 below.

Lemma 6. *(Offline Optimal Upper Bound) Let the optimal value achieved by LP $\boldsymbol{P}^*(\{r_i(e)\})$ on realization i be $OPT^*(i)$. We have*

$$\mathbb{E}_{i\sim I}[OPT^*(i)] \geq \mathbb{E}_{i\sim I}[OPT(i)]. \tag{145}$$

Proof. This lemma comes directly from the following facts: (i) $\text{OPT}(i)$ is the objective value of LP $\mathbf{P}$ on the realization i, and (ii) for the same realization i, LP $\mathbf{P}^*(\{r_i(e)\})$ is a relaxation of LP $\mathbf{P}$ with the knapsack constraint not imposed. □

In each time slot t, after OCKA constructs the capacity-friendly set q_t^{c} and the knapsack-friendly set q_t^{k}, it will accept every edge $(u, \sigma(t)) \in Q_t = q_t^{\text{c}} \cap q_t^{\text{k}}$. Therefore, the probability that an edge is included in Q_t is the joint probability that it is included in both q_t^{c} and q_t^{k}, which leads to the following competitive ratio:

Theorem 5. *(OCKA Competitive Ratio, Reproduced from Theorem 1 of the Main Paper) 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 $\beta = \sum_{e\in E} d(e)/K$.*

Proof. **Overview:** By Theorems 3–4, we give a lower bound on the probability that OCKA accepts each edge, which leads to the average online performance of OCKA. By Lemma 6, the competitive ratio is given as the ratio between the average performance of OCKA and the upper bound of $\mathbb{E}_{i\in I}[\text{OPT}(i)]$.

Detailed Proof: We first define $\text{ALG}_e(i)$ as the expected reward obtained by OCKA on the realization i from accepting edge e, and define $\text{OPT}_e^*(i)$ as the expected reward obtained by LP $\mathbf{P}^*$ on the realization i from accepting edge e. We use $t(e)$ to denote the time slot t that $e \in E_\sigma^-(\sigma(t))$. Then, we have

$$\begin{aligned}
\mathbb{E}_{i\sim I}[\text{ALG}(i)] &= \sum_{e\in E} \mathbb{E}_{i\sim I}[\text{ALG}_e(i)] && (146)\\
&= \sum_{e\in E} \sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\mathbb{E}_{i\sim I}[\text{ALG}_e(i) \mid r_i(e)=\tilde{r}] && (147)\\
&= \sum_{e\in E} \sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\mathbb{E}_{i\sim I}[r_i(e)\cdot \Pr\{e\in Q_{t(e)} \mid i\} \mid r_i(e)=\tilde{r}] && (148)\\
&= \sum_{e\in E} \sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\cdot \tilde{r}\cdot \Pr_{i\sim I}\{e\in Q_{t(e)} \mid r_i(e)=\tilde{r}\}. && (149)
\end{aligned}$$

Since (i) $Q_{t(e)} = q_{t(e)}^{\text{k}} \cap q_{t(e)}^{\text{c}}$, (ii) the construction of $q_{t(e)}^{\text{c}}$ and $q_{t(e)}^{\text{k}}$ are independent of each other, and (iii) the construction of $q_{t(e)}^{\text{k}}$ is independent of the realization of the edge reward, we have

$$\begin{aligned}
\Pr_{i\sim I}\{e\in Q_{t(e)} \mid r_i(e)=\tilde{r}\} &= \Pr_{i\sim I}\{e\in q_{t(e)}^{\text{c}} \mid r_i(e)=\tilde{r}\}\cdot \Pr_{i\sim I}\{e\in q_{t(e)}^{\text{k}} \mid r_i(e)=\tilde{r}\} && (150)\\
&\geq \frac{\gamma}{2\beta}\mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{e'\in E}) \mid r_i(e)=\tilde{r}\right], && (151)
\end{aligned}$$

where the last inequality comes from Theorems3–4. By Eqs. (149)–(151), we have

$$\mathbb{E}_{i\sim I}[\text{ALG}(i)] \geq \frac{\gamma}{2\beta}\sum_{e\in E}\sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\cdot\tilde{r}\cdot \mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{e'\in E}) \mid r_i(e)=\tilde{r}\right]. \quad (152)$$

Next, we analyze $\mathbb{E}_{i\sim I}[\text{OPT}^*(i)]$, where we have

$$\begin{aligned}
\mathbb{E}_{i\sim I}[\text{OPT}^*(i)] &= \sum_{e\in E} \mathbb{E}_{i\sim I}[\text{OPT}_e^*(i)] && (153)\\
&= \sum_{e\in E} \sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\mathbb{E}_{i\sim I}[\text{OPT}_e^*(i) \mid r_i(e)=\tilde{r}] && (154)\\
&= \sum_{e\in E} \sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\mathbb{E}_{i\sim I}[r_i(e)\cdot x_e^*(\{r_i(e')\}_{e'\in E}) \mid r_i(e)=\tilde{r}] && (155)\\
&= \sum_{e\in E} \sum_{\tilde{r}\in \text{supp}(\mathcal{R}(e))} \Pr\{r(e)=\tilde{r}\}\cdot\tilde{r}\cdot \mathbb{E}_{i\sim I}[x_e^*(\{r_i(e')\}_{e'\in E}) \mid r_i(e)=\tilde{r}]. && (156)
\end{aligned}$$

By Lemma 6, inequality (152), and Eq. (156), the competitive ratio of OCKA on OCGMK is

$$\frac{\mathbb{E}_{i\sim I}[\text{ALG}(i)]}{\mathbb{E}_{i\sim I}[\text{OPT}(i)]} \geq \frac{\mathbb{E}_{i\sim I}[\text{ALG}(i)]}{\mathbb{E}_{i\sim I}[\text{OPT}^*(i)]} \geq \frac{\gamma}{2\beta} = \alpha. \quad (157)$$

Theorem 5 characterizes the effectiveness of our new OCRS framework, which is designed to counteract the coupled effect of multi-capacity vertices and the knapsack constraint. □

Thus, the competitive ratio of OCKA is approximately $0.159/\beta$. Our algorithm also works when the knapsack constraint is not imposed, in which case we remove the computation of the knapsack-friendly set q_t^{k} and 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 6.

