تمرین اول

نام درس: سیستم های چندعاملی

استاد درس: دکتر ناصر مزینی

نام: محمد حقیقت

شماره دانشجویی: 403722042

گرایش: هوش مصنوعی

دانشکده: مهندسی کامپیوتر

نیم سال دوم 1403-1404

A) انواع ارزش در حراجی‌ها

I.

در حراجی‌ها، نحوه ارزش‌گذاری کالا توسط شرکت‌کنندگان می‌تواند متفاوت باشد و این تفاوت مبنای تقسیم‌بندی انواع ارزش را تشکیل می‌دهد:

ارزش مشترک (Common Value):

در این نوع حراجی، ارزش واقعی کالا برای همه شرکت‌کنندگان یکسان است، اما هیچ‌کس از این ارزش دقیقا اطلاع ندارد. به عبارت دیگر، کالا دارای یک "ارزش واقعی" واحد است که اگر همه اطلاعات در دسترس بود، همه بر سر آن توافق می‌کردند. با این حال، هر شرکت‌کننده بر اساس اطلاعات ناقص و حدس و گمان خود، برآورد متفاوتی از این ارزش دارد. هدف اصلی شرکت‌کنندگان این است که با توجه به اطلاعاتی که در اختیار دارند (که معمولا به آن "سیگنال" می‌گویند)، بهترین تخمین را از ارزش واقعی بزنند.

ارزش خصوصی مستقل (Independent Private Value):

در این مدل، ارزش کالا برای هر شرکت‌کننده منحصر به فرد و شخصی است. این ارزش فقط به خود فرد مربوط می‌شود و از ارزش‌گذاری دیگران تأثیر نمی‌پذیرد. هر شرکت‌کننده دقیقا می‌داند که کالا برای او چقدر ارزش دارد و این اطلاعات، خصوصی و محرمانه است. تصمیم‌گیری برای پیشنهاد قیمت صرفا بر اساس این ارزش شخصی و استراتژی فردی انجام می‌شود.

به طور خلاصه، تفاوت اصلی بین این دو نوع ارزش در این است که در ارزش مشترک، یک ارزش واقعی و واحد برای کالا وجود دارد که ناشناخته است، در حالی که در ارزش خصوصی مستقل، هر شرکت‌کننده ارزش منحصر به فرد و شناخته شده‌ای برای خود دارد.

II.

مثال برای ارزش مشترک(Common Value):

مزایده بهره‌برداری از یک میدان نفتی یا گازی جدید

فرض کنید دولت یک کشور، مزایده‌ای برای واگذاری حق اکتشاف و تولید از یک میدان نفتی تازه کشف شده برگزار می‌کند. شرکت‌های نفتی مختلفی در این مزایده شرکت می‌کنند.

ارزش مشترک: در واقعیت، میزان نفت یا گاز موجود در این میدان (و در نتیجه، سودآوری نهایی آن) برای همه شرکت‌ها یکسان است؛ یعنی یک "ارزش واقعی" واحد برای این میدان وجود دارد.

عدم قطعیت و تخمین: اما هیچ‌یک از شرکت‌ها دقیقا نمی‌دانند چقدر نفت یا گاز در این میدان وجود دارد. هر شرکت با استفاده از تیم‌های زمین‌شناسی، داده‌های لرزه‌نگاری، و مدل‌های پیش‌بینی خود، تخمینی از حجم ذخایر و هزینه‌های استخراج آن می‌زند. این تخمین‌ها (که اصطلاحا به آن‌ها "سیگنال" می‌گویند) متفاوت خواهد بود.

نفرین برنده(Winner's Curse): در چنین مزایده‌ای، شرکت برنده معمولا شرکتی است که بالاترین پیشنهاد را داده است. اما گاهی اوقات، این شرکت ممکن است همان شرکتی باشد که خوش‌بینانه‌ترین (و شاید نادرست‌ترین) تخمین را از ارزش میدان نفتی زده است. یعنی، آن‌ها ممکن است بیش از ارزش واقعی میدان، برای آن پیشنهاد داده باشند، که به این پدیده "نفرین برنده" می‌گویند.

مثال برای ارزش خصوصی مستقل(Independent Private Value):

حراجی یک کالای عتیقه یا کلکسیونی

فرض کنید یک حراجی برای فروش یک مجسمه بسیار کمیاب برگزار می‌شود. کلکسیونرهای مجسمه از سراسر دنیا در آن شرکت می‌کنند.

ارزش خصوصی مستقل: ارزش این مجسمه برای هر کلکسیونر می‌تواند کاملا شخصی و مستقل باشد.

برای یک کلکسیونر، این مجسمه ممکن است آخرین قطعه‌ای باشد که مجموعه خاص او را کامل می‌کند، بنابراین حاضر است مبلغ بسیار بالایی برای آن بپردازد.

برای کلکسیونر دیگر، این مجسمه ممکن است فقط یک نمونه از هزاران مجسمه کمیاب دیگر باشد و اهمیت کمتری برای او داشته باشد، بنابراین ارزش کمتری برای آن قائل است.

یک نفر ممکن است به دلیل علاقه خاص به تاریخچه آن مجسمه، ارزش بیشتری برایش قائل باشد، در حالی که دیگری صرفا به دنبال ارزش سرمایه‌گذاری آن باشد.

