An intelligent matching recommendation algorithm for a manufacturing capacity sharing platform with fairness concerns

Xie, L., Zhang, J., Meng, Q., Jin, Y., & Liu, W. (2022). An intelligent matching recommendation algorithm for a manufacturing capacity sharing platform with fairness concerns. *International Journal of Production Research*. Advance online publication. https://doi.org/10.1080/00207543.2022.2155999

Published in:
International Journal of Production Research

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

An Intelligent Matching Recommendation Algorithm for A Manufacturing Capacity Sharing Platform with Fairness Concerns

Lei Xie[1], Jianghua Zhang[1], Qingchun Meng[1], Yan Jin[2], Weibo Liu[1*]

1. School of Management, Shandong University, Jinan 250100, China
2. School of Mechanical and Aerospace Engineering, Queen's University Belfast, Ashby Building, Belfast BT9 5AH, UK

Authors:

Lei Xie
School of Management, Shandong University
Jinan 250100, China
Email: lxie@sdu.edu.cn

Jianghua Zhang
School of Management, Shandong University
Jinan 250100, China
Email: zhangjianghua@sdu.edu.cn

Qingchun Meng
School of Management, Shandong University
Jinan 250100, China
Email: meqich@sdu.edu.cn

Yan Jin
School of Mechanical and Aerospace Engineering, Queen's University Belfast, Ashby Building
Belfast BT9 5AH, UK
Email:y.jin@qub.ac.uk

Corresponding Author:

Weibo Liu
School of Management, Shandong University
Jinan 250100, China
Tel: +86-13176653119
Email: wliu09@qub.ac.uk

Abstract: A supply and demand mismatch, or imbalance of the amount of supplies in the market, is always an issue and can happen all the time. Capacity sharing is an effective way to address this problem, and the capacity sharing platform facilitates the optimal matching between multiple capacity buyers and sellers. In the context of Industry 4.0, many industries are adopting intelligent algorithms to assist in decision-making. This paper presents an optimal or near-optimal matching algorithm to cope with a large volume of capacity sharing problems. The fairness of the matching solution is captured by including three objectives from platform, sellers and buyers. In this paper, a 2-dimensional crossover and an order-first mutation are developed and employed with genetic algorithms (GA), including GA and NSGA-II. Additionally, a novel repair mechanism is proposed by considering various constraints to transform infeasible solutions into feasible ones. Two matching schemes are studied based on whether orders from buyers can be split or not. The results show that both algorithms based on traditional GA and NSGA-II are effective for different schemes. In addition, it is found that GA has better performance in the case of "more sellers" and NSGA-II shows better performance in the "more buyers" case.

Key words: Capacity Sharing; Genetic Algorithm; Multiple Objectives

1. Introduction

The mismatch between production capacity and demand is taking a toll on manufacturers. Due to the high shortage costs, some manufacturers may invest in capacities to meet high demand during the peak sales season. However, when the peak sales season is over, the overinvestment in capacity may lead to excess capacity. For example, the outbreak of COVID-19 caused the market demand for masks and other medical products to increase dramatically. Manufacturers expanded production capacity or used surge capacity to meet market demand. Considering that the situation in China has improved, manufacturers of one-time-use nonwoven masks and one-time-use medical masks may face the problem of excess capacity [1]. To reduce investment and increase capacity flexibility, an increasing number of companies are outsourcing to an original equipment manufacturer (OEM) for production. As a well-known snack brand in China, three squirrel snack products include a wide range of products, such as nuts and dried fruits. Different manufacturers produce these products in Hangzhou Province,

Zhejiang Province, etc. Similar examples include Nike and Apple. This production model allows companies to focus more on what they are good at - branding, marketing, etc.

With the development of the internet and logistics technology, an increasing number of companies share capacities through the capacity sharing platform. Such a platform provides information about the capacity demand and supply and supervises transactions. The platform lowers the barriers to participation in capacity sharing and helps an increasing number of SMEs to participate in capacity sharing. For example, 1688 (https://factory.1688.com/index.html), a part of Alibaba Inc., allows manufacturers to trade manufacturing capacities for various types of products, such as clothing, toy, digital products, etc. With such a capacity sharing platform, the capacity seller publishes information about its capacity, including the price range, the minimum order quantity, and the number of workers. The platform also shows information about cumulative transaction volume (CTV) and rating for each capacity seller. These factors will be jointly considered by the buyer when determining which buyer to match.

A spontaneous match may happen between the buyer and seller. When determining which buyer to choose, each buyer will evaluate all the feasible sellers and make a score for each of them, then select one from those with scores higher than expected (reference point). However, matching may fail due to information asymmetry. Improving information transparency, enables the platform to provide a higher match success rate than the spontaneous match. In addition, the platform also offers a recommendation mechanism to the buyer for reference. For example, 1688 provides a recommended matcher for the buyer, and the buyer can freely determine whether to accept the matcher. In the context of Industry 4.0, an increasing number of companies are using intelligent algorithms to schedule production, control quality, select partners, etc. Platforms such as Amazon, Tmall, Jingdong, and TikTok use algorithms to analyze users' personalized needs and recommend products (or services) for them. The platform can also provide matching services for capacity buyers and sellers through an intelligent algorithm. Keeping track of transaction data, the platform understands the buyer's choice of the seller, and these data allow the algorithm to give the best recommendations. As a result, it will recommend that the seller with scores be larger than the buyer's reference point to ensure that the buyer is willing to accept this recommendation.

To fulfill corporate social responsibility or achieve longer-term growth, the platform should consider the sustainability performance of the capacity sharing system. In this paper, we treat

sustainability as consisting of two parts. On the one hand, manufacturing capacity sharing helps improve the environment and increase resource utilization. For example, in Wucheng County, Dezhou, China, 96 shared factories were established with the support of the government to help small manufacturers revive. It has been reported that the average concentrations of PM2.5, PM10, SO_2, and NO_2 in Wucheng County in 2017 decreased by 19.1%, 17.1%, 11.4%, and 10% year-on-year, respectively (Xie and Han 2020). The platform facilitates capacity sharing, which has a positive impact on the environment. On the other hand, the platform should balance the benefits for both the buyer and seller to prevent them from leaving the capacity sharing system instead of just caring about its own profit. The fairness concern in decision-making contributes to the long-term development of the capacity sharing business. In fact, apart from the maximum profit, the platform may also pursue different objectives at different stages of development. For example, to encourage buyers and sellers to participate in capacity sharing, 1688 does not charge buyers or sellers for the time being. In the literature, Patro et al. (2020) emphasize that for a platform serving a two-sided market, overemphasis on customer satisfaction by the platforms may affect the well-being of the producers. Hence, they focus on recommendations deployed on two-sided platforms, considering the fairness between the customers and producers. Therefore, in this paper, we consider the platform to make recommendations to optimize a mixed objective, including the platform's profit, the sum of the sellers' profit, and the sum of the buyers' surplus utilities.

In this paper, the theoretical contributions include a 2-dimensional crossover and an order-first mutation which are employed for genetic algorithms. A novel repair mechanism is also proposed considering various constraints to transform infeasible solutions into feasible ones. The results show that the new algorithms based on traditional GA and NSGA-II are both effective for different matching schemes.

The practical contributions include the following. This paper provides two matching schemes for the platform: (1) A scheme with a significantly higher objective is proposed by considering that a buyer's order can be assigned to multiple sellers while a seller can accept multiple buyers' orders. (2) Another more convenient one is made by matching the buyer and seller one by one. In addition, the algorithms proposed in this paper provide a platform with better matching solutions for different decision targets and different numbers of buyer sellers. Specifically, the GA has a better performance under the "more sellers" case, and the NSGA-II has a better performance under the "more buyers"

case. Therefore, the algorithms proposed in this paper can help the platform adjust recommendations dealing with the buyers and sellers of different quantitative relationships.

The remainder of this paper is organized as follows. A review of the related literature is provided in Section 2. The problem is described in Section 3, and the algorithms are introduced in Section 4. In Section 5, the results are analyzed and discussed. Finally, Section 6 concludes the work. All the data and results from the experiments are published with Mendeley Data, DOI: 10.17632/sx88drkkth.2.

2. Literature review

The research problem in this paper is related to two streams of literature. The first one is associated with the capacity sharing business through a platform, and the other is relevant to the algorithms for matching.

2.1 Platform capacity sharing

The area of outsourcing, OEM or capacity sharing is of growing interest in theoretical research and in practice. It is not difficult to understand that the OEM and contract manufacturer (CM) may also act as downstream competitors. The price and quantity decisions may be complicated and influenced by the channel power structures (Y. Wang, Niu, and Guo 2013). The different outsourcing structures, such as control and delegation, will also affect the decisions of both OEM and CM (Y. Wang, Niu, and Guo 2014). Focusing on the decision-making sequencing of the capacity-sharing price and retail price, Guo and Wu (2018) studied the capacity sharing business between competitors and discussed the impact of ex-ante and ex-post contracts on operations decisions. These studies contribute significantly to the capacity sharing strategies of oligopolies or monopolies.

With the development of the internet and logistics, an increasing number of middle and small enterprises can participate in capacity sharing through a platform. It is not due to a problem of monopoly or oligopoly. Hence, the problem of capacity sharing through a platform is significantly different from those studied above. Cachion, Daniels and Lobel (2017) studied the pricing schemes implemented on a service platform, finding that surge pricing benefits both buyers and sellers. The capacity was assumed to be self-scheduling, meaning that the capacity providers decided for themselves how often to work. Considering a monopoly transaction platform providing multiple

services with self-scheduling service providers and customers, Zhou et al. (2019) studied the different performance of three contracts, i.e., free-customers-commission (FCC) contracts, dynamic-customers-commission (DCC) contracts, and optimal contracts, on profits, customer surplus, etc. Similarly, Lin and Zhou (2019) studied the selection of pricing policy for a monopoly service platform with self-scheduling capacity providers.

To the best of the authors' knowledge, existing studies focus on the matching problems of bikes, (Agatz et al. 2012) (Furuhata et al. 2013) (X. Wang, Agatz, and Erera 2018), vehicles (He, Hu, and Zhang 2019) (Jorge and Correia 2013) and houses (H. Li and Srinivasan 2019). Boysen, Briskorn and Schewerdfeger (2019) considered an optimization-based matching of supply and demand from a general perspective, which covered various objects, including car sharing, ride sharing, and house sharing. However, their research did not take the characteristics of manufacturing capacity sharing into consideration. For example, when looking for capacity partners, manufacturers will consider multiple factors, such as price, reliability, manufacturing lead time, and minimum order quantity. However, few studies concentrate on manufacturing capacity sharing platforms. In addition, most of the existing relevant literature focuses on a profit-maximizing platform's pricing strategy or contract policy to coordinate the capacity sharing system. Few studies have agreed on platforms matching multiple buyers with multiple sellers considering fairness concerns. This paper attempts to fill this gap.