Theorem 6. *(OCKA Competitive Ratio without Knapsack Constraint, Reproduced from Theorem 2 of the Main Paper) When the knapsack constraint is not imposed, the competitive ratio that OCKA achieves on OCGMK is $\alpha' = \frac{1}{2}$.*

Proof. **Overview:** The proof of this theorem is similar to that of Theorem 5, except that we focus only on the capacity-friendly set q_t^{c}.

Detailed Proof: In this case, by the construction of LP $\mathbf{P}^*$, $\mathbb{E}_{i\sim I}[\mathrm{OPT}^*(i)]$ is still an upper bound of the average performance of the offline optimal algorithm, so Lemma 6 holds. Furthermore Eq. (156) still holds. Then, we analyze $\mathbb{E}_{i\sim I}[\mathrm{ALG}(i)]$. First, Eq. (149) still holds. Next, when the knapsack constraint is not imposed, OCKA will accept every edge in the capacity-friendly set q_t^{c}. As a result,

$$\Pr_{i\sim I}\{e \in Q_{t(e)} \mid r_i(e) = \tilde{r}\} = \Pr_{i\sim I}\{e \in q^{\mathrm{c}}_{t(e)} \mid r_i(e) = \tilde{r}\} \tag{158}$$

$$\geq \frac{1}{2}\mathbb{E}_{i\sim I}\left[x_e^*(\{r_i(e')\}_{e'\in E}) \mid r_i(e) = \tilde{r}\right].. \tag{159}$$

Finally, by Lemma 6, Eq (156), and inequality (159), the competitive ratio of OCKA becomes

$$\frac{\mathbb{E}_{i\sim I}[\mathrm{ALG}(i)]}{\mathbb{E}_{i\sim I}[\mathrm{OPT}(i)]} \geq \frac{\mathbb{E}_{i\sim I}[\mathrm{ALG}(i)]}{\mathbb{E}_{i\sim I}[\mathrm{OPT}^*(i)]} \geq \frac{1}{2} = \alpha'. \tag{160}$$

□

We note that the competitive ratio achieved in (Ezra et al. 2020) for the original OGM is also $\frac{1}{2}$. This result shows that OCKA achieves the same performance on the more challenging multi-capacity online general matching problem as the previous best result in the single-capacity setting. This improvement is enabled by considering the Carathéodory set in OCKA and by the novel design of the probability $\phi_u(\sigma(t))$ to balance between the remaining capacity and the expected capacity usage of each vertex.

In Appendix F, we further provide numerical results to study the impact of various system parameters on the competitive ratio of OCKA.

F Numerical Study on Competitive Ratio

In Figure 4 we show the ratio between the performance $\mathbb{E}[\mathrm{ALG}]$ of the OCKA algorithm and the expected offline optimal result $\mathbb{E}[\mathrm{OPT}]$ along with the derived competitive ratio.