شناخت ارزش شخصی: هر یک از این افراد دقیقا می‌داند که این مجسمه برای او شخصا چقدر ارزش دارد و سقف قیمتی که حاضر است بپردازد چقدر است. قیمت پیشنهادی آن‌ها فقط بر اساس این

ارزش شخصی آن‌ها تعیین می‌شود و نه برآورد آن‌ها از ارزش آن برای دیگران. در این حالت، "نفرین برنده" به معنای مدل ارزش مشترک کمتر اتفاق می‌افتد، زیرا هر کس تا سقف ارزش شخصی خود پیشنهاد می‌دهد و اگر برنده شود، به این معنی است که کالا را به قیمتی خریده که برایش ارزش داشته است.

III.

ارزش‌های همبسته (Correlated Values):

در این سناریو، ارزش کالا برای هر شرکت‌کننده دارای یک بخش خصوصی (مثل ارزش خصوصی مستقل) و یک بخش مشترک (که تحت تأثیر اطلاعات دیگران قرار می‌گیرد) است. به عبارت دیگر، ارزش نهایی کالا برای هر فرد تا حدودی به اطلاعات خصوصی خود او بستگی دارد و تا حدودی نیز تحت تأثیر اطلاعاتی است که دیگران آشکار می‌کنند یا برآوردی که از ارزش کالا دارند. اطلاعاتی که یک شرکت‌کننده دارد، می‌تواند سرنخی در مورد ارزش شخصی دیگران یا ارزش مشترک کالا باشد.

تفاوت با دو حالت قبلی:

تفاوت با ارزش مشترک: در ارزش مشترک، فقط یک "ارزش واقعی" واحد و ناشناخته وجود دارد و همه برآوردی از آن می‌زنند. در ارزش‌های همبسته، علاوه بر بخش مشترک، یک بخش خصوصی هم وجود دارد.

تفاوت با ارزش خصوصی مستقل: در ارزش خصوصی مستقل، ارزش هر فرد کاملا مستقل و فقط برای خودش است و اطلاعات دیگران تأثیری بر آن ندارد. اما در ارزش‌های همبسته، اطلاعات دیگران (مانند پیشنهادهای آن‌ها یا سیگنال‌هایی که در طول حراجی می‌دهند) می‌تواند بر برآورد هر فرد از ارزش نهایی کالا برای خودش تأثیر بگذارد. به نوعی، ارزش شخصی من ممکن است با ارزش شخصی شما ارتباط داشته باشد.

مثال واقعی: حراجی حقوق فرکانس رادیویی برای شرکت‌های مخابراتی.

تصور کنید دولت مزایده‌ای برای واگذاری بلوک‌های فرکانس رادیویی (مثلا برای خدمات 5G) به شرکت‌های مخابراتی برگزار می‌کند.

جزء خصوصی (Independent Private Value): هر شرکت مخابراتی، بر اساس زیرساخت‌های فعلی، تعداد مشتریان موجود در مناطق مختلف، برنامه‌های توسعه آینده، و استراتژی کلی کسب‌وکار خود، یک ارزش خصوصی برای هر بلوک فرکانس قائل است. برای مثال، شرکتی که در یک منطقه خاص

پوشش ضعیفی دارد، ممکن است برای فرکانس‌های آن منطقه ارزش بالاتری قائل باشد تا بتواند پوشش خود را بهبود بخشد.

جزء مشترک / همبسته (Common/Correlated Value):

ارزش بازار مشترک: موفقیت و ارزش نهایی این فرکانس‌ها تا حد زیادی به تعداد شرکت‌کنندگان دیگر در بازار و میزان رقابت بستگی دارد. اگر شرکت‌های رقیب قوی‌تر و با پول بیشتری وارد حراجی شوند، این فرکانس‌ها به طور کلی برای همه با ارزش‌تر تلقی می‌شوند (زیرا نشان می‌دهد که بازار G5 پتانسیل بالایی دارد و رقابت بر سر آن زیاد است).

اطلاعات همبسته: وقتی شرکت A می‌بیند که شرکت B و C (رقبای اصلی‌اش) پیشنهادهای بسیار بالایی می‌دهند، این می‌تواند به عنوان یک سیگنال یا اطلاعات جدید برای شرکت A باشد. این اطلاعات ممکن است به شرکت A این پیام را بدهد که:

یا رقبایش به اطلاعاتی دست یافته‌اند که او ندارد و این فرکانس‌ها پتانسیل بیشتری از آنچه او فکر می‌کرده‌اند، دارند.

یا اینکه رقابت در بازار آینده 5G بسیار شدید خواهد بود و داشتن این فرکانس‌ها برای حفظ سهم بازار حیاتی‌تر از آن چیزی است که قبلا تصور می‌کرده است.

بنابراین، ارزش نهایی که شرکت A برای فرکانس‌ها قائل می‌شود، نه تنها بر اساس نیازهای خصوصی خودش، بلکه بر اساس اطلاعاتی که از رقابت و پیشنهادهای دیگران به دست می‌آورد، تغییر می‌کند و همبستگی نشان می‌دهد. ارزش نهایی فرکانس‌ها برای شرکت A به عملکرد و تصمیمات دیگران در حراجی هم بستگی پیدا می‌کند.