2.2 Algorithms for matching problems

Various algorithms for matching problems have been developed. Cheng et al. (2017) developed the concept of a supply–demand matching hypernetwork. A Matching Net was proposed incorporating a manufacturing service network, a manufacturing task network, and hyperedges between those two networks representing the correlations between service and task. Li, Dong, and Song (2012) proposed an intelligent service searching and matching method for cloud manufacturing services. A two-step matching strategy was proposed in which the candidate services were selected according to their type and status, and a maximum matching degree was pursued to match requests and services according to their functional and non-functional information. The similarities between different service bodies were calculated based on their semantic distances. This method tends to match complicated services and requests with hierarchical properties, and the similarities are dependent on subjective definitions. In

this paper we consider many parallel constraints without hierarchical requirements incurred by buyers or sellers. Additionally, due to the large number of game players involved, the solution space will be increased dramatically since it is dependent on the numbers of buyers and sellers. From the perspective of practice, an intelligent algorithm is required to obtain an optimal or near-optimal matching solution in a short time. The metaheuristic algorithms designed for handling large amounts of data can take on this role effectively.

Since the research related to matching problems is limited, the algorithms on the transportation problem are reviewed considering their similarities. For the transportation problem in a two-stage supply chain network, Molla-Alizadeh-Zavardehi, Hajiaghaei-Keshteli, and Tavakkoli-Moghaddam (2011) developed an artificial immune algorithm (AIA) and a GA based on the spanning tree and Prüfer number representation. The production scheduling and shipping lead time were studied in an integrated manner with a two-level genetic algorithm (Ma, Chan, and Chung 2013). Ghassemi Tari and Hashemi (2016a) developed a priority-based genetic algorithm (GA) for allocating vehicles to deliver goods from sources to destinations. A weight mapping crossover and swap mutation were proposed for the one-dimensional chromosome. They (Ghassemi Tari and Hashemi 2016b) also studied the transportation problem with the objective of cost minimization. The shipment volume from suppliers to customers was optimized by the genetic algorithm based on the spanning tree. Similarly, one-point and two-point crossovers were used, and mutation based on swaps was adopted for the genetic algorithm. A hybrid particle swarm algorithm was developed by (Ghassemi Tari and Hashemi 2016b) for the fixed charge transportation problem. A one-dimensional chromosome structure particle associated with artificial immune learning was used. To effectively reach the local optimum, Othman et al. (2011) employed fuzzy logic controllers to adopt the crossover parameters of the GA. Lee (2016) proposed a genetic algorithm (GA)-based optimization model to support anticipatory shipping that determines the allocation of products to different distribution centers with criteria of transportation cost and travel time.

The nondominated sorting genetic algorithm II (NSGA-II) (Deb et al. 2002) was developed for multi-objective optimization problems and has proven effective in many cases (Cao et al. 2013; Yağmur and Kesen 2022; Rohaninejad et al. 2021). Song and Chen (2018) proposed a knowledge-informed NSGA-II for the land allocation problem within a limited area considering both construction sprawl and land conservation. Yuan et al. (2021) improved the NSGA-II algorithm for machining

machines and workpieces with the aims of minimizing the maximum completion time, tardiness, machine load and energy consumption. Wang et al. (2021) developed an improved NSGA-II algorithm by integrating the k-means algorithm and local search strategy for matching demand and supply in the cloud manufacturing system from the user point of view. The NSGA-II algorithm was adopted in (Rezaei and Davoodi 2012) for a joint pricing, lot-sizing and supplier selection problem considering total profit, inconsistency, and deficiency with a set of constraints.

The studies concerning transportation problems are closely related to our research. However, they only consider the transportation cost, but the profits or requirements from both sides are not included. In contrast, we consider three objectives simultaneously, including the platform's profit, the sum of the sellers' profit, and the sum of the buyers' surplus utilities. Additionally, many constraints are included, such as buyer's preference for sellers' assessment, price, delivery due date, etc. Therefore, a more complicated and larger problem is studied. Accordingly, the metaheuristic algorithm is found to be more suitable.

3. The problem description

In this paper, a capacity sharing platform serving m capacity buyers (with variables and parameters indexed by i) and n capacity sellers (with variables and parameters indexed by j) for a specific product is considered. The matching solution is optimized for maximizing not only the platform's profit but also the sellers' and/or buyers' utilities, achieving targeted support for the development of buyers and sellers. Therefore, a multi-objective optimization problem is studied in this paper. In addition, the problem size with m and n is not predefined so that many possible scenarios can be considered. The notations are given in Table 1.

Table 1 Notations

The seller	
$j = 1,2,\dots,n$	Index of capacity seller
S_j	The capacity provided by seller$j$
ST_j	Delivery time committed to by seller$j$
RA_j	Rating/satisfaction for seller$j$
O_j	The cumulative transaction volume of seller$j$

MOQ_j	Minimum order quantity of seller$j$
P_{jmax}	The maximum price set by seller$j$
P_{jmin}	The minimum acceptable price set by seller$j$
c_j	The per-unit capacity cost of seller$j$
The buyer	
$i = 1,2, \dots, m$	Index of capacity buyer
D_i	The demand for capacity by buyer$i$
DT_i	Delivery time required by buyer$i$
P_{imax}	The maximum acceptable price set by buyer$i$
U_{ij}	The score of the seller j evaluated by buyer$i$
U_i	The reference score of buyer i
R_X	Rank for the factor X, with$X = p_{ij}, RA_j, O_j, ST_j$
β_X	Weights for the factor X, with$X = p_{ij}, RA_j, O_j, ST_j$
The platform	
θ	Service charge (proportion of transaction price)
c_p	Service cost per-unit capacity
Q_{ij}	The volume of capacity from the seller to the buyer (decision variable)

3.1 The capacity seller

For a specific product, we consider n competing sellers to publish the information about the price range, minimum order quantity, etc., and wait for the buyers' choice. Like most sellers on the platform, they generally only post their information waiting for buyers to choose and do not actively select buyers. The sellers will accept all the buyers' orders satisfying the primary conditions, including the price range, minimum order quantity, etc. As sellers often publish a price range, the capacity price is a function of the order quantities, which is similar to a price discount contract. More specifically, the transaction price is given as $p_{ij} = \max\{P_{jmax} - bQ_{ij}, P_{jmin}\}$, where parameter b denotes the price sensitivity to the orders. An example of the seller's information is given in Figure 1.

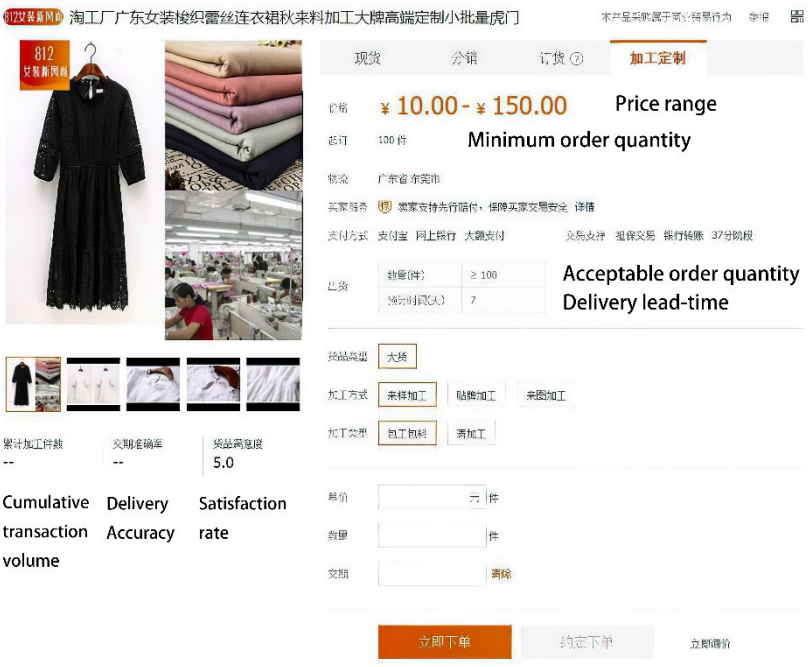


Figure 1 Caption: An example of the seller's information on the 1688 platform

Figure 1 Alt text: Photograph of the 1688 platform that publishes seller's demand information online

3.2 The capacity buyer

For the same product, we consider m capacity buyers to search matches on the platform. The price or expected profit is not the only factor considered by the buyer. In fact, there are many factors, such as the delivery time, rating, and cumulative transaction volume, that determine a buyer's choice. Moreover, each buyer values these factors differently. Some focus on the price, while others may regard rating as necessary. Meanwhile, the cumulative transaction volume, which indicates a seller's service capability and experience, will be another essential reference for the choice of the buyer. We assume that there exist basic requirements for the rating and cumulative transaction volume for each buyer. The buyer only ranks and scores those sellers who meet the basic requirements. Therefore, when we model the buyer's choice by a scoring system, for buyer i, only n_{1i} $(n_{1i} \leq n)$ of n sellers meet the basic requirements and will be considered by buyer i. Moreover, for these n_{1i} sellers, buyer i can only see n_{2i} $(n_{2i} \leq n_{1i})$ of them and then rank and score the selected seller's price, rating, and other factors. All these factors will be linked by a set of weights, which are specific to each buyer. The score of seller j evaluated by buyer i is given as

$$U_{ij} = \beta_1 R_{p_{ij}} + \beta_2 R_{RA_j} + \beta_3 R_{O_j} + \beta_4 R_{ST_j}, \quad (1)$$

where $R_X(X = p_{ij}, RA_j, O_j, ST_j)$ is the rank of factor X for seller j. For example, with n_{2i} sellers' delivery time, seller j_1's delivery time ST_{j_1} is the second shortest. Then, $R_{ST_{j_1}} = 2$. With a given reference score U_i, each buyer will choose or accept the seller with the score to be larger than the reference score, i.e., $U_{ij} \geq U_i$. Buyer satisfaction will increase in the difference between the matching seller's score and the reference score, which is represented by $U_{ij} - U_i$.

3.3 The platform

The platform holds some transaction information of sellers and buyers and can obtain necessary data from them to assist in decision-making. Since the platform provides matching recommendations to the buyer, it suggests information such as the buyer's factor preferences to provide more targeted matching recommendations. For the buyer, he/she can choose whether to accept the recommendation and provide preference information. If the buyer does not want to provide preference information, the platform will make recommendations for the buyer based on equal preferences.

Serving the capacity buyer and seller, the platform charges a service fee per unit capacity from the seller. The service charge is a percentage θ of the transaction price p_{ij}, i.e., θp_{ij} per unit capacity. The decision variables are represented by the capacity volume Q_{ij} shared between buyer i and seller j. The recommendation scheme achieves multiple objectives, including the platform's profit, the sum of the sellers' profit, and the sum of the buyers' surplus utilities. The reason we do not use buyers' profits is that buyers who choose a seller usually focus on a variety of factors, including delivery rates, lead times, and more, not just prices or profits.