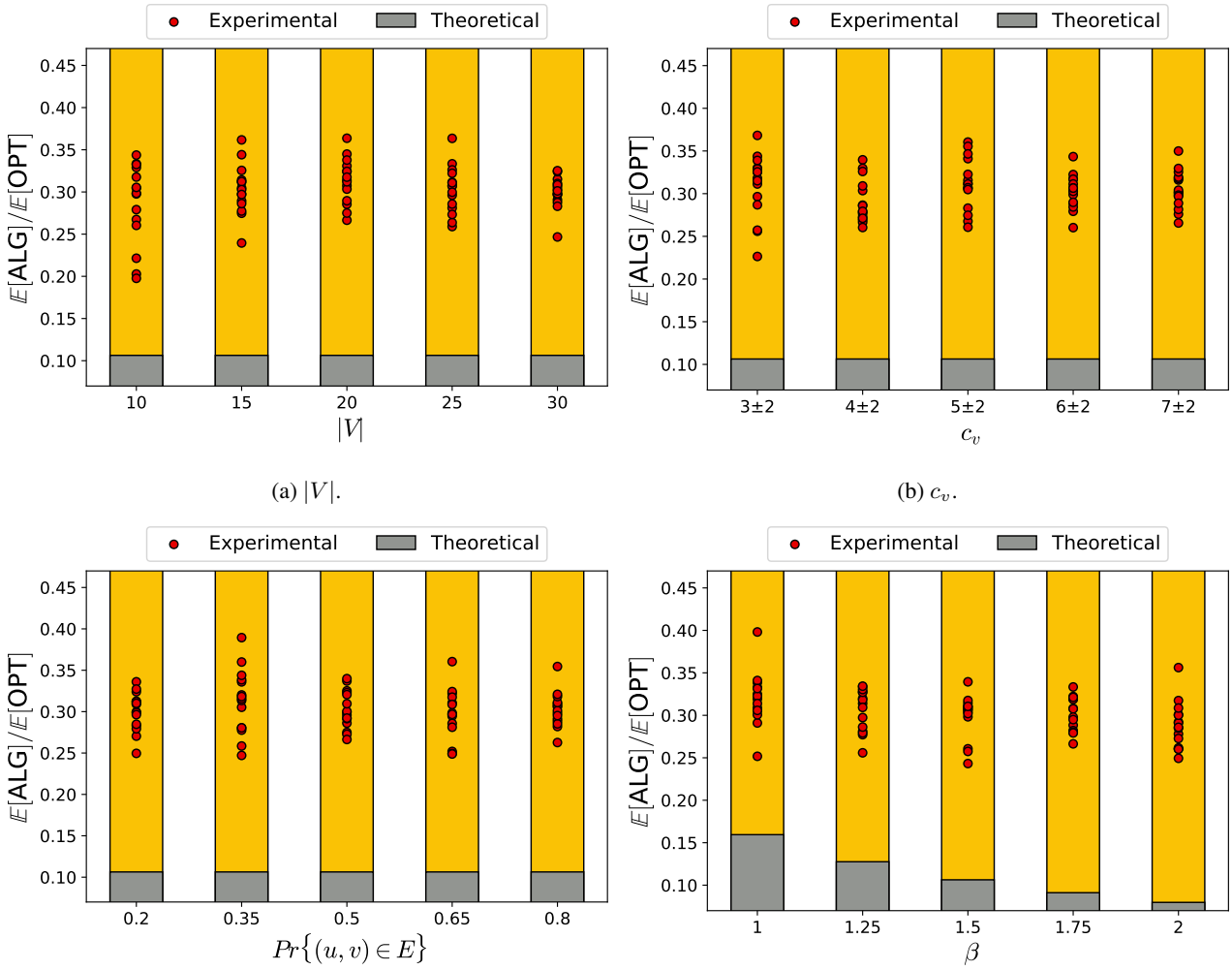
F.1 Experiment Setting

In each simulation round, we generate one problem instance and estimate the expected offline optimal result $\mathbb{E}[\mathrm{OPT}]$ by 500 rounds of Monte Carlo simulation. Each problem instance is generated from a set of parameters. These parameters include the number of vertices $|V|$, the range of the vertex capacity, the probability $\Pr\{(u,v) \in E\}$ that there is an edge between two vertices u and v, the cost budget K, and the ratio β between the expected overall cost of all edges and the cost budget. We generate the edges E, the vertex capacity $\{c_v\}_v$, the distributions of edge reward $\mathcal{R}(e)$, and the edge cost $d(e)$ for each point (problem instance) from these parameters. We also keep a set of default values for these parameters, with $|V| = 20$, the range of vertex capacity $[3, 7]$, $\Pr\{(u,v) \in E\} = 0.5$, and $\beta = 1.5$. The arriving order of vertices is shuffled with the `shuffle` function in the Python library. We set the cost budget K to $K = 1$, so that the edge cost $d(e)$ is normalized with respect to the cost budget K, and β equals the overall cost of all edges.

For each problem instance, we establish an edge (u, v) between vertices u and v with probability $\Pr\{(u,v) \in E\}$ independently. We uniformly randomly select a number from the capacity range for each vertex as its capacity. The reward distribution $\mathcal{R}(e)$ of each edge has three possible values, each generated from the uniform distribution over $[0, 1]$. The probabilities assigned to these three values are also generated from the uniform distribution over $[0, 1]$, and then normalized to ensure that their sum equals 1. For the edge cost $d(e)$, we first generate each $d(e)$ from the uniform distribution over $\{1, 2, \ldots, 10\}$ and we then normalize every $d(e)$ so that $\sum_e d(e) = \beta$.

In each realization generated from a problem instance, the edge reward $r(e)$ is sampled from the reward distribution $\mathcal{R}(e)$. We generate 20 realizations of each problem instance to estimate the average performance of $\mathbb{E}[\mathrm{ALG}]$ of the OCKA algorithm.

We plot the ratio $\frac{\mathbb{E}[\mathrm{ALG}]}{\mathbb{E}[\mathrm{OPT}]}$ as one red point in Figure 4 in every problem instance and every setting, shown together with derived competitive ratio represented by a gray bar. We plot 15 points for each bar, corresponding to the 15 problem instances under that setting.



(a) $|V|$.

(b) c_v.

(c) $\Pr\{(u,v) \in E\}$.

(d) β.

Figure 4: Numerical study on competitive ratio.

F.2 Results

In Figure 4a, we investigate the competitive ratio of the OCKA algorithm under different numbers of vertices $|V|$. We let the number of vertices $|V|$ increase from 10 to 30. In Figure 4b, we investigate the competitive ratio of the OCKA algorithm under different ranges of vertex capacity. We let the range of vertex capacity change from $[1, 5]$ to $[5, 9]$. In Figure 4c, we investigate the competitive ratio of the OCKA algorithm under different numbers of edges. We let the probability $\Pr\{(u, v) \in E\}$ increase from 0.2 to 0.8. In Figure 4d, we investigate the competitive ratio of the OCKA algorithm under different ratios β. We let β increase from 1 to 2.

We observe that the lowest red dot in each bar, representing the worst-case performance of the OCKA algorithm, is closer to the derived competitive ratio for smaller β or smaller $|V|$, suggesting that the derived bound is tighter for those cases. Furthermore, we notice that the highest red dots in each bar approach 0.4, which is close to the competitive ratio of the case without the knapsack constraint. These red dots represent the problem instances with certain edge costs that make it easy to manage the cost budget; thus, the decision of OCKA is dominated by the Online Assignment phase in these problem instances.

G Trace-Driven Evaluation

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

G.1 User Pairing in Dating Application

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 dataset (Fisman et al. 2006). The SpeedDating 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 restructured into the HFM dataset for training recommendation models. The HFM dataset contains 123 users and over 1000 interaction records, along with critical features (e.g., age, gender, religion, user biographies, and dating preferences) that significantly influence user decisions in dating contexts.