در ارزش‌های همبسته، ارزش نهایی برای هر شرکت‌کننده ترکیبی از ترجیحات و نیازهای خصوصی اوست، اما این ارزش توسط انتظارات و اطلاعاتی که از رفتار دیگران (به ویژه پیشنهادهای آن‌ها) به دست می‌آید، تعدیل و تحت تأثیر قرار می‌گیرد.

B) چالش‌ها در حراجی‌های انگلیسی (English Auctions)

حراج‌های انگلیسی به دلیل ماهیت شفاف و رقابتی خود، که در آن شرکت‌کنندگان به‌تدریج پیشنهادات خود را افزایش می‌دهند تا برنده مشخص شود، در معرض سوءاستفاده‌های خاصی قرار دارند.

پیشنهاد جعلی (Shill Bidding)

در این روش، فروشنده یا همدستان او با استفاده از حساب‌های جعلی یا هماهنگی با افراد دیگر، پیشنهادهای غیرواقعی ارائه می‌دهند تا قیمت کالا را به‌طور مصنوعی افزایش دهند. به عنوان مثال، در یک حراج آنلاین آثار هنری در سال ۲۰۲۵، فروشنده یک اثر دیجیتال (مانند NFT) را عرضه می‌کند. او از چند حساب جعلی برای ارائه پیشنهادهای غیرواقعی استفاده می‌کند تا تقاضای کاذب ایجاد کرده و خریداران واقعی را به پرداخت مبلغ بالاتر ترغیب کند. پیشنهادها از ۱۰۰۰ دلار به ۵۰۰۰ دلار افزایش می‌یابد، اما بخشی از این پیشنهادها از حساب‌های جعلی است که قصد خرید ندارند.

این روش با ایجاد رقابت کاذب، خریداران واقعی را فریب می‌دهد و آن‌ها را وادار به پرداخت مبلغی بیش از ارزش واقعی کالا می‌کند. این امر اعتماد به فرآیند حراج را تضعیف کرده و ممکن است باعث کاهش مشارکت در حراج‌های آینده شود، زیرا خریداران احساس می‌کنند نمی‌توانند در رقابتی عادلانه شرکت کنند.

پیشنهاد لحظه آخری (Bid Sniping)

پیشنهاد لحظه آخری زمانی رخ می‌دهد که فردی یا نرم‌افزار خودکار در ثانیه‌های پایانی حراج، پیشنهادی بالاتر ارائه می‌دهد و فرصتی برای واکنش به دیگر شرکت‌کنندگان باقی نمی‌گذارد. به عنوان مثال، در یک حراج آنلاین خودروهای کلاسیک که به‌صورت زنده پخش می‌شود، یک شرکت‌کننده از نرم‌افزار خاصی استفاده می‌کند تا در دو ثانیه پایانی حراج، پیشنهادی به مبلغ ۲۱,۰۰۰ دلار (کمی بالاتر از پیشنهاد فعلی ۲۰,۰۰۰ دلار) ثبت کند. این کار باعث می‌شود رقبا نتوانند به‌موقع واکنش نشان دهند.

این روش فرصت برابر برای رقابت را از بین می‌برد، زیرا شرکت‌کنندگان عادی که به‌صورت دستی پیشنهاد می‌دهند، نمی‌توانند با سرعت نرم‌افزارهای خودکار رقابت کنند. این امر فرآیند شفاف و تدریجی حراج‌های انگلیسی را مختل کرده و ممکن است باعث دلسردی شرکت‌کنندگان واقعی شود. همچنین، این روش می‌تواند به کاهش ارزش واقعی کالا در حراج منجر شود، زیرا رقابت طبیعی را محدود می‌کند.

هر دو مکانیزم پیشنهاد جعلی و پیشنهاد لحظه آخری، با دستکاری در رقابت یا بهره‌برداری از زمان‌بندی حراج، عدالت آن را به خطر می‌اندازند. برای حفظ اعتماد و یکپارچگی حراج‌ها، اقداماتی مانند تأیید هویت شرکت‌کنندگان، تمدید زمان حراج در صورت پیشنهادهای لحظه آخری، یا استفاده از فناوری‌های تشخیص تقلب ضروری است

C) آسیب‌پذیری‌های حراج ویکری: مکانیزم‌های تقلب و راه‌حل‌های نوآورانه

حراج ویکری، که به‌عنوان حراج مهر و موم شده با قیمت دوم شناخته می‌شود، به دلیل تشویق به ارائه پیشنهادهای صادقانه (ارزش واقعی) در تئوری مورد توجه است. با این حال، این نوع حراج در برابر برخی رفتارهای استراتژیک ناسالم آسیب‌پذیر است.

