Pseudocode and detailed information of two baselines

As stated in the main text, the simulated annealing algorithm and the genetic algorithm are selected as the two baselines. Here, we provide detailed information about how the two algorithms work in our model setting.

With regards to the simulated annealing algorithm, the state and the temperature are two core concepts when using this algorithm. In our model, we regard "the feasible solution" as the state and "the seller's profit" as the temperature. Then, the adopted simulated annealing algorithm in this paper has the following main steps: in the first step, the simulated annealing algorithm starts from a random feasible solution (i.e., state) and looks for a higher profit (i.e., temperature) in the surrounding states, where the surrounding states are generated by randomly changing the current solution to its close feasible solutions; in the second step, we randomly select one of the surrounding states as the promising state to replace the current state based on the designed probability acceptance function; in the third step, the process will be terminated when the temperature converges. TABLE A1 shows the pseudocode of the adopted simulated annealing algorithm and the corresponding implementation parameters.

TABLE A1. PSEUDOCODE OF THE ADOPTED SIMULATED ANNEALING ALGORITHM.

Input:	The customer set *N* and their social network $\boldsymbol{G}$
Output:	The optimal price p^*, the seller's highest profit pr^* and the optimal parameter η^*.
Initialization:	To initialize the state by randomly generating a flexible solution $x_0 = [\boldsymbol{s}_0, p_0, \eta_0, \boldsymbol{r}_0]$, where $\boldsymbol{s}_0$ is the action network, p_0 is the price, η_0 is the award and $\boldsymbol{r}_0$ is the referral network;
	To calculate the seller's profit pr_0 based on x_0;
	time = 0.
Process:	**While** *time* < 1000
	To randomly find k flexible solutions close with x_0 as the surrounding state set $\{x_i\}_{i=1}^{k}$ // *k* is set as 10
	To calculate the seller's profit pr_i according to x_i, where $i = 1, 2, \cdots, k$
	To reorder the element in $\{pr_i\}_{i=1}^{k}$ in a descending order and the corresponding element in $\{x_i\}_{i=1}^{k}$
	For *i* = 1 to k
	key = random(0,1)
	If *key* < exp $\left(\frac{-0.4pr_0}{pr_i \log(time+1)}\right)$ // Probability acceptance function
	$x_0 = x_i$, $pr_0 = pr_i$
	Break For
	End If
	End For
	If the fluctuation of pr_0 is less than 0.0005 for 5 consecutive times. // Temperature converges
	Output $x^* = x_0$, $pr^* = pr_0$
	Break While
	End if
	time = *time* +1
	End While

With regards to the genetic algorithm, the population and the fitness are two core concepts when using this algorithm. In our model, we regard "the feasible solution" as the population and "the seller's profit" as the fitness. With the aim of maximizing the fitness, the adopted genetic algorithm in this paper has the following main steps: in the first step, we randomly generate *k* feasible solutions as *k* populations, where *k* is set as 10; in the second step, we generate next populations according to the variation rules on the basis of the current populations; the third step is to repeat the second step until the achieved fitness. TABLE A2 shows the pseudocode of the adopted genetic algorithm and the corresponding implementation parameters.

TABLE A2. PSEUDOCODE OF THE ADOPTED GENETIC ALGORITHM.

Input:		The customer set *N* and their social network $\boldsymbol{G}$
Output:		The optimal price p^*, the seller's highest profit pr^* and the optimal parameter η^*.
Initialization:	(1)	To initialize *k* populations by generating *k* feasible solutions $\{x_i = [s_i, p_i, \eta_i, r_i]\}_{i=1}^{k}$, where $s_i,\ p_i,\ \eta_i, and\ r_i$ is the action network, the price, the award and the referral network of *i*-th population, respectively. Here, x_i is decoded in a binary format as strings of 0s and 1s;
	(2)	To calculate the seller's profit pr_i according to x_i, where $i = 1, 2, \cdots, k$;
	(3)	*time* = 0.
Process:	(1)	**While** *time* < 1000
	(2)	To reorder the element in $\{pr_i\}_{i=1}^{k}$ in a descending order and the corresponding element in $\{x_i\}_{i=1}^{k}$
	(3)	$pr^* = pr_1$ and $x^* = x_1$
	(4)	$x_{next} = \emptyset$ // Empty set
	(5)	**For** *j* = 1 to *k*
	(6)	*key* = random(0,1)
	(7)	**If** *key* < 0.4/*j*
	(8)	$x_{next} = x_{next} \cup \{x_i\}$ // Direct heredity
	(9)	**Else If** *key* < 0.8/*j*,
	(10)	To generate a new population y by replacing part of x_j by that of x_i, where x_i is randomly chosen from $\{x_i\}_{i=1}^{k}$. // Crossover heredity
	(11)	$x_{next} = x_{next} \cup \{y\}$
	(12)	**Else**, To generate a new population z by randomly reforming x_i
	(13)	$x_{next} = x_{next} \cup \{z\}$ // Mutation heredity
	(14)	**End If**
	(15)	**End For**
	(16)	$\{x_i\}_{i=1}^{k} = x_{next}$
	(17)	To calculate the seller's profit pr_i according to x_i, where $i = 1, 2, \cdots, k$
	(18)	**If** the fluctuation of pr^* is less than 0.0005 for 5 consecutive times. // Fitness converges
	(19)	**Output** pr^* and x^*
	(20)	**Break while**
	(21)	**End If**
	(22)	*time* = *time*+1
	(23)	**End While**