Given the weight $\alpha_1 \sim \alpha_3$ for each subobjective, the objective function is given as:

$$\max_{Q_{ij}} y = \alpha_1 \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{n}(\theta p_{ij} - c_p)Q_{ij}}_{platform's\ profit} + \alpha_2 \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{n_{2i}}(U_{ij} - U_i)}_{buyers'\ surplus\ utilities}$$
$$+ \alpha_3 \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{n}\left(p_{ij}(1-\theta) - c_j\right)Q_{ij}}_{sellers'\ profits} \quad (2)$$

The constraints include the following:

$$\sum_{i=1}^{m} Q_{ij} \leq S_j \quad (3)$$

$$\sum_{j=1}^{n} Q_{ij} \leq D_j \quad (4)$$

$$Q_{ij} \geq MOQ_j \quad (5)$$

$$DT_i \geq ST_j \tag{6}$$

$$U_{ij} \geq U_i \tag{7}$$

$$Q_{ij} \geq 0 \tag{8}$$

In the objective function, the first term is the platform's profit and the corresponding weight. As advertising and other revenues are not directly related to the capacity matching mechanism, we consider that the platform's profit mainly comes from service charges and that the unit service cost is assumed to be zero. The second term is the sum of the buyers' surplus utilities and the corresponding weight. The third term is the sum of the sellers' profits and the corresponding weight.

Constraint (3) captures the fact that the sum of the capacities shared by the seller j should be less than or equal to its maximum capacity. Constraint (4) guarantees that the capacity recommended to buyer i is not larger than its demand. The unsatisfied capacity will be traded spontaneously. Constraint (5) means that the trading volume should be larger than the minimum order quantity of the seller. Constraint (6) ensures that seller j's delivery time can meet the requirement of buyer i. Constraint (7) requires the platform to select a match from a pool of sellers that the buyer is willing to accept. Constraint (8) ensures that all the capacities recommended are larger than or equal to 0.

4. Supply and demand matching algorithms

To optimize the multiple objectives in terms of the platform's profit, the sum of the sellers' profit, and the sum of the buyers' surplus utilities, new algorithms are proposed in this paper. The genetic algorithm is employed because it has global optimization capability in the presence of multiple local minima in the parameter space. The main operators that enable the GA to search globally and locally are selection, crossover and mutation. Therefore, the contribution of this study is to design new operators for the GA for the given problem. Since multiple objectives are considered simultaneously, the nondominated sorting genetic algorithm II (NSGA-II) is employed.

The core problem is to allocate the transaction volume between sellers and buyers. Therefore, the 2-dimensional (2-D) chromosome is adopted. Due to the large space of 2-dimensional chromosomes, effective crossover and mutation operators are needed. As shown in Figure 2, the 2-dimensional chromosome includes transaction volumes between each buyer and seller. In the chromosome, every row represents a buyer, while each column indicates sellers. The values in each block are the

transaction volume that has occurred. The red blocks with -1 represent that one of constraints (3)-(8) is violated so that the transaction cannot happen.

Buyers \ Sellers	1	2	3	4	5	6	7	8	9
1	21	12	41	19	25	24	23	14	-1
2	11	-1	13	14	-1	0	30	8	0
3	8	20	-1	42	17	-1	7	-1	-1
4	27	-1	32	0	20	6	24	33	0
5	34	34	29	-1	0	-1	7	-1	23
6	18	22	10	0	25	16	35	31	20
7	8	2	19	0	30	16	20	16	19

Figure 2 Caption: A sample of a 2-dimensional chromosome (the part with gray background)

Figure 2 Alt Text: Photograph of a two-dimensional chromosome with red and gray blocks, where the gray blocks are transaction volumes between buyers and sellers, and the red blocks are nonexistent.

4.1 2-D crossover

To conduct an effective search, six types of crossovers are used, as shown in Figure 3and Figure 4. For the Row Crossover, a random point in terms of rows is selected for both parent chromosomes, and the lower portions of the 2-d chromosome from both parents are selected and exchanged so that two new children are generated. A similar operation is employed in terms of column in Column Crossover. In the multirow crossover and multicolumn crossover, a random number of rows or columns are selected and exchanged with two parents. To enlarge the search diversity, the main diagonal and minor diagonal crossovers are designed. With the main diagonal crossover, the top right triangle parts from both parents are exchanged, while with the minor diagonal, the top left triangle parts are exchanged. Note that with six types of crossovers as alternatives in the new genetic algorithms, the crossovers are implemented in each run according to the probabilities allocated.