نقص طراحی: تبانی در پیشنهادها (Collusion)

در حراج ویکری، بالاترین پیشنهاددهنده برنده می‌شود اما تنها دومین پیشنهاد بالا را پرداخت می‌کند. این ساختار، اگرچه در تئوری پیشنهادهای صادقانه را ترویج می‌کند، در عمل امکان تبانی بین پیشنهاددهندگان را فراهم می‌سازد. در این حالت، گروهی از پیشنهاددهندگان می‌توانند به‌طور مخفیانه توافق کنند که تنها یکی از آن‌ها پیشنهاد بالایی نزدیک به ارزش واقعی ارائه دهد و دیگران پیشنهادهای بسیار پایینی (مانند نزدیک به قیمت پایه) ثبت کنند. این کار باعث می‌شود برنده با قیمتی بسیار پایین‌تر از ارزش واقعی کالا برنده شود.

به عنوان مثال، در یک حراج آنلاین در سال ۲۰۲۵ برای یک نام دامنه کمیاب، گروهی از پیشنهاد دهندگان توافق می‌کنند که یکی از آن‌ها پیشنهادی به مبلغ ۱۰,۰۰۰ دلار ارائه دهد و دیگران پیشنهادهایی در حد ۱۰۰ دلار ثبت کنند. در نتیجه، برنده تنها ۱۰۰ دلار پرداخت می‌کند، در حالی که ارزش واقعی دامنه بسیار بالاتر است.

دلیل اصلی نقص:

مهر و موم بودن پیشنهادها و عدم شفافیت کامل در فرآیند حراج، امکان تبانی را تسهیل می‌کند. از آنجا که تنها پیشنهاد برنده و قیمت دوم اعلام می‌شود و سایر پیشنهادها مخفی می‌مانند، پیشنهاد دهندگان متخلف می‌توانند بدون ترس از شناسایی یا فشار رقابتی، پیشنهادهای پایین ارائه دهند. این عدم شفافیت، نظارت بر رفتارهای ناسالم را دشوار می‌سازد.

راه‌حل نوآورانه: مکانیزم افشای پویا با شفافیت جزئی

برای کاهش آسیب‌پذیری در برابر تبانی، می‌توان یک فرآیند پیشنهاد چندمرحله‌ای با شفافیت جزئی معرفی کرد:

مرحله پیشنهاد مهر و موم شده: پیشنهاددهندگان ارزش واقعی خود را به‌صورت محرمانه ارائه می‌دهند.

افشای جزئی اطلاعات: سیستم حراج، بدون افشای پیشنهادهای فردی، آمارهای کلی مانند میانه پیشنهادها یا توزیع پیشنهادها را به‌صورت ناشناس منتشر می‌کند.

مرحله تنظیم نهایی: پیشنهاددهندگان می‌توانند بر اساس اطلاعات بازار منتشرشده، پیشنهادهای خود را (تنها به سمت افزایش) اصلاح کنند.

چرا این روش مؤثر است:

اختلال در تبانی: با ارائه اطلاعات کلی در مورد پیشنهادها، اطمینان پیشنهاددهندگان متخلف از اجرای توافق مخفیانه کاهش می‌یابد، زیرا نمی‌توانند پیشنهادهای دقیق دیگران را تأیید کنند.

حفظ صداقت: انگیزه پیشنهاد صادقانه (ویژگی اصلی حراج ویکری) همچنان حفظ می‌شود، اما اطلاعات بازار، انگیزه پیشنهادهای پایین غیرواقعی را کاهش می‌دهد.

کاهش نیاز به نظارت سنگین: برخلاف روش‌های سنتی تشخیص تقلب که پس از حراج اعمال می‌شوند، این روش به‌صورت ساختاری انگیزه‌های تبانی را تضعیف می‌کند.

راه‌حل جایگزین: استفاده از قراردادهای هوشمند مبتنی بر بلاکچین، که در آن تاریخچه پیشنهادها به‌صورت غیرقابل‌تغییر ثبت شده و پس از حراج افشا می‌شود. این امر امکان شناسایی و مجازات تبانی را به‌صورت پسینی فراهم می‌کند، بدون اینکه ساختار اصلی حراج تغییر کند.

آسیب‌پذیری حراج ویکری در برابر تبانی ناشی از محرمانه بودن پیشنهادها و عدم شفافیت کامل است. با معرفی شفافیت جزئی یا مکانیزم‌های پویا مانند افشای آمار کلی یا قراردادهای هوشمند، می‌توان انگیزه‌های تبانی را کاهش داد و در عین حال ویژگی‌های مثبت این حراج، مانند تشویق به پیشنهاد صادقانه، را حفظ کرد.

بخش دوم

```python
def random_argmax(arr):
    """
    Randomly selects one index among the maximum values in an array.
    """
    max_val = np.max(arr)
    max_indices = np.where(arr == max_val)[0]
    return np.random.choice(max_indices)
```

تابع random_argmax(arr):

این یک تابع کمکی است. زمانی که یک آرایه arr چندین عنصر با مقدار بیشینه یکسان داشته باشد، این تابع به طور تصادفی اندیس (موقعیت) یکی از این مقادیر بیشینه را انتخاب می‌کند.

ابتدا مقدار بیشینه را در آرایه ورودی arr پیدا می‌کند.

سپس تمام اندیس‌هایی که این مقدار بیشینه در آن‌ها قرار دارد را مشخص می‌کند.

در نهایت، یکی از این اندیس‌ها را به صورت تصادفی انتخاب می‌کند.