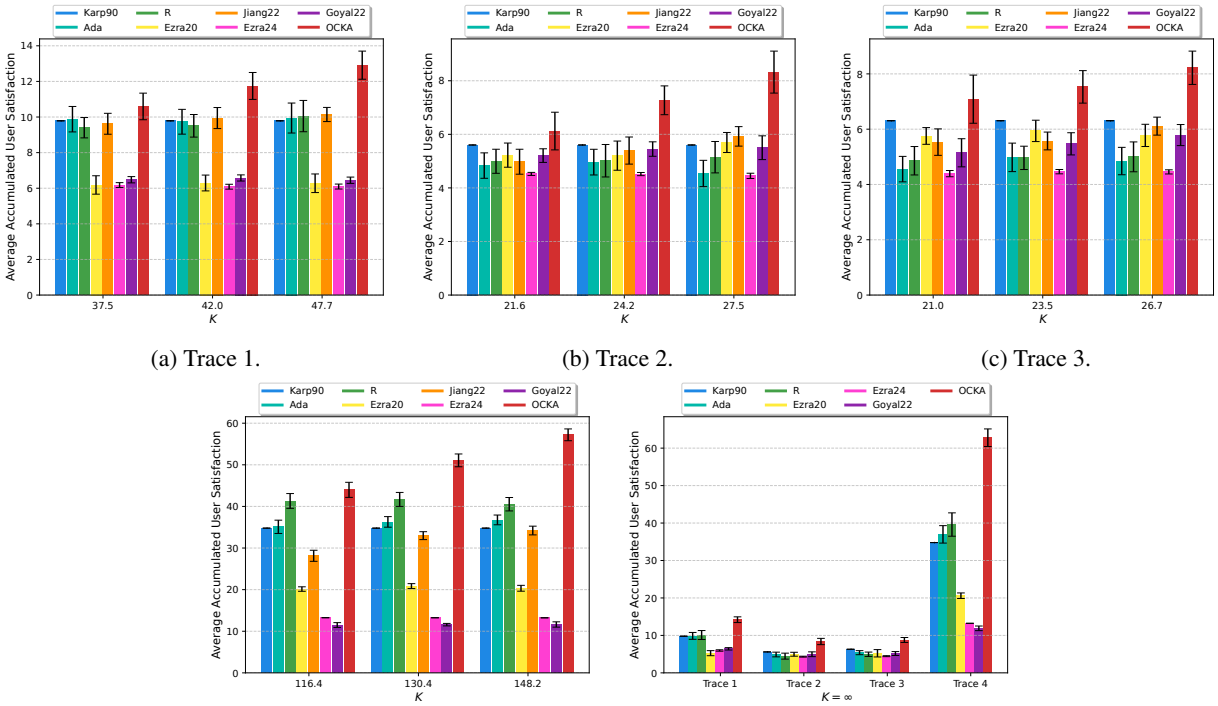
In this dating scenario, users (vertices) arrive one by one and need to be paired with other eligible (edges) users. Pairing two users yields user satisfaction (edge reward) and incurs a legal risk for the service provider (edge cost). The accumulated risk incurred by pairing users is budgeted as the 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 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. For example, dating platforms such as Tinder frequently launch time-limited promotional events to engage existing users (Dellinger 2019), where the platform needs to access relevant user information in advance. The exact satisfaction from pairing two users is obtained via ChatGPT-4 API (OpenAI 2023) inference once both have arrived.

Trace Parameter We choose 4 traces from the HFM dataset, labeled as Traces 1–4. Each of Traces 1–3 contains 30 users, and Trace 4 contains 100 users. For each trace, we use the indices of users in the dataset as their arriving order σ. Before users arrive, we use the ChatGPT-4 API (OpenAI 2023) to estimate the perceived legal risk $d(e)$ associated with pairing two users, based on their biographies, dating preferences, and personal information (including gender, age, self-reported personality traits, etc.), all of which are provided in the dataset. For each trace, the overall perceived legal risk $\sum_{e\in E} d(e)$ incurred by matching all eligible user pairs is 52.5 (Trace 1), 30.3 (Trace 2), 29.4 (Trace 3), or 163.0 (Trace 4). At the same time, we also employ a Multi-Layer Perceptron (MLP) to estimate the satisfaction of pairing any two users as a probability distribution $\mathcal{R}(e)$, using only their personal information. Upon each user's arrival, we further use another MLP to compute the actual user satisfaction $r(e)$ yielded by pairing them with each previously arrived user, leveraging both users' biographies, dating preferences, and personal information. The calculation of $d(e)$, $\mathcal{R}(e)$, and $r(e)$ aligns with the assumptions in OCGMK, where the edge cost and the reward distribution are known prior to vertex arrivals, while the exact reward of the edge connecting two vertices is revealed once both vertices have arrived.

G.2 Benchmarks

We compare the performance and runtime of OCKA against the following 7 benchmarks:

1. Karp90 (Karp, Vazirani, and Vazirani 1990): The system accepts an edge if neither of the vertices connected by this edge has depleted its capacity and the remaining cost budget is larger than the cost of this edge.
2. Adaptive (Ada): When vertex $\sigma(t)$ arrives in time slot t, Ada computes a probability $\rho(u, \sigma(t))$ for accepting each edge $(u, \sigma(t))$ revealed in $E_\sigma^-(\sigma(t))$. The probability $\rho(u, \sigma(t))$ consists of three terms. The first term, ρ_u, is calculated as the ratio of the remaining capacity of vertex u to its total capacity c_u. The second term, $\rho_{\sigma(t)}$, is the ratio of the remaining capacity of vertex $\sigma(t)$ to its total capacity $c_{\sigma(t)}$. The third term, ρ_{k}, is the ratio of the remaining cost budget to the initial cost budget K. Ada then accepts the edge $(u, \sigma(t))$ with probability $\rho = \rho_u \times \rho_{\sigma(t)} \times \rho_{\mathrm{k}}$.
3. Random (R): The system accepts each edge (assign the vertices connected by an edge to each other) with a probability of $1/2$.
4. Ezra20 (Ezra et al. 2020): The system employs a naive extension of the matching algorithm designed for single-capacity OGM (Ezra et al. 2020). When vertex v arrives in time slot t, the system decomposes v into c_v virtual vertices $\{v'_1, v'_2, \ldots, v'_{c_v}\}$, each with a capacity of one and the same set of neighbors as v i.e., $N(v'_c) = N(v)$, $\forall c \in [c_v]$. For each $c \in [c_v]$, the reward and cost of edge (u, v'_c) are the same as those of edge (u, v). The system then performs c_v iterations to make decisions for each virtual vertex v'_c. The virtual vertices $\{v'_c\}$ and their edges form a new graph G' with other vertices (except v) and other edges from G. To make the decision for the vertex v'_c, the system first samples the rewards of all edges in $E \setminus E_\sigma^-(v)$. It then solves an Integer Linear Programming version of $\mathbf{P}^*$ in graph G', where the decision space is restricted to integers 0 and 1. This process yields an optimal solution $\mathbf{x}^*$. If, in this optimal solution, the value associated with edge (u, v'_c) is 1, and the edge (u, v) is in $E_\sigma^-(v)$, the system assigns v'_c to u. If u has already been assigned to another virtual vertex $v'_{c'}$ (where $c' \neq c$), then the system cannot assign v'_c to u. After the c_v-th iteration, for each edge $(u, v) \in E_\sigma^-(v)$, if u has been assigned to any virtual vertex v'_c, then the system will accept the edge (u, v).
5. Jiang22 (Jiang, Ma, and Zhang 2022): The system simply accepts all edges in the knapsack-friendly set. When vertex v arrives, the system goes through each edge $(u, v) \in E_\sigma^-(v)$ to decide whether to assign u to v. The system keeps the distribution $\tilde{X}_{j-1}$ as the distribution of the consumed cost budget before it makes the decision for the j-th edge, and then



(a) Trace 1. (b) Trace 2. (c) Trace 3.