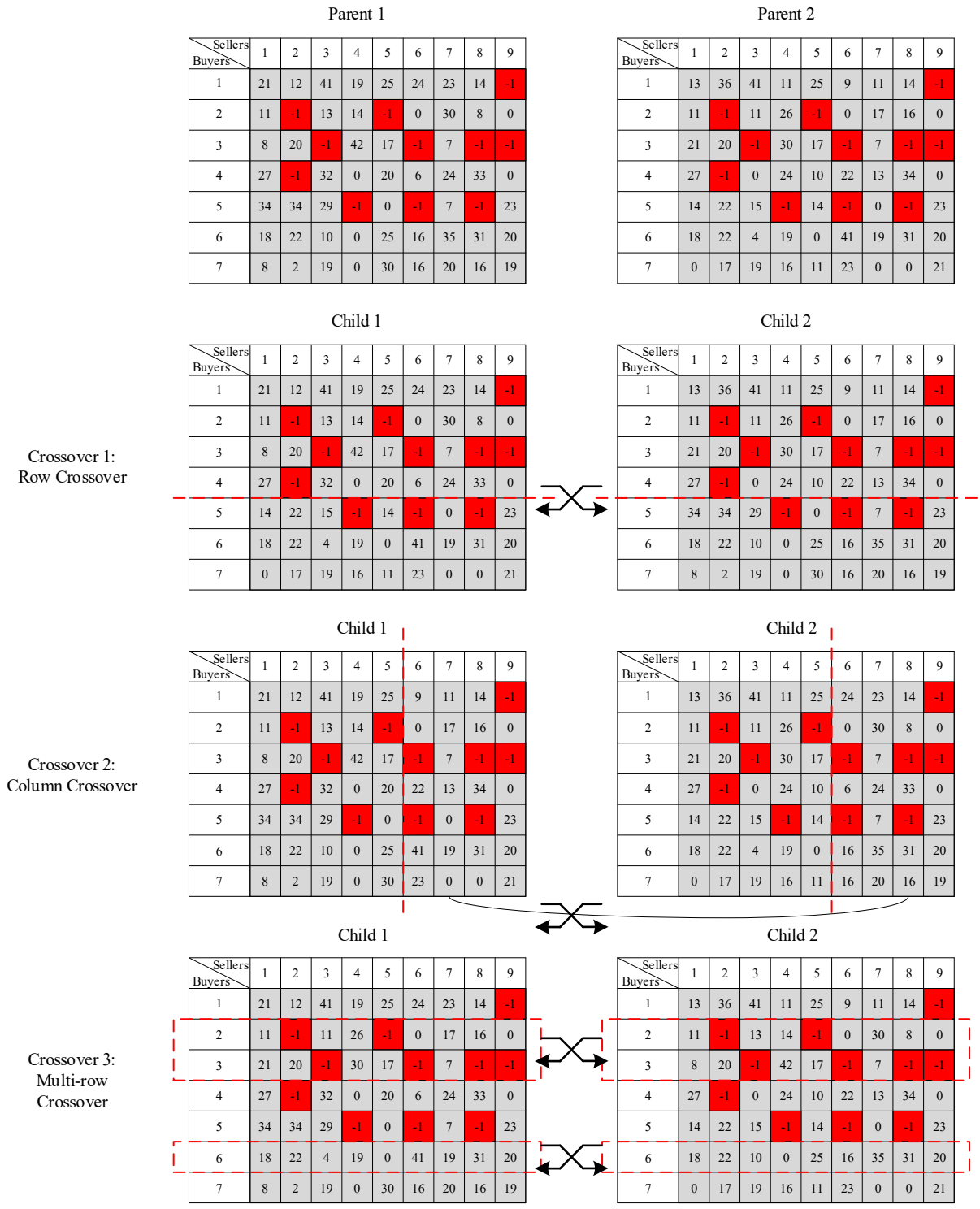


Figure 3 Caption: Sample demonstration Crossover 1-3

Figure 3 Alt Text: Photograph of row, column and multirow crossovers

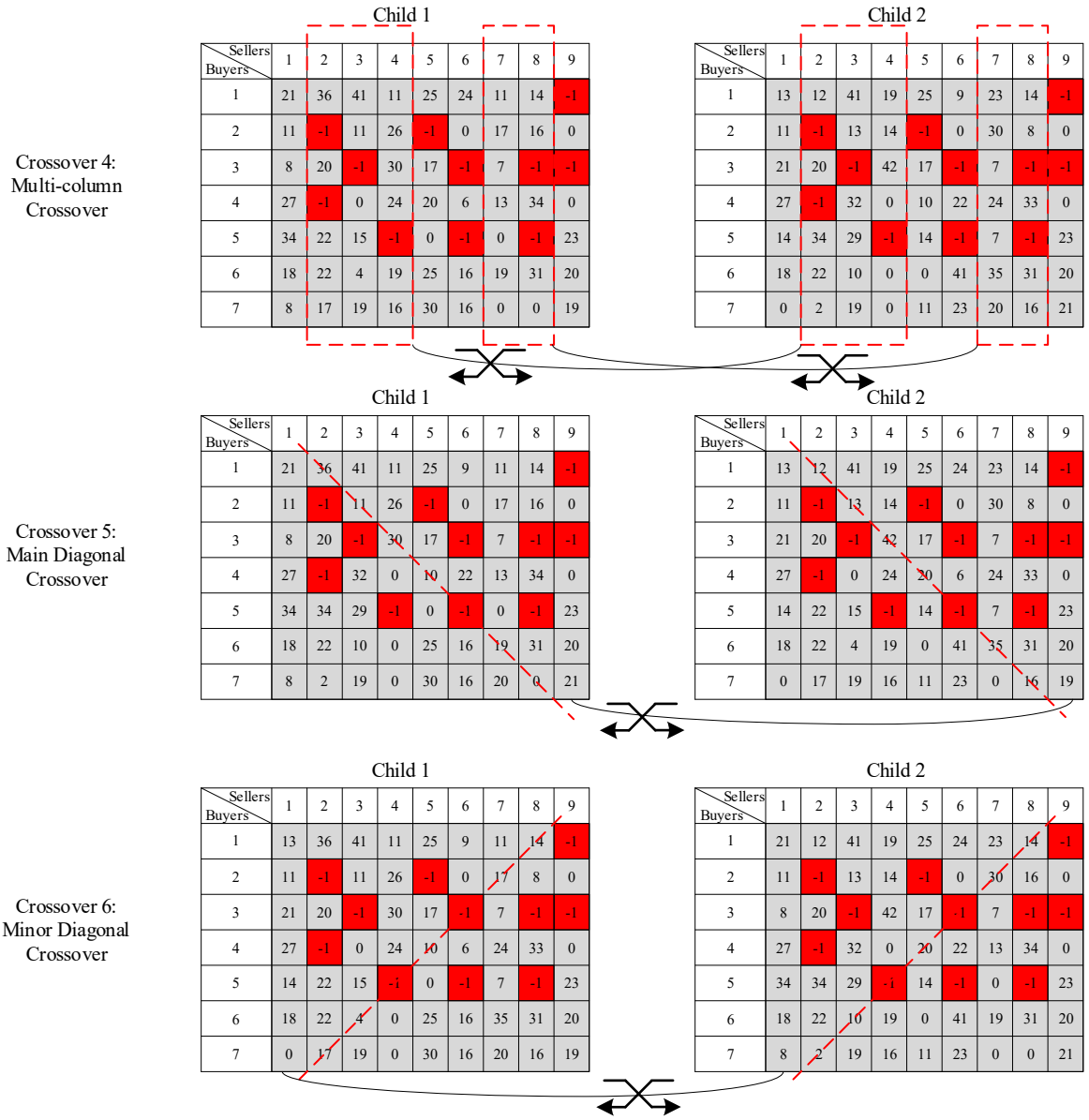


Figure 4 Caption: Sample demonstration for Crossover 4-6

Figure 4 Alt Text: Photograph of multicolumn, main diagonal and minor diagonal crossover

4.2 Order-first mutation

Mutation as a genetic operator maintains genetic diversity from one generation to the next. In this operation, one or more gene values in a chromosome from its initial state may be changed in each run of mutation. It is also a critical type of operator to conduct a local search for GAs. Concerning the 2-dimensional chromosome, a powerful mutation is needed. Additionally, since the profit of each order from buyers and platform is counted, only the total demand is satisfied. Therefore, herein, an order-first mutation is introduced. The rationale behind the order-first mutation is to fix the gap between the

sum of transaction volumes and demand. Therefore, if there is a demand gap for each buyer, the maximum or the random transaction volume will be enlarged to bridge the gap.

As two matching schemes are considered in this paper in which the order can be split or not, two mutations are designed. With split orders, the random value in each row is replaced by the gap between the buyer's demand and the sum of transaction volume so that the buyer's demand is satisfied. As shown in Figure 5, the three buyers' demands are not completely satisfied. Therefore, for these rows, the value from a random position (with green background) is replaced and enlarged to bridge the demand gap.

Before mutation (Order split)

Sellers / Buyers	1	2	3	4	5	6	7	8	9	Sum	Buyer Demand	Demand Satisfied?
1	21	12	41	19	25	24	23	14	-1	179	182	<
2	11	-1	13	14	-1	0	30	8	0	76	70	>
3	8	20	-1	42	17	-1	7	-1	-1	94	90	>
4	27	-1	32	0	20	6	24	33	0	142	160	<
5	34	34	29	-1	0	-1	7	-1	23	127	127	=
6	18	22	10	0	25	16	35	31	20	177	177	=
7	8	2	19	0	30	16	20	16	19	130	140	<

After mutation (Order split)

Sellers / Buyers	1	2	3	4	5	6	7	8	9	Sum	Buyer Demand	Demand Satisfied?
1	21	15	41	19	25	24	23	14	-1	182	182	=
2	11	-1	13	14	-1	0	30	8	0	76	70	>
3	8	20	-1	42	17	-1	7	-1	-1	94	90	>
4	27	-1	50	0	20	6	24	33	0	160	160	=
5	34	34	29	-1	0	-1	7	-1	23	127	127	=
6	18	22	10	0	25	16	35	31	20	177	177	=
7	8	2	19	0	30	26	20	16	19	140	140	=

Figure 5 Caption: Sample demonstration for order-first mutation with order split scheme

Figure 5 Alt Text: Photograph of newly designed mutation where orders are allowed to be split

For the nonsplit order scheme, the maximum value in each row is replaced by the buyer's demand directly, and the other values are set as 0, as shown in Figure 6. Note that for the scenarios with transactions over buyers' demand, the rows remain unchanged, and the repair mechanism is activated. The new repair mechanism is detailed in the next subsection.

Before mutation (Non-order split)

Sellers / Buyers	1	2	3	4	5	6	7	8	9	Sum	Buyer Demand	Demand Satisfied?
1	179	0	0	0	25	0	0	0	-1	204	182	>
2	0	-1	0	40	-1	0	30	0	0	70	70	=
3	0	40	-1	42	0	-1	0	-1	-1	82	90	<
4	0	-1	0	0	142	0	0	0	0	142	160	<
5	34	0	0	-1	0	-1	0	-1	83	117	127	<
6	0	0	177	0	0	0	0	0	0	177	177	=
7	0	120	0	0	30	0	0	0	0	150	140	>

Before mutation (Non-order split)

Sellers / Buyers	1	2	3	4	5	6	7	8	9	Sum	Buyer Demand	Demand Satisfied?
1	179	0	0	0	25	0	0	0	-1	204	182	>
2	0	-1	0	40	-1	0	30	0	0	70	70	=
3	0	0	-1	90	0	-1	0	-1	-1	90	90	=
4	0	-1	0	0	160	0	0	0	0	160	160	=
5	0	0	0	-1	0	-1	0	-1	127	127	127	=
6	0	0	177	0	0	0	0	0	0	177	177	=
7	0	120	0	0	30	0	0	0	0	150	140	>

Figure 6 Caption: Sample demonstration for order-first mutation with nonsplit order scheme

Figure 6 Alt Text: Photograph of newly designed mutation where orders are not allowed to split

4.3 Solution repair mechanism

Since many constraints are considered in the problem, such as buyer's demand, minimum order quantity of the seller, sellers' capacities, etc., many infeasible solutions may be generated after crossovers and mutations. Hence an effective repair mechanism is needed.

With split orders, three steps are included in the repair mechanism, as shown in Table 2. In step 1, we check if the MOQ for each seller is satisfied. If not, the transaction volume is redefined as the MOQ of each seller. The buyers' demand is verified in step 2. If the sum of transaction volume is more than buyers' demand, then the max is redefined as the gap between the demand and the sum of the remaining volumes. Step 3 verifies the sellers' capacities. If it exceeds sellers' capacity, the max is replaced by the gap between the capacity and the sum of the remaining volumes. As shown in Table 3, for the nonsplit order scheme, an extra step is added that if two transactions occur with one buyer, then the lower volume is set as 0 and the upper volume is defined as the buyer demand directly.

Table 2 Repair mechanism with split orders

Repair Mechanism (Split order)
Input: Infeasible solution
Step 1: Check each transaction volume and decide if MOQ is satisfied If not, define each volume as the MOQ; **Step 2:** Check if the transactions are over buyers' demand If yes, the max is reset as the gap between the demand and the sum of the rest volume (in terms of row); **Step 3:** Check if the transactions are over sellers' capacity If yes, the max is replaced by the gap between the capacity and the sum of the rest volume (in terms of column);
Output: Feasible solution

Table 3 Repair mechanism with nonsplit orders

Repair Mechanism (Non-Split order)
Input: Infeasible solution
Step 1: Check each transaction volume and decide if MOQ is satisfied If not, define each volume as the MOQ; **Step 2:** Check if the transactions are over buyers' demand If yes, the max is reset as the gap between the demand and the sum of the rest volume (in terms of row); **Step 3:** Check if there are two or more transactions in each row of the solution If yes, reset the lower value as 0 and upper value as the buyer's demand; **Step 4:** Check if the transactions are over sellers' capacity If yes, replace it with the gap between the capacity and the sum of the rest volume (in terms of column);
Output: Feasible solution

Customer	Provider							Demand
	1	2	3	4	5	6	7	
1	0	0	50	0	50	0	0	100
2	0	0	50	70	0	0	0	120
3	38	60	0	-1	-1	0	0	98
4	-1	-1	79	-1	0	0	0	79
5	0	46	-1	-1	-1	0	-1	46
6	0	42	0	0	0	44	0	86
7	-1	-1	142	-1	0	0	0	142
8	0	0	0	50	0	123	0	173
9	0	0	-1	-1	168	0	0	168
10	0	0	0	-1	0	0	254	254
Capacity	325	400	510	360	465	640	220	
MOQ	20	30	50	60	50	30	60	

Infeasible solution

Repair

Customer	Provider							Demand
	1	2	3	4	5	6	7	
1	0	0	50	0	50	0	0	100
2	0	0	50	70	0	0	0	120
3	38	60	0	-1	-1	0	0	98
4	-1	-1	79	-1	0	0	0	79
5	0	46	-1	-1	-1	0	-1	46
6	0	42	0	0	0	44	0	86
7	-1	-1	142	-1	0	0	0	142
8	0	0	0	60	0	113	0	173
9	0	0	-1	-1	168	0	0	168
10	0	0	0	-1	0	0	220	254
Capacity	325	400	510	360	465	640	220	
MOQ	20	30	50	60	50	30	60	

Repaired solution

Figure 7 Caption: An example of infeasible solution repair for order split scenario

Figure 7 Alt Text: Photograph of infeasible solution repair process under order split scenario

Figure 7 shows an infeasible solution repairing process for the order split scheme. According to the repair mechanism, the transaction volume of each column will be checked if the MOQ is satisfied. In this example, the 4th provider is recommended to produce 50 units of product for the 8th customer, violating the MOQ constraint of provider 4. So, the transaction volume, 50, is replaced by 60. Then, the customer demand will be confirmed. In this situation, the customer demand is over-satisfied because 60+123=183, which is larger than 173. Therefore, the transaction volume of 123 is reduced to 113. Once all MOQs are checked and satisfied, the provider capacity is then examined. For example, the 7th provider is advised to produce 254 units of product which are over its capacity. Therefore, 254 is updated with 220. If all provider capacities, MOQs and customer demands are verified and amended, the infeasible solution will be transformed into a feasible one.

The repairing process for an infeasible solution under the order nonsplit scheme is shown in Figure 8. Similar to the order split scheme, the MOQ constraint and customer demand will be checked first. If these two constraints are satisfied, the order will be checked if the order is split. Since the GA operators are employed, crossover and mutation may generate solutions with order split. In this example, the 10th customer order is completed with two providers, the 2nd and 7th providers. Under this situation, the upper volume, 200, is increased to 254 which is the 7th customer maximum demand, and the lower one is set as 0. Next, all provider capacities will be checked, and the transaction volumes are to be examined. The repair procedure will end until all constraints are satisfied.