برای رفع حالت تساوی زمانی که چند عامل بالاترین پیشنهاد یکسان را برای یک منبع ارائه می‌دهند، استفاده می‌شود.

```python
class Agent:
    """
    Represents an agent in the auction system.
    """
    def __init__(self, agent_id: int, valuations: List[float]):
        self.agent_id = agent_id
        self.valuations = valuations  # Valuations for each resource
        self.current_bid_resource = None  # Resource currently bidding on
        self.current_bid_amount = 0
        self.eliminated_resources = set()  # Resources eliminated from
        self.no_bid_count = {}  # Track consecutive no-bids per resource
        self.bid_history = []  # Track bidding history
```

نماینده یک شرکت‌کننده (پیشنهاد دهنده قیمت) در مزایده است.

ویژگی‌های کلیدی (Attributes):

agent_id: یک شناسه منحصربه‌فرد برای عامل.

valuations: لیستی از اعداد که هر عدد نشان‌دهنده میزان ارزش هر منبع برای آن عامل است. برای مثال، valuations[0] میزان ارزش منبع شماره صفر برای آن عامل است.

current_bid_resource: منبعی که عامل در حال حاضر در یک دور خاص برای آن پیشنهاد قیمت می‌دهد.

current_bid_amount: مبلغی که عامل در حال حاضر پیشنهاد داده است.

eliminated_resources: مجموعه‌ای از منابعی که عامل دیگر مجاز به پیشنهاد دادن برای آن‌ها نیست (مثلاً اگر به طور مداوم پیشنهاد قیمت پایینی ارائه دهد).

no_bid_count: یک دیکشنری برای ردیابی تعداد دورهای متوالی که یک عامل برای یک منبع خاص پیشنهادی ارائه نکرده است. این برای قانون حذف عامل استفاده می‌شود.

bid_history: لیستی برای ثبت فعالیت‌های پیشنهاددهی عامل در طول دورها.

```python
def reset_round(self):
    """Reset agent state for new round."""
    self.current_bid_resource = None
    self.current_bid_amount = 0

def can_bid_on_resource(self, resource_id: int) -> bool:
    """Check if agent can bid on a specific resource."""
    return resource_id not in self.eliminated_resources

def eliminate_from_resource(self, resource_id: int, verbose: bool = False):
    """Eliminate agent from bidding on a specific resource."""
    self.eliminated_resources.add(resource_id)
    if verbose:
        print(f"Agent {self.agent_id} eliminated from Resource {resource_id}")
```

reset_round: وضعیت پیشنهاددهی عامل (منبع و مبلغ پیشنهاد فعلی) را در ابتدای یک دور جدید مزایده بازنشانی می‌کند.

can_bid_on_resource: بررسی می‌کند که آیا عامل مجاز به پیشنهاد دادن برای یک منبع خاص است یا خیر (یعنی در مجموعه eliminated_resources او نباشد).

eliminate_from_resource: یک منبع را به مجموعه eliminated_resources عامل اضافه می‌کند.

```python
def calculate_bid_strategy(self, resource_id: int, current_price: float, epsilon: float) -> float:
    """
    Calculate strategic bid for a resource.
    Strategy: Bid incrementally above current price, but not more than valuation.
    """
    valuation = self.valuations[resource_id]
    min_bid = current_price + epsilon

    # Don't bid more than valuation
    if min_bid > valuation:
        return 0  # Can't afford to bid

    # Strategic bidding: bid slightly above minimum required
    # Add some randomness for different outcomes
    bid_increment = epsilon + random.uniform(0, epsilon * 0.5)
    strategic_bid = min(current_price + bid_increment, valuation)

    return strategic_bid
```

در اینجا منطق پیشنهاددهی عامل قرار دارد.

ورودی‌های آن resource_id (شناسه منبع)، current_price (قیمت فعلی آن) و یک مقدار کوچک epsilon (حداقل افزایش پیشنهاد) هستند.

عامل مبلغی کمی بیشتر از current_price (به طور مشخص، current_price + epsilon به علاوه یک مقدار تصادفی کوچک) پیشنهاد می‌دهد. با این حال، عامل هرگز بیش از valuation (ارزش‌گذاری) خود برای آن منبع پیشنهاد نخواهد داد. اگر حداقل پیشنهاد لازم از ارزش‌گذاری او بیشتر باشد، پیشنهاد صفر می‌دهد (یعنی پیشنهادی ارائه نمی‌کند).

```python
def select_resource_to_bid(self, auction_state: Dict, epsilon: float) -> Optional[Tuple[int, float]]:
    """
    Select which resource to bid on and determine bid amount.
    Strategy: Bid on resource with highest potential profit.
    """
    best_resource = None
    best_bid = 0
    best_profit = -1

    for resource_id in range(len(self.valuations)):
        if not self.can_bid_on_resource(resource_id):
            continue

        current_price = auction_state['prices'][resource_id]
        valuation = self.valuations[resource_id]

        # Calculate potential bid
        potential_bid = self.calculate_bid_strategy(resource_id, current_price, epsilon)

        if potential_bid > 0:
            potential_profit = valuation - potential_bid

            # Select resource with highest profit potential
            if potential_profit > best_profit:
                best_profit = potential_profit
```

این متد تعیین می‌کند که عامل در دور فعلی برای کدام منبع پیشنهاد دهد و چه مبلغی را پیشنهاد دهد.