(d) Trace 4. (e) $K = \infty$ (w/o knapsack).

Figure 5: (Reproduced from Figure 1 in the main paper)
Performance comparison on traces with varying risk appetite (i.e., edge cost budget) K.

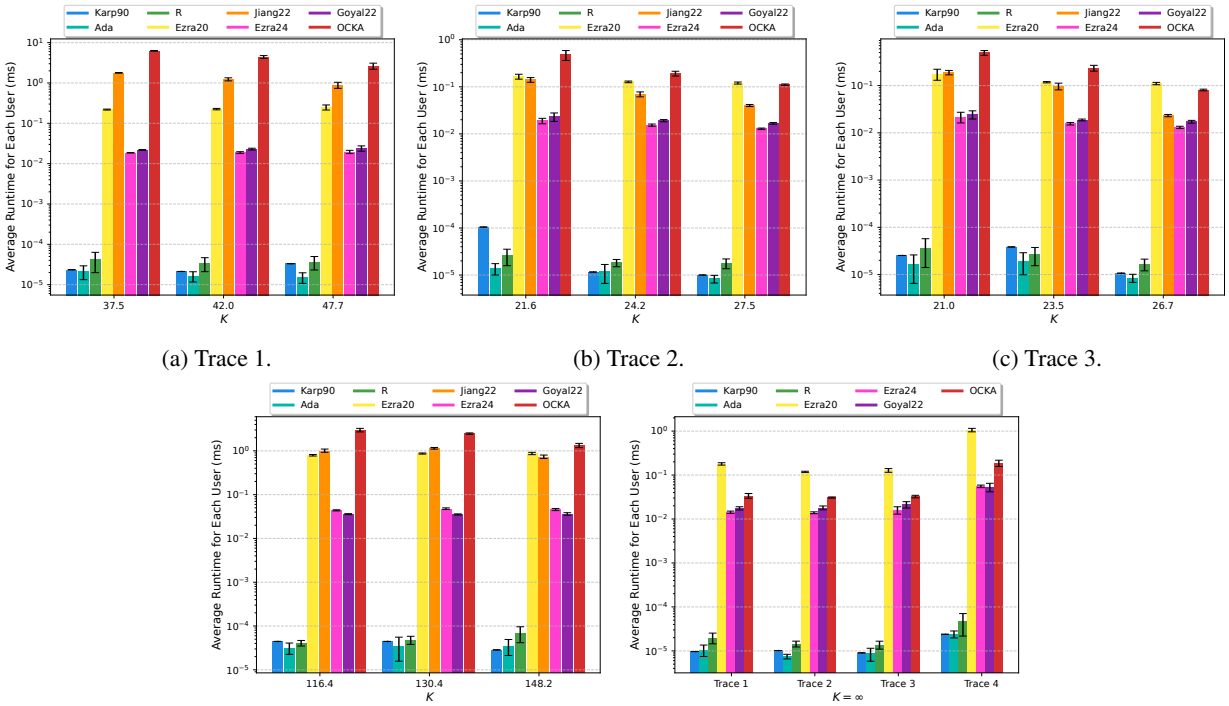
calculates a safe range and probability ψ_j using the same approach as OCKA. If the budget consumption is in the safe range $(\theta_j^-, \theta_j^+]$, the system accepts edge (u, v) with probability γ/β. If the budget consumption is exactly θ_j^-, the system accepts edge (u, v) with probability ψ_j. Otherwise, the system will not accept edge (u, v). This benchmark indicates the performance of a solution that solely focuses on the knapsack constraint in OCGMK, such as (Jiang, Ma, and Zhang 2022), while ignoring the joint evolution of capacity usage and budget consumption.

6. Ezra24 (Ezra et al. 2024): This benchmark ignores the first $\lfloor |V|/2 \rfloor$ vertices. Then, upon the arrival of vertex $\sigma(t)$, it removes a random vertex from the arrived vertices when the total number of arrived vertices is odd. This benchmark finds a maximum weighted matching among the arrived vertices, and accept the match of $\sigma(t)$ if the other vertex has unused capacity.
7. Goyal22 (Goyal 2022): This benchmark ignores the first $\lfloor 6|V|/17 \rfloor$ vertices. Then, upon the arrival of vertex $\sigma(t)$, it removes a randomly selected vertex from the set of previously arrived vertices if their total count is odd. It computes a maximum weight matching among the arrived vertices and accepts the match involving $\sigma(t)$ if the other vertex has remaining capacity. If $\sigma(t)$ is not included in the maximum weight matching and there are more than $|V| + 1 - t$ unmatched arrived vertices, this benchmark randomly selects one of them and matches it with $\sigma(t)$, if there is an edge connecting the two.

The accuracy of the OCKA, Ada, R, Ezra20, Jiang22, Ezra24, and Goyal22 algorithms is averaged over 20 runs, as they are randomized, while a single run is performed for the deterministic algorithm Karp90. In addition, we compute the standard errors of each algorithm's performance and runtime using Student's t-distribution at a 95% confidence level, with all algorithm runs within each experimental setting serving as the sample space (Ross 2020).

G.3 Experiment Setting

Evaluation on Original Traces We deploy OCKA and benchmarks on a MacBook Air (2022) equipped with an M2 chip to process user pairing requests collected from the HFM dataset. In Figs. 5 and 6, we compare the average accumulated user satisfaction and runtime, respectively, of OCKA against benchmarks on each trace. In Figs. 5a–5d and Figs. 6a–6d, we apply OCKA and benchmarks to each trace with different risk appetites. We evaluate OCKA on each trace with $\beta = \frac{\sum d(e)}{K}$ set to 1.4, 1.25, and 1.1, corresponding to risk appetites of 37.5, 42.0, and 47.7 for Trace 1; 21.6, 24.2, and 27.5 for Trace 2; 21.0, 23.5, and 26.7 for Trace 3; and 116.4, 130.4, and 148.2 for Trace 4, respectively. In Figs. 5e and 6e, we compare the performance and runtime of OCKA against benchmarks on the 4 traces with the knapsack constraint not imposed.