Customer	Provider							Demand
	1	2	3	4	5	6	7	
1	0	0	0	100	0	0	0	100
2	0	0	0	0	120	0	0	120
3	98	0	0	-1	-1	0	0	98
4	-1	-1	79	-1	0	0	0	79
5	0	46	-1	-1	-1	0	-1	46
6	0	86	0	0	0	0	0	86
7	-1	-1	0	-1	0	0	142	142
8	0	0	0	0	173	0	0	173
9	0	0	-1	-1	168	0	0	168
10	0	200	0	-1	0	0	54	254
Capacity	325	400	510	360	465	640	220	
MOQ	20	30	50	60	50	30	60	

Infeasible solution

Repair

Customer	Provider							Demand
	1	2	3	4	5	6	7	
1	0	0	0	100	0	0	0	100
2	0	0	0	0	120	0	0	120
3	98	0	0	-1	-1	0	0	98
4	-1	-1	79	-1	0	0	0	79
5	0	46	-1	-1	-1	0	-1	46
6	0	86	0	0	0	0	0	86
7	-1	-1	0	-1	0	0	142	142
8	0	0	0	0	173	0	0	173
9	0	0	-1	-1	168	0	0	168
10	0	254	0	-1	0	0	0	254
Capacity	325	400	510	360	465	640	220	
MOQ	20	30	50	60	50	30	60	

Repaired solution

Figure 8 Caption: An example of infeasible solution repair for order nonsplit scenario

Figure 8 Alt Text: Photograph of infeasible solution repair process under order nonsplit scenario

4.4 GA- and NSGA-II-based algorithms

Based on the new crossover and mutation operations, two GA-based algorithms are developed. The frameworks of each algorithm are given in Table 4 and Table 5.

With the traditional GA-based algorithm, the initial populations are generated first. Then, the roulette wheel selection method is adopted for choosing good performance solutions as the parent population for crossover and mutation. Within the crossover operator, one type of new crossover is selected according to the predefined probabilities if the crossover probability is satisfied. The output of the crossover is input into mutation, and mutation is triggered if the mutation probability is met. Finally, the population is updated after crossover. To save the best chromosome, the worst chromosome is replaced by the best chromosome in each iteration. The iteration will stop until the time limit is reached.

Since NGSA-II shares high performance with multiple-objective optimization problems, it is studied and employed in this paper. NSGA-II has a similar framework to the traditional GA except for the selection method in which the nondominated sorting is used. With the nondominated sorting method, the population is categorized and ranked according to the dominant relationship among solutions. Due to the crowding distance employed in the nondominated sorting method, diverse solutions are assigned high probabilities to be chosen for the next generation. Therefore, the population diversity can be enhanced with the NSGA-II algorithm. Among all populations, the final solution is

selected according to the weighted objective value. Note that in the NSGA-II-based algorithm, the new population is updated with the outstanding ones from the combined parent and children populations. Herein, the nondominated sorting is used again.

Table 4 GA-based algorithm with new operators

GA based algorithm
Input: the numbers of buyers and sellers, buyer requirements namely, demand, due date, price range etc., seller requirements such as capacity, rating rankings, MOQ etc., population size, crossover probability, mutation probability etc.
Population Initilization While running time<=time limit Operator 1: roulette wheel selection according to the weighted objectives Operator 2: crossover → repair mechanism Operator 3: mutation → repair mechanism Update new population End
Output: The best solution

Table 5 NSGA-II-based algorithm with new operators

NSGA-II based algorithm
Input: the numbers of buyers and sellers, buyer requirements namely, demand, due date, price range etc., seller requirements such as capacity, rating rankings, MOQ etc., population size, crossover probability, mutation probability etc.
Population Initilization While running time<=time limit Operator 1: tournament selection according to the nondomination sort Operator 2: crossover → repair mechanism Operator 3: mutation → repair mechanism Update new population with the nondomination sort method by combining parent and children populations End
Output: The best solution

5. Analysis of the results

5.1 Algorithm parameter setting

To investigate the performance of the new algorithms, random problems were generated. The problem parameters include the number of buyers and the number of sellers. To determine the size of the problems, the number of sellers and buyers were set as 20 and 15, respectively. Three combinations were selected, including [20-20], [20-15] and [15-20]. The front value is the quantity of buyers, and the back value is the quantity of sellers. Regarding the sellers and buyers, some properties are needed, such as buyer demand, seller capacity, delivery time, etc. For example, the seller capacities are uniformly distributed in the interval of [200, 1000]. For a fair performance comparison, all algorithms were coded in MATLAB R2020a and run on an Intel Core CPU i9-9900K computer with 32G memory. The time limit for each problem or algorithm was defined as 300 seconds. The detailed parameter settings of the problems are shown in Table 6.

Table 6 Parameter setting for random problems

Problem parameter	**Value**
Number of sellers	20, 15
Number of buyers	20, 15
Seller capacity	U[200, 1000]
MOQ	U[10, 30]
Delivery time	U[1, 3.5]
Seller max price	U[30, 60]
Seller min price	U[10, 30]
The cumulative transaction volume	U[1000, 1400]
Buyer demand	U[200, 1000]
Buyer delivery time required	U[3, 9]
The maximum acceptable price by buyers	U[25, 40]
The reference score of buyer	U[1, 25]

Table 7 Parameter settings for both algorithms

Hyperparameters	**GA (Split)**	**GA (Nonsplit)**	**NSGA-II (Split)**	**NSGA-II (Nonsplit)**
Population size	200	2500	200	2000
Crossover probability	0.8	0.8	0.8	0.8
Mutation probability	0.3	0.3	0.3	0.3
Probabilities for each type of crossover	0.3, 0.2, 0.1, 0.3, 0.05, 0.05	0.3, 0.2, 0.1, 0.3, 0.05, 0.05	0.3, 0.2, 0.1, 0.3, 0.05, 0.05	0.3, 0.2, 0.1, 0.3, 0.05, 0.05

Table 7 lists a number of parameters used in all algorithms, namely, the population size, crossover probability, mutation probability, and crossover probabilities. An orthogonal experimental design with three factors and five levels was conducted to determine the population size, crossover probability, and mutation probability. For example, five levels of the population size for the order split scenario include 100, 200, 300, 400, and 500. For the order nonsplit scenario, all five levels of population size are set as 1000, 1500, 2000, 2500, and 3000 respectively. Then, the probabilities for each type of crossover were decided based on the parameter combination test. Different probability combinations of each crossover were tested, and the final probabilities were defined as 0.3, 0.2, 0.1, 0.3, 0.05 and 0.05. In our experiments, a high crossover probability was adopted as the 2-dimensional chromosome is used as the solution space is larger than the 1-dimensional chromosome. Therefore, a high probability of crossover is required to enhance its search ability, as well as the mutation probability.

To conduct a fair performance comparison, several existing algorithms, including the gray wolf optimizer (GWO), particle swarm optimization (PSO) and grasshopper optimization algorithm (GOA), were considered.

5.2 Discussion of results

We considered three cases, (1) more sellers than buyers, (2) more buyers than sellers and (3) an equal number of sellers and buyers, and to cover all possible scenarios, the values of m and n were defined as given in Table 8.

Table 8 The number of buyers and sellers

m	n	Cases
20	15	More sellers than buyers
15	20	More buyers than sellers
20	20	Equal number of sellers and buyers

In each case, we gave different weights to the three sub-objectives to distinguish between the different decision-making objectives of the platform. More specifically, the weight combinations and the corresponding explanations are given in Table 9.

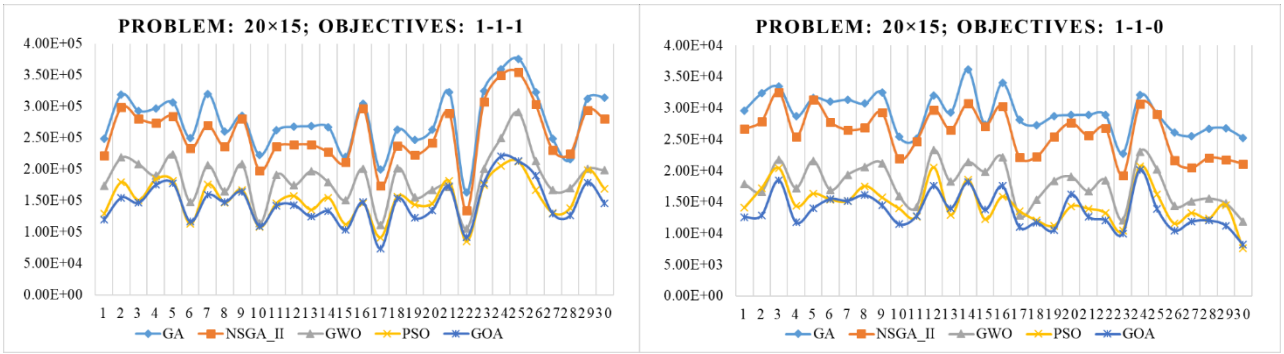
Table 9 The values of weights and explanations

α_1	α_2	α_3	Explanation
1	0	0	Maximizing the platform's profit
1	1	0	Encouraging the development of buyers
1	0	1	Encouraging the development of sellers
1	1	1	Encouraging the development of both buyers and the sellers

Based on Table 8 and Table 9, 12 experiments were simulated, each repeated 30 times (each with random value). For example, the case with title "20-15,1-1-0" represents the case with 20 sellers and 15 buyers and a platform maximizing its own profits and the buyers' surplus utilities. We analyzed two kinds of matching schemes, split one buyer's order and nonsplit one buyer's order.

5.2.1 Split order scheme

In this case, one buyer's order can be split and allocated to two or more sellers. The two new algorithms (GA and NSGA) were employed to optimize the solution and compared. The results of the experiments are shown in Figure 9, Figure 10, and Figure 11, and more detailed results have been published online. Readers can access the original data with Mendeley Data, doi: 10.17632/sx88drkkth.2.