عامل تمام منابعی را که هنوز واجد شرایط پیشنهاد دادن برای آن‌ها است، بررسی می‌کند. برای هر کدام، با استفاده از calculate_bid_strategy یک پیشنهاد بالقوه را محاسبه می‌کند. سپس "سود بالقوه" (ارزش‌گذاری خود برای منبع منهای پیشنهاد بالقوه‌اش) را محاسبه می‌کند. عامل تصمیم می‌گیرد برای منبعی پیشنهاد دهد که بالاترین سود بالقوه را ارائه می‌دهد.

اگر هیچ منبعی سود بالقوه مثبتی ارائه ندهد، عامل پیشنهادی نمی‌دهد.

```python
class Auction:
    """
    Manages the multi-round auction process.
    """
    def __init__(self, n_resources: int, m_agents: int, valuations: List[List[float]], epsilon: float):
        self.n_resources = n_resources
        self.m_agents = m_agents
        self.epsilon = epsilon

        # Initialize agents
        self.agents = []
        for j in range(m_agents):
            agent_valuations = [valuations[i][j] for i in range(n_resources)]
            self.agents.append(Agent(j, agent_valuations))

        # Initialize auction state
        self.prices = [0.0] * n_resources  # Current prices for each resource
        self.allocations = [-1] * n_resources  # Which agent owns each resource (-1 = unallocated)
        self.round_number = 0
        self.auction_history = []
```

کلاس Auction: مدیریت فرآیند کلی مزایده، شامل چندین دور، تخصیص منابع و به‌روزرسانی قیمت‌ها.

n_resources: تعداد کل منابع موجود در مزایده.

m_agents: تعداد کل عوامل شرکت‌کننده.

epsilon: حداقل میزانی که پیشنهادات باید افزایش یابند.

agents: لیستی از اشیاء (آبجکت‌های) Agent که در مزایده شرکت می‌کنند.

prices: لیستی که در آن prices[i] بالاترین پیشنهاد فعلی (و در نتیجه قیمت فعلی) برای منبع i است. با مقدار اولیه صفر.

allocations: لیستی که در آن allocations[i] شناسه عاملی است که در حال حاضر منبع i را در اختیار دارد. با مقدار اولیه -1 (تخصیص نیافته).

round_number: شماره دور فعلی مزایده را نگه می‌دارد.

auction_history: وضعیت مزایده (پیشنهادات، تخصیص‌ها، قیمت‌ها) را در پایان هر دور ثبت می‌کند.

```python
def reset_auction(self):
    """Reset auction to initial state."""
    self.prices = [0.0] * self.n_resources
    self.allocations = [-1] * self.n_resources
    self.round_number = 0
    self.auction_history = []

    # Reset all agents
    for agent in self.agents:
        agent.eliminated_resources = set()
        agent.no_bid_count = {}
        agent.bid_history = []
```

مزایده را به وضعیت اولیه خود بازنشانی می‌کند (قیمت‌ها، تخصیص‌ها، شماره دور، وضعیت عوامل)، که امکان اجرای چندباره با همان تنظیمات اولیه را فراهم می‌کند.

```python
def get_auction_state(self) -> Dict:
    """Get current auction state."""
    return {
        'prices': self.prices.copy(),
        'allocations': self.allocations.copy(),
        'round': self.round_number
    }
```

یک دیکشنری حاوی قیمت‌های فعلی، تخصیص‌ها و شماره دور را برمی‌گرداند. این اطلاعات توسط عوامل برای تصمیم‌گیری در مورد پیشنهاددهی استفاده می‌شود.

```python
def process_round_bids(self, verbose: bool = False) -> bool:
    """
    Process all bids for the current round.
    Returns True if any changes occurred, False if auction should end.
    """
    # Reset agents for new round
    for agent in self.agents:
        agent.reset_round()

    # Collect bids from all agents
    round_bids = {}  # resource_id -> [(agent_id, bid_amount), ...]

    if verbose:
        print(f"\n--- Round {self.round_number + 1} Bid Collection ---")
        print(f"Current prices: {[f'{p:.2f}' for p in self.prices]}")
        print(f"Current allocations: {self.allocations}")

    for agent in self.agents:
        bid_info = agent.select_resource_to_bid(self.get_auction_state(),
```

این منطق اصلی برای یک دور مزایده است.

بازنشانی عوامل: وضعیت پیشنهاددهی هر عامل را برای دور جدید بازنشانی می‌کند.

جمع‌آوری پیشنهادات: هر عامل متد select_resource_to_bid خود را فراخوانی می‌کند تا تصمیم بگیرد آیا و برای چه چیزی پیشنهاد دهد. تمام پیشنهادات جمع‌آوری می‌شوند.

پردازش پیشنهادات برای هر منبع: برای هر منبع:

اگر پیشنهادی وجود داشته باشد، بالاترین پیشنهاد را پیدا می‌کند.

اگر در بالاترین پیشنهاد تساوی وجود داشته باشد، از random_argmax (به طور ضمنی با استفاده از random.randint روی لیست پیشنهاددهندگان با حداکثر پیشنهاد) برای انتخاب تصادفی برنده استفاده می‌شود.