(a) Trace 1. (b) Trace 2. (c) Trace 3.

(d) Trace 4. (e) $K = \infty$ (w/o knapsack).

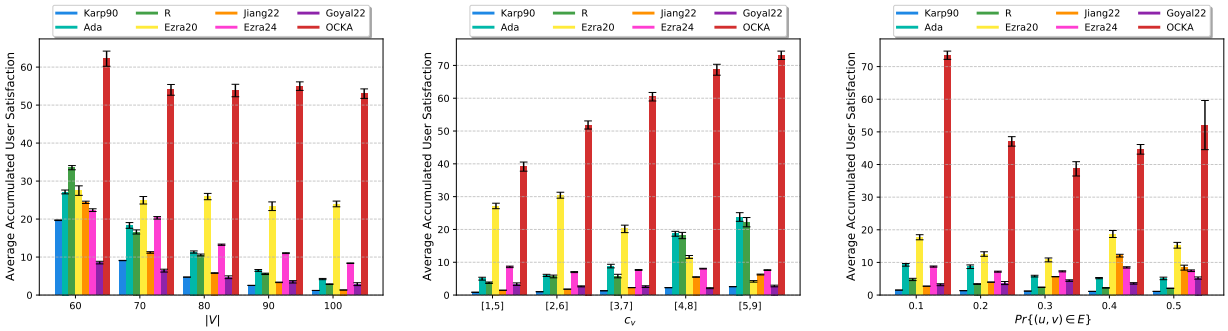
Figure 6: (Reproduced from Figure 2 in the main paper)
Runtime comparison on traces with varying risk appetite (i.e., edge cost budget) K.

Trace-Driven Simulation To more comprehensively evaluate the performance and runtime of OCKA under the impact of varying number of vertices, number of edges, and vertex capacities, we conduct additional trace-driven simulations using data characteristics extracted from Trace 4. In Figs. 7–8, we also compare the performance of OCKA against benchmarks in simulation experiments based on Trace 4. In each simulation round, we generate 20 realizations of each problem instance to estimate the average performance of each algorithm on that problem instance, and we generate 15 problem instances for each set of parameters to estimate the average performance of each algorithm on that set of parameters. The average runtime per time slot of each algorithm on each set of parameters is also recorded when recording their performance.

The parameters for generating problem instances include the number of vertices $|V|$, the range of the vertex capacity, and the probability $\Pr\{(u, v) \in E\}$ that there is an edge between two vertices u and v. We keep a set of default values for these parameters, which are extracted from Trace 4. The default parameters include the vertices recorded in Trace 4 with $|V| = 100$, the capacity of each vertex in Trace 4, the capacity of each vertex recorded in Trace 4, the edges recorded in Trace 4, and the edge cost recorded in Trace 4. We set the default value of $\Pr\{(u, v) \in E\}$ as 0. Furthermore, we keep β as 1.3 across all settings and set the edge cost budget K accordingly.

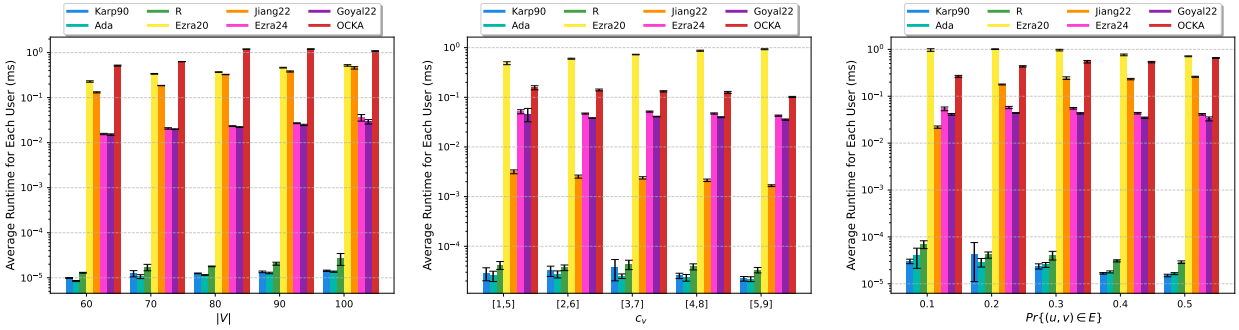
For each problem instance, we construct the graph by randomly selecting $|V|$ vertices from Trace 4 and retaining the edges recorded in Trace 4. Then, for each pair of vertices not connected by an edge in Trace 4, we establish an edge between them with probability $\Pr\{(u, v) \in E\}$. Since we want to investigate the performance of algorithms under an "adversarial" environment, we let an edge that arrives later have a higher reward but with a lower cost. For a cleaner presentation, we use $\tau(e)$ to denote the time slot t that the reward of e is revealed, so that $e \in E_\sigma^-(\sigma(\tau(e)))$, and use $\Delta\tau(e)$ for the interval between the arrival time of the endpoints of an edge. The reward distribution $\mathcal{R}(e)$ of each edge has three possible values, each generated from the uniform distribution over $[0, 1]$, then multiplied with $2.5 \exp\left(\frac{\Delta\tau(e)}{12.5|V|}\right)$. The probabilities assigned to these three values are also generated from the uniform distribution over $[0, 1]$, and then normalized to ensure that their sum equals 1. The cost of each extra edge is drawn uniformly at random from the range between the highest and lowest costs of the edges originally recorded in Trace 4. For Figs. 7b and 8b, we randomly select a number from the capacity range for each vertex as its capacity, with each number having an equal probability of being chosen. For settings in the other figures, the capacity of each vertex is the same as recorded in Trace 4.