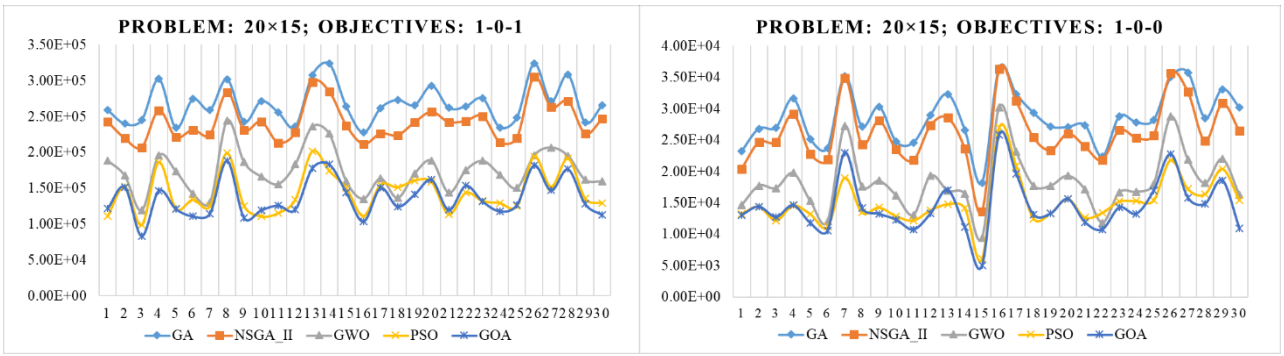
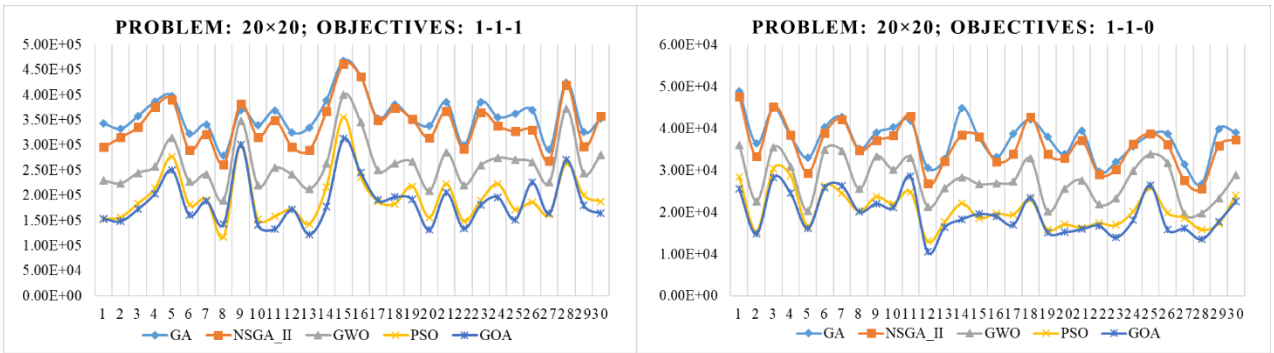


Figure 9 Caption: Algorithm performance comparison with the problem of 20×15

Figure 9 Alt Text: Photograph of algorithm performance comparison with the problem in terms of 20 sellers and 15 buyers in the scenario of order split



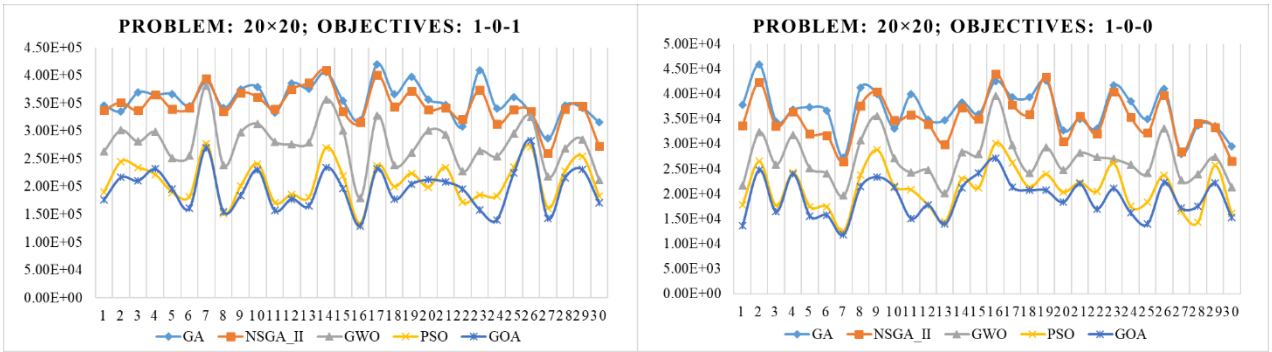


Figure 10 Caption: Algorithm performance comparison with the problem of 20×20

Figure 10 Alt Text: Photograph of algorithm performance comparison with the problem in terms of 20 sellers and 20 buyers in the scenario of order split

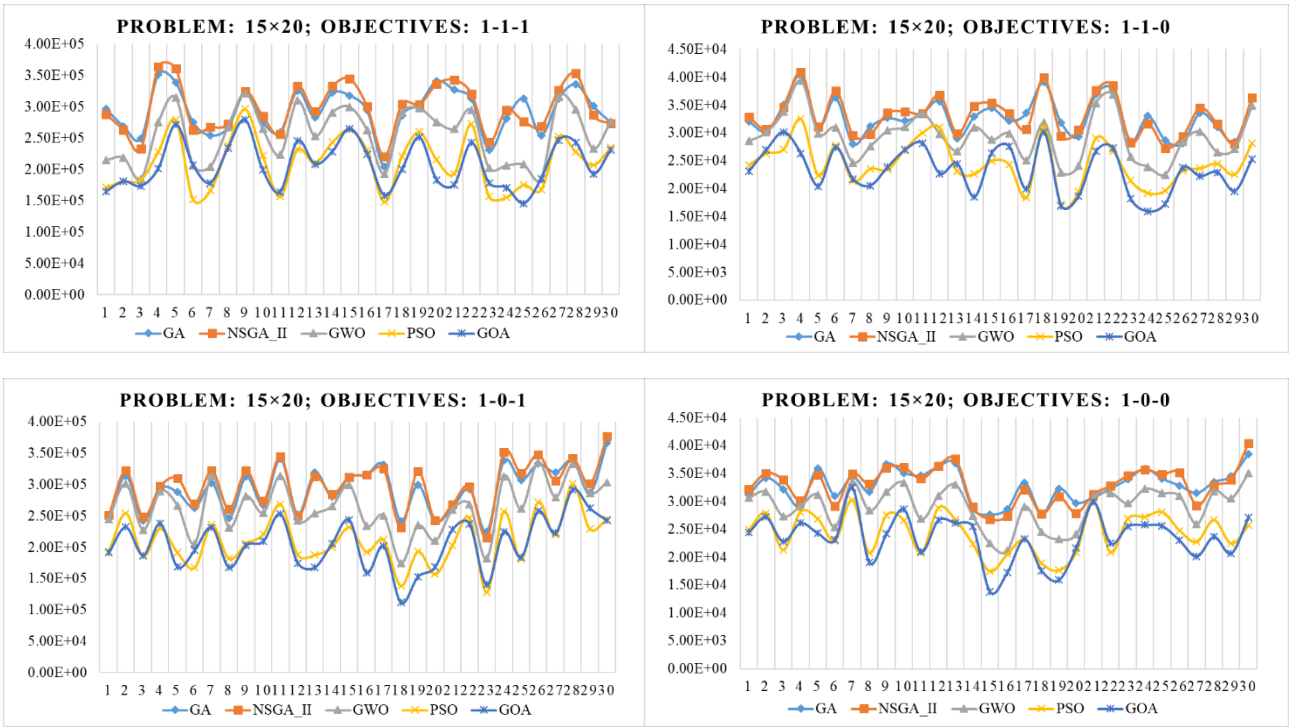


Figure 11 Caption: Algorithm performance comparison with the problem of 15×20

Figure 11 Alt Text: Photograph of algorithm performance comparison with the problem in terms of 15 sellers and 20 buyers in the scenario of order split

Although the problems were randomized for each experiment, the performance comparison between these two algorithms reflects a particular trend. Figures 9, 10, and 11 show that employing the newly designed crossover and mutation mechanism, the GA and NSGA-II clearly perform better than the reference algorithms GWO, PSO, and GOA on all three different problems. The GWO algorithm is at the second level, followed by PSO and GOA.

The performance difference between the GA and NSGA-II is that in the “more buyers than sellers” case, the NSGA-II algorithm has better performance than the GA algorithm with a high probability; in the “more sellers than buyers” and “equal numbers of buyers and sellers” case, the GA algorithm has better performance than the NSGA-II algorithm with a high probability. This is because in the GA-based algorithm, the weighted objectives are directly employed to select excellent offspring so that it shows better performance in 2/3 of the cases. Concerning the 15-20 case, the number of sellers is larger than the number of buyers, which results in a large selection space for buyers. In this case, the nondominated sorting method plays a key role in selecting the Pareto optimal solution, which outperforms the others with different objectives. Therefore, the NSGA-II-based algorithm provides

better matching solutions for the 15-20 case.

5.2.2 Nonsplit order scheme

Considering that buyers may not be willing to split their orders for convenience or that sellers may only be able to serve one buyer at a time, the “one-buyer-one-seller” matching scheme was considered. Both GA- and NSGA-II-based algorithms were adopted to generate optimal matching schemes. Herein, due to the similar results from the reference algorithms, only the results of the PSO algorithm (denoted by PSO-Integ) are provided and compared with the GA and NSGA-II. The detailed results are available online with Mendeley Data, DOI: 10.17632/sx88drkkth.2.

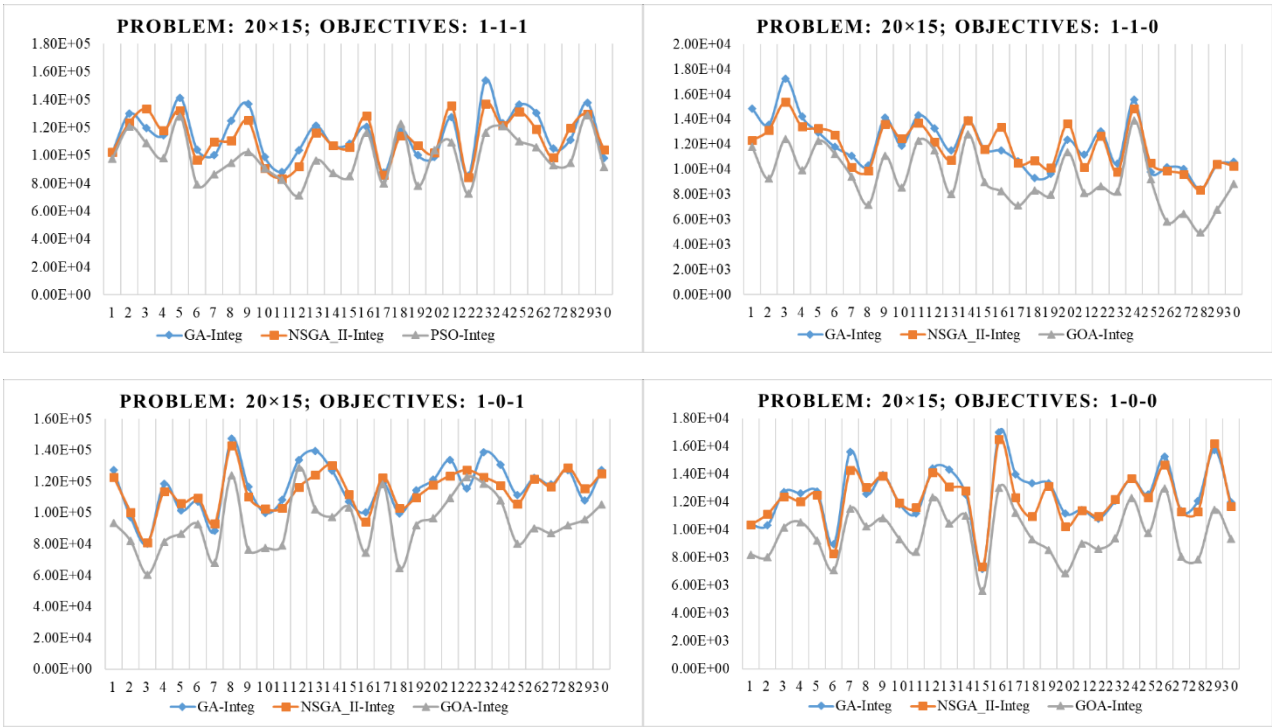


Figure 12 Caption: Algorithm performance comparison with the problem of 20×15

Figure 12 Alt Text: Photograph of algorithm performance comparison with the problem in terms of 20 sellers and 15 buyers in the scenario of order nonsplit