پیشنهاد برنده باید حداقل current_price + epsilon باشد.

اگر پیشنهاد معتبر باشد، منبع به عامل برنده تخصیص داده می‌شود و price (قیمت) برای آن منبع به مبلغ پیشنهاد برنده به‌روز می‌شود.

پرچم changes_made در صورت هرگونه تغییر در تخصیص یا قیمت، True تنظیم می‌شود.

به‌روزرسانی شمارش عدم پیشنهاد و حذف‌ها: متد update_no_bid_counts را فراخوانی می‌کند.

ثبت تاریخچه: فعالیت‌های دور را ذخیره می‌کند.

مقدار changes_made را برمی‌گرداند. مزایده در صورتی پایان می‌یابد که یک دور بدون هیچ تغییری رخ دهد.

```python
def update_no_bid_counts(self, round_bids: Dict, verbose: bool = False):
    """Update no-bid counts and eliminate agents if necessary."""
    eliminations_this_round = []

    for agent in self.agents:
        for resource_id in range(self.n_resources):
            if resource_id not in agent.eliminated_resources:
                # Check if agent bid on this resource
                agent_bid_on_resource = (
                    resource_id in round_bids and
                    any(bid[0] == agent.agent_id for bid in round_bids[resource_id])
                )

                if not agent_bid_on_resource:
                    # Increment no-bid count
                    if resource_id not in agent.no_bid_count:
                        agent.no_bid_count[resource_id] = 0
```

برای هر عامل و هر منبعی که هنوز از آن حذف نشده است:

اگر عامل در دور فعلی برای آن منبع پیشنهادی نداده باشد، no_bid_count او برای آن منبع یک واحد افزایش می‌یابد.

اگر no_bid_count یک عامل برای یک منبع به ۲ برسد (یعنی برای دو دور متوالی که می‌توانسته، برای آن پیشنهاد نداده است)، او از پیشنهاد دادن برای آن منبع خاص (eliminate_from_resource) حذف می‌شود.

اگر عامل برای یک منبع پیشنهادی داده باشد، no_bid_count او برای آن منبع به صفر بازنشانی می‌شود.

```python
def run_auction(self, verbose: bool = False) -> Tuple[List[int], List[float]]:
    """
    Run the complete auction process.
    Returns final allocations and prices.
    """
    if verbose:
        print(f"\n=== Starting Auction ===")
        print(f"Resources: {self.n_resources}, Agents: {self.m_agents}, Epsilon: {self.epsilon}")
        print("Agent valuations:")
        for agent in self.agents:
            print(f"  Agent {agent.agent_id}: {[f'{v:.2f}' for v in agent.valuations]}")

    max_rounds = 50  # Prevent infinite loops

    while self.round_number < max_rounds:
        if verbose:
            print(f"\n--- Round {self.round_number + 1} ---")

        changes_made = self.process_round_bids(verbose)
```

مزایده را از ابتدا تا انتها اجرا می‌کند.

به طور مکرر process_round_bids را فراخوانی می‌کند.

مزایده تا زمانی ادامه می‌یابد که changes_made در یک دور True باشد و به حد مجاز تعداد دورها (max_rounds) نرسیده باشد.

در صورت True بودن verbose، خروجی توضیحی چاپ می‌کند.

allocations (تخصیص‌های نهایی) و prices (قیمت‌های نهایی) را برمی‌گرداند.

```python
def print_final_results(self):
    """Print final allocation results."""
    for resource_id in range(self.n_resources):
        if self.allocations[resource_id] != -1:
            agent_id = self.allocations[resource_id]
            price = self.prices[resource_id]
            print(f"Resource {resource_id + 1} → Agent {agent_id + 1} at price {price:.2f}")
        else:
            print(f"Resource {resource_id + 1} → Unallocated")
```

خلاصه‌ای از اینکه کدام عامل کدام منبع را و با چه قیمتی برنده شده است، چاپ می‌کند.

```python
def run_multiple_auctions(n_resources: int, m_agents: int, valuations: List[List[float]],
                          epsilon: float, num_runs: int = 5, verbose: bool = False):
    """
    Run multiple auction instances and compare results.
    """

    results = []

    for run in range(num_runs):
        print(f"{'='*20} RUN {run + 1} {'='*20}")

        # Create new auction instance
        auction = Auction(n_resources, m_agents, valuations, epsilon)

        # Run auction
        allocations, prices = auction.run_auction(verbose=verbose)

        # Print results
        auction.print_final_results()
```

اجرای کامل شبیه‌سازی مزایده به دفعات متعدد با همان تنظیمات اولیه (ارزش‌گذاری‌ها، تعداد عوامل/منابع، اپسیلون).

پارامترهای مزایده و num_runs (تعداد اجراها) را به عنوان ورودی می‌گیرد.

در یک حلقه، برای هر اجرا:

یک نمونه (instance) جدید از Auction ایجاد می‌کند.

مزایده را با استفاده از auction.run_auction() اجرا می‌کند.

نتایج نهایی آن اجرا را چاپ می‌کند.

تخصیص‌ها و قیمت‌های آن اجرا را ذخیره می‌کند.