In Figs. 7a and 8a, we compare the performance of OCKA against benchmarks under different numbers of vertices $|V|$. We let the number of vertices $|V|$ increase from 60 to 100. In Figs. 7b and 8b, we compare the performance of OCKA against benchmarks under different ranges of vertex capacity. We let the range of vertex capacity change from $[1, 5]$ to $[5, 9]$. In Figs. 7c



(a) $|V|$. (b) c_v. (c) $\Pr\{(u, v) \in E\}$.

Figure 7: Performance comparison in simulation, for varying number of vertices $|V|$, number of edges c_v, and probability of extra edges $\Pr\{(u, v) \in E\}$.



(a) $|V|$. (b) c_v. (c) $\Pr\{(u, v) \in E\}$.

Figure 8: Runtime comparison in simulation, for varying number of vertices $|V|$, number of edges c_v, and probability of extra edges $\Pr\{(u, v) \in E\}$.

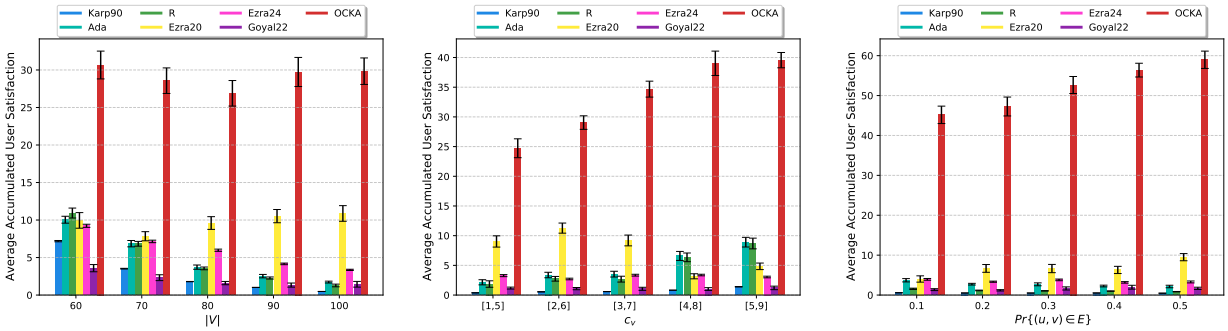
and 8c, we compare the performance of OCKA against benchmarks under different numbers of extra edges. We let the probability $\Pr\{(u, v) \in E\}$ increase from 0.1 to 0.5.

Trace-Driven Simulation without Knapsack Constraint We also compare the performance of OCKA against benchmarks in Figs. 9–10 with the knapsack constraint not imposed, in order to illustrate the excellent capability of OCKA to accommodate capacitated vertices in OCGMK. We consider every benchmark except Jiang22. The experiments in Figs. 9–10 have the same parameter settings as in Figs. 7–8, except that the knapsack constraint is not considered in this set of experiments, meaning neither the knapsack cost of edges nor the decision maker's cost budget is taken into account.

G.4 Performance in User Satisfaction

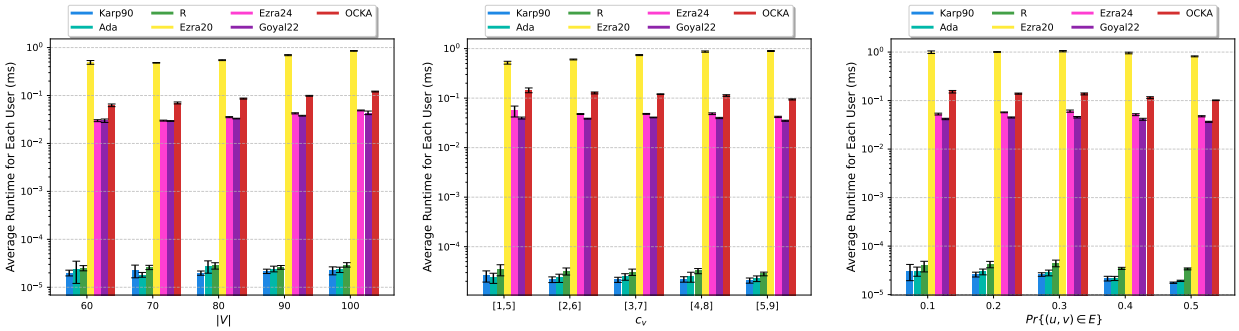
In most settings, OCKA substantially outperforms all benchmarks. We observe that the advantage of OCKA rises when the risk appetite increases. This is because when the knapsack budget is very limited, all algorithms face a highly constrained decision space, leaving little room for OCKA to gain a substantial 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. We also have some observations on the performance of each of the benchmarks:

1. Karp90, Random, and Adaptive: We observe that the algorithms Greedy and Random are not aware of the vertex capacities constraint or the knapsack constraint, so they both deplete vertex capacities and the cost budget early. The algorithm Adaptive is not aware of the vertex capacities, so it never conserves them for a potentially better future match. Therefore, all three of these algorithms perform poorly.
2. Ezra20: The Ezra20 benchmark is outperformed by OCKA in all settings. This is primarily because it naively treats the multi-matching ability of a vertex as separate vertices. This approach often results in numerous invalid assignments by repeatedly assigning two vertices to each other. Additionally, when accepting edges, it fails to consider the capacity of each vertex, further degrading its performance. The performance of Ezra20 suffers also because it ignores the knapsack constraint, so it tends to deplete the cost budget early while missing the opportunity to secure a potentially better future match.



(a) $|V|$. (b) c_v. (c) $\Pr\{(u, v) \in E\}$.

Figure 9: Performance comparison in simulation without knapsack constraint.



(a) $|V|$. (b) c_v. (c) $\Pr\{(u, v) \in E\}$.

Figure 10: Runtime comparison in simulation without knapsack constraint.

3. Jiang22: The Jiang22 benchmark is outperformed by OCKA because it does not consider the capacity of each vertex. In OCGMK, the consumptions of vertex capacity and cost budget are closely intertwined. The ignorance of Jiang22 in capacity usage often results in premature depletion of vertex capacities, leading to missed opportunities for making higher-reward assignments.
4. Ezra24 and Goyal22: Ezra24 and Goyal22 were designed for OGM under the secretary matching branch, where the edge reward is unknown and vertices arrive under the uniformly random order (Ezra et al. 2024; Goyal 2022). Ezra24 and Goyal22 perform worse than OCKA because they do not utilize the prior information of the probability distribution of edge rewards. Moreover, they do not consider multi-capacity vertices or the knapsack constraint in OCGMK.