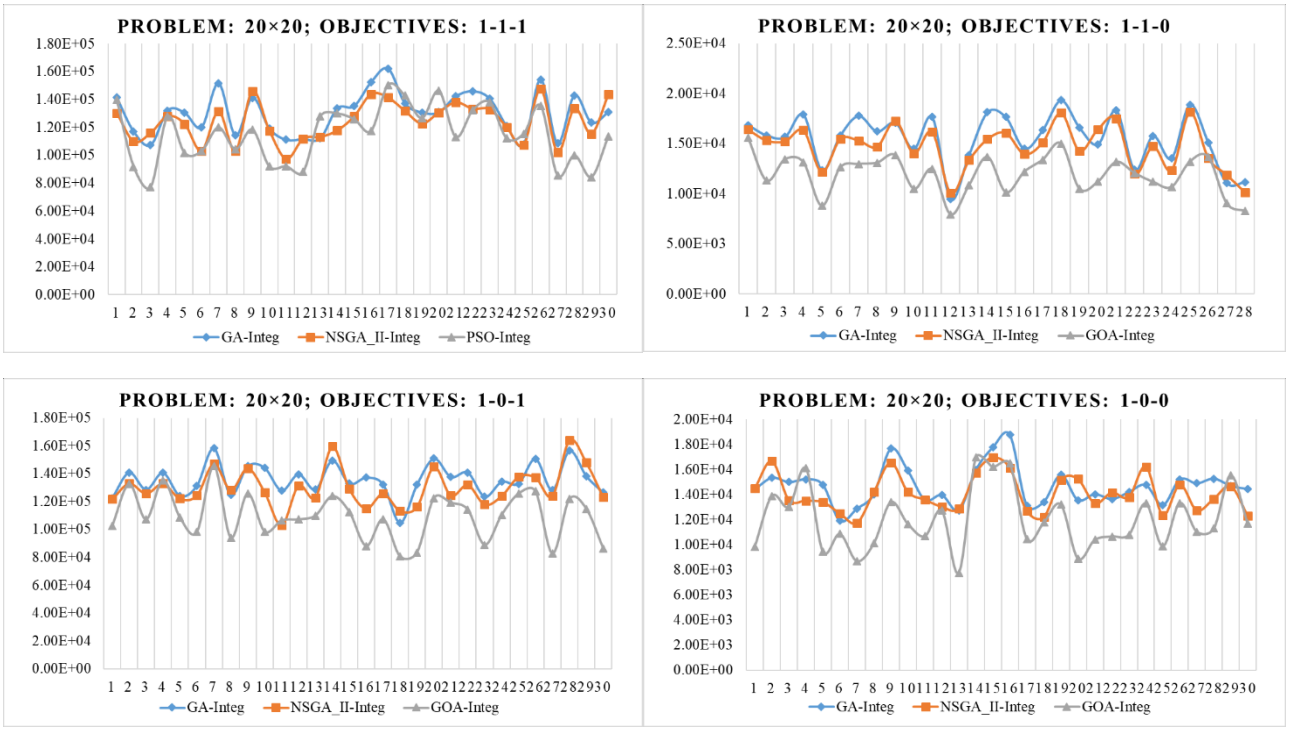


Figure13 Algorithm performance comparison with the 20×20 problem

Figure 13 Alt Text: Photograph of algorithm performance comparison with the problem in terms of 20 sellers and 20 buyers in the scenario of order nonsplit

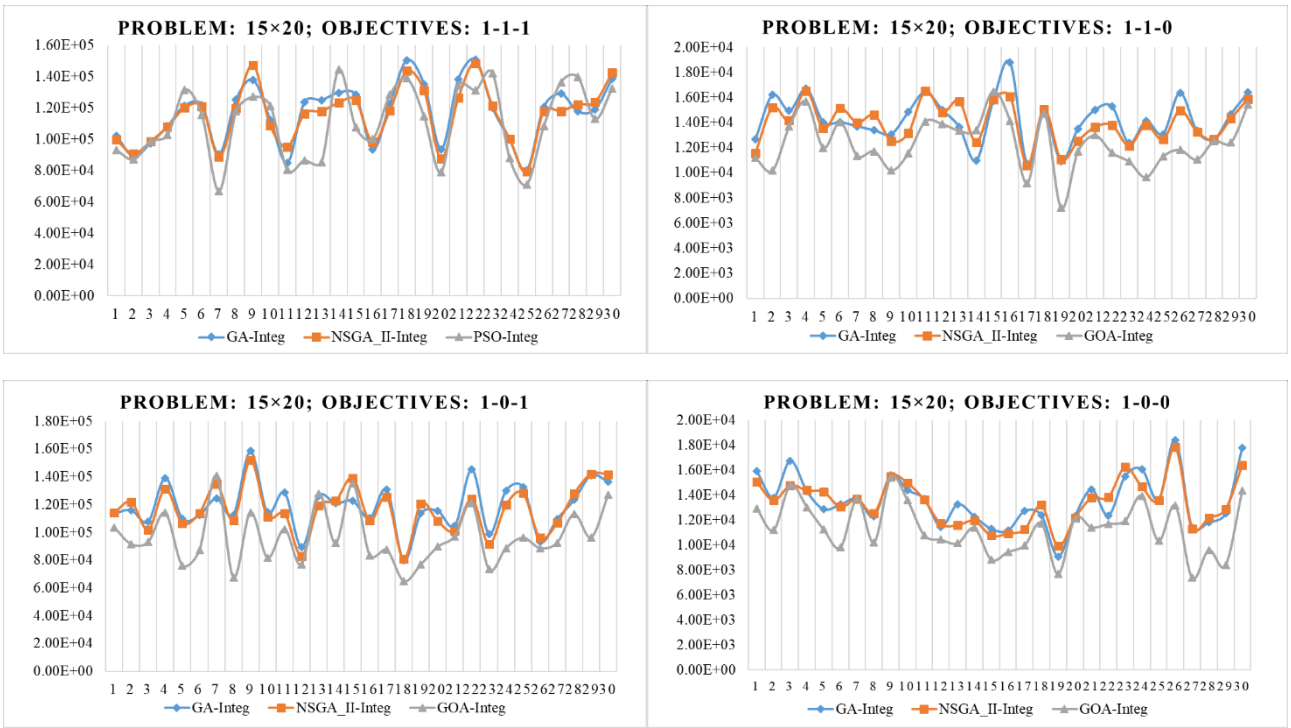


Figure 14 Caption: Algorithm performance comparison with the problem of 15×20

Figure 14 Alt Text: Photograph of algorithm performance comparison with the problem in terms of 15 sellers and 20 buyers in the scenario of order nonsplit

From Figures 12-14, it can be seen that GA and NSGA-II show significantly better performance than GOA with the nonsplit order scheme, especially with scenarios of 20×15 and 20×20. For the 15×20 scenario, the performance of all considered algorithms fluctuates when each objective is defined as equal weight.

Obviously, the performance of the nonsplit matching scheme is worse than that of the split matching scheme with 15-20, 20-15, and 20-20 cases, as shown in Figure 15. This is because the order-split scheme allows buyers' demand to be satisfied by several sellers with high profits. Interestingly, the results show that the performance gap between these two matching schemes decreases when more manufacturers join the capacity sharing platform. The reason is that the buyers own a larger selection space to optimize order allocation for maximizing the overall profits.

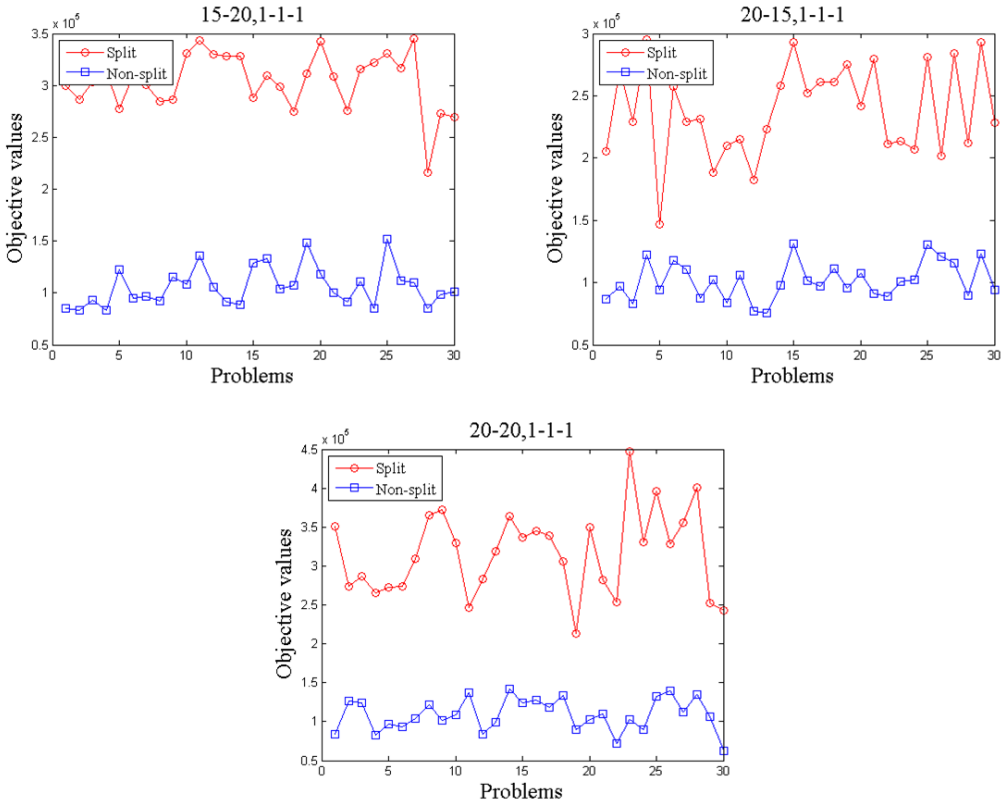


Figure 15 Caption: Performance comparison of the GA-based algorithm with order split and nonsplit cases

Figure 15 Alt Text: Photograph of GA-based algorithm performance comparison with order split and nonsplit cases

6. Conclusions

In this paper, the matching problem for manufacturing capacity sharing is studied to optimize the imbalance of supplies and demand. Seven constraints are considered simultaneously, including the requirement from buyers' or sellers' capacities. From the perspectives of the platform, sellers and buyers, three objectives are studied, namely, the platform's profit, the sum of the sellers' profit, and the sum of the buyers' surplus utilities. Additionally, two different matching schemes are discussed to satisfy various requirements from buyers that orders can be completed by one or more sellers.