لیستی از نتایج را برمی‌گرداند، که هر آیتم در لیست حاوی تخصیص‌ها و قیمت‌ها برای یک اجرای کامل مزایده است.

این برای مشاهده اینکه چگونه تصادفی بودن در رفع تساوی و استراتژی پیشنهاددهی ممکن است منجر به نتایج متفاوت حتی با شرایط اولیه یکسان شود، مفید است.

```
print("Example Input:")
# Example input
n = 2  # Number of resources
m = 3  # Number of agents
V = [
    [50, 80, 70],  # Valuations of agents for resource 1
    [60, 90, 30]   # Valuations of agents for resource 2
]
ε = 10  # Minimum bid increment

# Run multiple auctions with verbose output
results = run_multiple_auctions(n, m, V, ε, num_runs=5, verbose=False)
```

با ورودی های پیش فرض مسئله یک بار کد را اجرا کردیم که به نتیجه زیر رسیدیم:

```
Example Input:
==================== RUN 1 ====================
Resource 1 → Agent 3 at price 65.71
Resource 2 → Agent 2 at price 80.08
==================== RUN 2 ====================
Resource 1 → Agent 3 at price 63.44
Resource 2 → Agent 2 at price 90.00
==================== RUN 3 ====================
Resource 1 → Agent 3 at price 60.38
Resource 2 → Agent 2 at price 87.09
==================== RUN 4 ====================
Resource 1 → Agent 3 at price 69.46
Resource 2 → Agent 2 at price 81.52
==================== RUN 5 ====================
Resource 1 → Agent 3 at price 67.24
Resource 2 → Agent 2 at price 82.15
```

```python
# Generate random parameters
n = random.randint(2, 5)  # Random number of resources (2-5)
m = random.randint(2, 6)  # Random number of agents (2-6)

# Generate random valuations matrix
# Each agent's valuation for each resource is between 10 and 100
V = []
for resource_id in range(n):
    resource_valuations = []
    for agent_id in range(m):
        valuation = random.randint(10, 100)
        resource_valuations.append(valuation)
    V.append(resource_valuations)

# Random epsilon between 5 and 15
ε = random.randint(5, 15)

print(f"Number of resources: {n}")
print(f"Number of agents: {m}")
print(f"Minimum bid increment (ε): {ε}")
print("\nAgent valuations for each resource:")
for i, resource_vals in enumerate(V):
    print(f"Resource {i + 1}: {resource_vals}")

print(f"\nAgent-wise valuations:")
for agent_id in range(m):
    agent_vals = [V[resource_id][agent_id] for resource_id in range(n)]
    print(f"Agent {agent_id + 1}: {agent_vals}")

# Run multiple auctions with verbose output
results = run_multiple_auctions(n, m, V, ε, num_runs=5, verbose=False)
```

یکبار با مقادیر رندوم اجرا کردیم:

تعداد تصادفی منابع: بین ۲ تا ۵ منبع

تعداد تصادفی عامل‌ها: بین ۲ تا ۶ عامل

ارزش‌گذاری‌های تصادفی: ارزش‌گذاری هر عامل برای هر منبع به صورت تصادفی انجام می‌شود

بین ۱۰ تا ۱۰۰ تولید می‌شود

اپسیلون تصادفی: حداقل افزایش پیشنهاد بین ۵ تا ۱۵

خروجی:

```
Number of resources: 4
Number of agents: 6
Minimum bid increment (ε): 15

Agent valuations for each resource:
Resource 1: [96, 58, 60, 35, 19, 85]
Resource 2: [98, 90, 41, 23, 99, 48]
Resource 3: [97, 86, 25, 82, 15, 54]
Resource 4: [78, 64, 94, 57, 18, 74]

Agent-wise valuations:
Agent 1: [96, 98, 97, 78]
Agent 2: [58, 90, 86, 64]
Agent 3: [60, 41, 25, 94]
Agent 4: [35, 23, 82, 57]
Agent 5: [19, 99, 15, 18]
Agent 6: [85, 48, 54, 74]
==================== RUN 1 ====================
Resource 1 → Agent 1 at price 91.02
Resource 2 → Agent 5 at price 94.18
Resource 3 → Agent 4 at price 78.41
Resource 4 → Agent 3 at price 87.26
==================== RUN 2 ====================
Resource 1 → Agent 6 at price 72.27
Resource 2 → Agent 5 at price 94.87
Resource 3 → Agent 1 at price 97.00
Resource 4 → Agent 3 at price 91.23
==================== RUN 3 ====================
Resource 1 → Agent 1 at price 93.79
Resource 2 → Agent 5 at price 99.00
Resource 3 → Agent 4 at price 76.10
Resource 4 → Agent 3 at price 79.58
==================== RUN 4 ====================
Resource 1 → Agent 6 at price 79.73
Resource 2 → Agent 5 at price 99.00
Resource 3 → Agent 4 at price 80.09
Resource 4 → Agent 3 at price 94.00
==================== RUN 5 ====================
Resource 1 → Agent 6 at price 78.08
Resource 2 → Agent 5 at price 99.00
Resource 3 → Agent 1 at price 97.00
Resource 4 → Agent 3 at price 80.95
```

برای رفع برخی ایرادات و ابهامات از AI استفاده شده است.