When the knapsack constraint is removed, OCKA outperforms all benchmarks by an even larger advantage. We also observe that the performance gap between OCKA and Ezra20 becomes more significant. Specifically, Ezra20 applies a single-capacity matching solution on the capacitated vertices using "capacity decomposition", treating a vertex's multi-matching ability as separate vertices. This approach overlooks the interdependencies among the capacities of a capacitated vertex, leading to invalid decisions, such as repeatedly matching the same pair of vertices, which eventually results in degraded performance.

G.5 Algorithm Runtime Comparison

Our main observation is that the average time required for OCKA to make decisions in each time slot is under 1 second in most settings, which is small compared with the task arrival interval in many online matching problems. Additionally, OCKA's runtime is comparable to that of Jiang22 and sometimes shorter than that of Ezra20. Given the excellent performance of OCKA, its minor increase in runtime further demonstrates OCKA's efficiency in handling multi-capacity vertices and the knapsack constraint in the online general matching problem.

Furthermore, we observe that the average runtime of the algorithms Karp90, Adaptive, and Random is significantly lower than that of the other algorithms, reflecting their straightforward approach but also aligning with their limited overall performance. The runtime of Ezra24 and Goyal22 is higher, since, upon the arrival of each vertex, they need to perform maximum weight matching, which is a time-consuming Integer Programming optimization problem.

The results in Fig. 8 and Fig. 10 also indicate that the number of vertices and edges significantly affects the runtime of algorithms Ezra20, Jiang22, and OCKA. This is because, in each time slot t, these three algorithms evaluate all edges incident on

vertex $\sigma(t)$ and its previously arrived neighbors. As the number of edges in the graph increases, the algorithms need to evaluate more edges per time slot. Similarly, an increase in the number of vertices results in more edges, leading to a corresponding rise in runtime.

Finally, the results in Figs. 6e, 8b, and 10 show that, when vertices have high capacities or the knapsack constraint is not imposed, the runtime of Ezra20 exceeds that of OCKA. This is because Ezra20 treats multi-capacity vertices as multiple single-capacity vertices and makes decisions for them separately. As a result, the runtime of Ezra20 grows with the vertex capacity. When the knapsack constraint is not imposed, Ezra20 inevitably takes more time than OCKA for each vertex. Even when the knapsack constraint is present, if the vertex capacities are large, the runtime required by Ezra20 for each vertex could still exceed the runtime of OCKA.

References

Alaei, S.; Hajiaghayi, M.; and Liaghat, V. 2012. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the ACM Conference on Electronic Commerce*, 18–35.

Andani, I. G. A.; Puello, L. L. P.; and Geurs, K. 2021. Modelling effects of changes in travel time and costs of toll road usage on choices for residential location, route and travel mode across population segments in the Jakarta-Bandung region, Indonesia. *Transportation Research Part A: Policy and Practice*, 145: 81–102.

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.

Borodin, A.; and MacRury, C. 2025. Online bipartite matching in the probe-commit model. *Mathematical Programming*, 1–54.

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

Brubach, B.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2016. New algorithms, better bounds, and a novel model for online stochastic matching. In *Proceedings of the Annual European Symposium on Algorithms*.

Chawla, S.; Christou, D.; Dang, T.; Huang, Z.; Kehne, G.; and Rezvan, R. 2025. A multi-dimensional online contention resolution scheme for revenue maximization. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 1720–1757.

Chekuri, C.; Vondrák, J.; and Zenklusen, R. 2011. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 783–792.

Chen, Z.; Huang, Z.; Li, D.; and Tang, Z. G. 2025. Prophet secretary and matching: the significance of the largest item. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 1371–1401.

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.; Gravin, N.; and Lucier, B. 2015. Combinatorial auctions via posted prices. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithms*, 123–135. Society for Industrial and Applied Mathematics.

Feldman, M.; Svensson, O.; and Zenklusen, R. 2016. Online contention resolution schemes. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 1014–1033.

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.

Fu, H.; Lu, P.; Tang, Z. G.; Turkieltaub, A.; Wu, H.; Wu, J.; and Zhang, Q. 2022. Oblivious online contention resolution schemes. In *Proceedings of the SIAM Symposium on Simplicity in Algorithms*, 268–278.

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.

Naor, J.; Srinivasan, A.; and Wajc, D. 2025. Online dependent rounding schemes for bipartite matchings, with applications. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 3116–3154.

Nesterov, Y.; and Nemirovski, A. 1994. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics.

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

Papastavrou, J. D.; Rajagopalan, S.; and Kleywegt, A. J. 1996. The dynamic and stochastic knapsack problem with deadlines. *Management Science*, 42(12): 1706–1718.

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.

Ross, S. M. 2020. *Introduction to Probability and Statistics for Engineers and Scientists*. Academic press.

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

Sumita, H.; Ito, S.; Takemura, K.; Hatano, D.; Fukunaga, T.; Kakimura, N.; and Kawarabayashi, K.-i. 2022. Online task assignment problems with reusable resources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5199–5207.

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.

Sun, B.; Zeynali, A.; Li, T.; Hajiesmaili, M.; Wierman, A.; and Tsang, D. H. 2020. Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, volume 4, 1–32.

Wang, H.; Yan, Z.; and Bei, X. 2022. A nonasymptotic analysis for re-solving heuristic in online matching. *Production and Operations Management*, 31(8): 3096–3124.

Wang, Y.; and Wong, S. C.-w. 2015. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, 1070–1081.

Xu, P. 2024. Parameter-dependent competitive analysis for online capacitated coverage maximization through boostings and attenuations. In *Proceedings of the International Conference on Machine Learning*, volume 235, 54831–54851.

Yang, L.; Zeynali, A.; Hajiesmaili, M. H.; Sitaraman, R. K.; and Towsley, D. 2021. Competitive algorithms for online multi-dimensional knapsack problems. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, volume 5, 1–30.

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.