To optimize the matching problems between buyers' demand and sellers' capacity, two meta-heuristics based on GA are developed: one is based on the traditional GA, and the other is based on NSGA-II. The contribution of this paper relies on the new 2-dimensional crossover and order-first mutation operators for the GA. In addition, an effective solution repair mechanism is developed. The experimental results show that both algorithms are effective for different schemes. Furthermore, it is found that the GA has better performance in the case of "more sellers", and the NSGA-II shows better performance in the "more buyers" case.

Acknowledgments

This research was supported by the Youth Foundation of Social Science and Humanity, China Ministry of Education (Grant No. 20YJC630166, 21YJC630084), Social Science Foundation of Shandong Province of China (Grant No. 20DGLJO10), the National Natural Science Foundation of China (Grant No. 72101136), the Youth Foundation of Shandong Natural Science Foundation of China (Grant No. ZR2021QG001, ZR2021QG045) and the Young Scholars Program of Shandong University.

Data Availability Statements

The datasets generated during and/or analyzed during the current study are available in the Mendeley Data repository, DOI: 10.17632/sx88drkkth.2.

References

Agatz, Niels, Alan Erera, Martin Savelsbergh, and Xing Wang. 2012. "Optimization for Dynamic Ride-Sharing: A Review." *European Journal of Operational Research* 223 (2). Elsevier B.V.: 295–303. doi:10.1016/j.ejor.2012.05.028.

Boysen, Nils, Dirk Briskorn, and Stefan Schwerdfeger. 2019. "Matching Supply and Demand in a Sharing Economy: Classification, Computational Complexity, and Application." *European Journal of Operational Research* 278 (2). Elsevier B.V.: 578–595. doi:10.1016/j.ejor.2019.04.032.

Cachon, Gérard P., Kaitlin M. Daniels, and Ruben Lobel. 2017. "The Role of Surge Pricing on a Service Platform with Self-Scheduling Capacity." *Manufacturing and Service Operations Management* 19 (3): 368–384. doi:10.1287/msom.2017.0618.

Cao, Yan, Xinggang Luo, C. K. Kwong, and Jiafu Tang. 2013. "Supplier Pre-Selection for Platform-Based Products: A Multi-Objective Approach." *International Journal of Production Research* 52 (1). Taylor & Francis: 1–19. doi:10.1080/00207543.2013.807376.

Cheng, Ying, Fei Tao, Dongming Zhao, and Lin Zhang. 2017. "Modeling of Manufacturing Service Supply–Demand Matching Hypernetwork in Service-Oriented Manufacturing Systems." *Robotics and Computer-Integrated Manufacturing* 45. Elsevier: 59–72. doi:10.1016/j.rcim.2016.05.007.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T Meyarivan. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2).

Furuhata, Masabumi, Maged Dessouky, Fernando Ordóñez, Marc Etienne Brunet, Xiaoqing Wang, and Sven Koenig. 2013. "Ridesharing: The State-of-the-Art and Future Directions." *Transportation Research Part B: Methodological* 57. Elsevier Ltd: 28–46. doi:10.1016/j.trb.2013.08.012.

Ghassemi Tari, Farhad, and Zahra Hashemi. 2016a. "A Priority Based Genetic Algorithm for Nonlinear Transportation Costs Problems." *Computers and Industrial Engineering* 96. Elsevier Ltd: 86–95. doi:10.1016/j.cie.2016.03.010.

Ghassemi Tari, Farhad, and Zahra Hashemi. 2016b. "A Priority Based Genetic Algorithm for Nonlinear Transportation Costs Problems." *Computers and Industrial Engineering* 96. Elsevier Ltd: 86–95. doi:10.1016/j.cie.2016.03.010.

Guo, Liang, and Xiaole Wu. 2018. "Capacity Sharing between Competitors." *Management Science* 64 (8): 3554–3573. doi:10.1287/mnsc.2017.2796.

He, Long, Zhenyu Hu, and Meilin Zhang. 2019. "Robust Repositioning for Vehicle Sharing." *Manufacturing & Service Operations Management* 22 (2). INFORMS: 241–256. doi:10.1287/MSOM.2018.0734.

Jorge, Diana, and Gonçalo Correia. 2013. "Carsharing Systems Demand Estimation and Defined Operations: A Literature Review." *European Journal of Transport and Infrastructure Research* 13 (3): 201–220. doi:10.18757/ejtir.2013.13.3.2999.

Lee, C. K.H. 2016. "A GA-Based Optimisation Model for Big Data Analytics Supporting Anticipatory Shipping in Retail 4.0." *International Journal of Production Research* 55 (2). Taylor & Francis: 593–605. doi:10.1080/00207543.2016.1221162.

Li, Hui Fang, Xun Dong, and Chang Gang Song. 2012. "Intelligent Searching and Matching Approach for Cloud Manufacturing Service." *Jisuanji Jicheng Zhizao Xitong/Computer Integrated*

Manufacturing Systems, CIMS 18 (7): 1485–1493.

Li, Hui, and Kannan Srinivasan. 2019. "Competitive Dynamics in the Sharing Economy: An Analysis in the Context of Airbnb and Hotels." *Marketing Science* 38 (3): 365–391. doi:10.1287/mksc.2018.1143.

Lin, Xiaogang, and Yong Wu Zhou. 2019. "Pricing Policy Selection for a Platform Providing Vertically Differentiated Services with Self-Scheduling Capacity." *Journal of the Operational Research Society* 70 (7). Taylor & Francis: 1203–1218. doi:10.1080/01605682.2018.1487822.

Ma, H. L., Felix T.S. Chan, and S. H. Chung. 2013. "Minimising Earliness and Tardiness by Integrating Production Scheduling with Shipping Information." *International Journal of Production Research* 51 (8). Taylor & Francis Group : 2253–2267. doi:10.1080/00207543.2012.718452.

Molla-Alizadeh-Zavardehi, S., M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam. 2011. "Solving a Capacitated Fixed-Charge Transportation Problem by Artificial Immune and Genetic Algorithms with a Prüfer Number Representation." *Expert Systems with Applications* 38 (8). Elsevier Ltd: 10462–10474. doi:10.1016/j.eswa.2011.02.093.

Othman, Zalinda, Mohammad Reza Rostamian Delavar, Sarah Behnam, and Sina Lessanibahri. 2011. "Adaptive Genetic Algorithm for Fixed-Charge Transportation Problem." *IMECS 2011 - International MultiConference of Engineers and Computer Scientists 2011* 1 (March): 96–101.

Patro, Gourab K., Abhijnan Chakraborty, Niloy Ganguly, and Krishna Gummadi. 2020. "Incremental Fairness in Two-Sided Market Platforms: On Smoothly Updating Recommendations." *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (01): 181–188. doi:10.1609/aaai.v34i01.5349.

Rezaei, Jafar, and Mansoor Davoodi. 2012. "A Joint Pricing, Lot-Sizing, and Supplier Selection Model." *International Journal of Production Research* 50 (16). Taylor & Francis Group : 4524–4542. doi:10.1080/00207543.2011.613866.

Rohaninejad, Mohammad, Reza Tavakkoli-Moghaddam, Behdin Vahedi-Nouri, Zdeněk Hanzálek, and Shadi Shirazian. 2021. "A Hybrid Learning-Based Meta-Heuristic Algorithm for Scheduling of an Additive Manufacturing System Consisting of Parallel SLM Machines." *International Journal of Production Research*. Taylor & Francis. doi:10.1080/00207543.2021.1987550.

Song, Mingjie, and Dongmei Chen. 2018. "An Improved Knowledge-Informed NSGA-II for Multi-Objective Land Allocation (MOLA)." *Geo-Spatial Information Science* 21 (4). Taylor and Francis Ltd.: 273–287. doi:10.1080/10095020.2018.1489576.

Wang, Tianri, Pengzhi Zhang, Juan Liu, and Liqing Gao. 2021. "Multi-User-Oriented Manufacturing Service Scheduling with an Improved NSGA-II Approach in the Cloud Manufacturing System." *International Journal of Production Research* 60 (8). Taylor & Francis: 2425–2442. doi:10.1080/00207543.2021.1893851.

Wang, Xing, Niels Agatz, and Alan Erera. 2018. "Stable Matching for Dynamic Ride-Sharing Systems." *Transportation Science* 52 (4): 850–867. doi:10.1287/trsc.2017.0768.

Wang, Yulan, Baozhuang Niu, and Pengfei Guo. 2013. "On the Advantage of Quantity Leadership When Outsourcing Production to a Competitive Contract Manufacturer." *Production and Operations Management* 22 (1): 104–119. doi:10.1111/j.1937-5956.2012.01336.x.

Wang, Yulan, Baozhuang Niu, and Pengfei Guo. 2014. "The Comparison of Two Vertical Outsourcing Structures under Push and Pull Contracts." *Production and Operations Management* 23 (4): 610–625. doi:10.1111/poms.12025.

Xie, Lei, and Hongshuai Han. 2020. "Capacity Sharing and Capacity Investment of Environment-

Friendly Manufacturing: Strategy Selection and Performance Analysis." *International Journal of Environmental Research and Public Health* 17 (16): 1–20. doi:10.3390/ijerph17165790.

Yağmur, Ece, and Saadettin Erhan Kesen. 2022. "Bi-Objective Coordinated Production and Transportation Scheduling Problem with Sustainability: Formulation and Solution Approaches." *International Journal of Production Research*. Taylor & Francis. doi:10.1080/00207543.2021.2017054.

Yuan, Minghai, Yadong Li, Lizhi Zhang, and Fengque Pei. 2021. "Research on Intelligent Workshop Resource Scheduling Method Based on Improved NSGA-II Algorithm." In *Robotics and Computer-Integrated Manufacturing*. Vol. 71. Elsevier Ltd. doi:10.1016/j.rcim.2021.102141.

Zhou, Yong Wu, Xiaogang Lin, Yuanguang Zhong, and Wei Xie. 2019. "Contract Selection for a Multi-Service Sharing Platform with Self-Scheduling Capacity." *Omega (United Kingdom)* 86. Elsevier Ltd: 198–217. doi:10.1016/j.omega.2018.07.011.