Metaheuristic Optimization on Tensor-Type Solution via Swarm Intelligence and Its Application in the Profit Optimization in Designing Selling Scheme

Metaheuristic Optimization on Tensor-type Solution via Swarm Intelligence and its Application in the Profit Optimization in Designing Selling Scheme *

Frederick Kin Hing Phoa[1][0000−0002−7417−8982], Hsin-Ping Liu[2], Yun-Heh (Jessica) Chen-Burger[3], and Shau-Ping Lin[2][0000−0003−3423−991X]

[1] Institute of Statistical Science, Academia Sinica, Taipei 115, Taiwan
fredphoa@stat.sinica.edu.tw
[2] National Taiwan University, Taipei 106, Taiwan
[3] Heriot-Watt University, Edinburgh EH14 4AS, UK

Abstract. Nature-inspired metaheuristic optimization has been widely used in many problems in industry and scientific investigations, but their applications in designing selling scheme are rare because the solution space in this kind of problems is usually high-dimensional, and their constraints are sometimes cross-dimensional. Recently, the Swarm Intelligence Based (SIB) method is proposed for problems in discrete domains, and it is widely applied in many mathematical and statistical problems that common metaheuristic methods seldom approach. In this work, we introduce an extension of the SIB method that handles solutions with many dimensions, or tensor solution in mathematics. We further speed up our method by implementing our algorithm with the use of CPU parallelization. We then apply this extended framework to real applications in designing selling scheme, showing that our proposed method helps to increase the profit of a selling scheme compared to those suggested by traditional methods.

Keywords: Swarm Intelligence · Tensor-type Particle · CPU Parallelization · Selling Scheme.

1 Introduction

Inspired by the behavior of biological systems and physical or chemical systems in nature [1], nature-inspired metaheuristic algorithms have been widely recognized as a powerful tool for optimization problems that are usually extremely complex and challenging in the real world. Swarm intelligence [2], which describes a collective intelligent behavior of self-organized and decentralized systems, is a

* This work is partially supported by the Academia Sinica grant number AS-TP-109-M07 and the Ministry of Science and Technology (Taiwan) grant numbers 107-2118-M-001-011-MY3 and 109-2321-B-001-013.

major class of metaheuristics. Some well-known algorithms, such as Genetic Algorithms (GA) [3], Artificial Bee Colony [4], Particle Swarm Optimization (PSO) [5], the Swarm Intelligence Based (SIB) method [8], and many others, belong to this algorithm family. Among all, PSO is widely used in engineering problems and some scientific investigations in the past decades. It has been well-designed for solving high-dimensional optimization problems in various fields. [7] pointed out four distinctive features for the popularity of the use of PSO among engineers. There are many versions and variants of PSO after its first introduction in [6], we denote PSO as the traditional framework of PSO mentioned in [6] for the rest of this paper unless specified.

Similar to many metaheuristics, PSO works well in a continuous domain as its velocity and position are well-defined with physical meanings, but it may not be the best candidate for problems with solutions in a non-continuous domain, which is not necessarily a discrete domain but some domains with "holes". Such optimization methods are common in mathematics and statistics, especially when solutions are in the matrix form and full of categorical variables like choices or symbols. Even though [7] and many others suggest to tackle thiese discrete scenario via a simple round-off, it is not trivial to verify if the resulting solution is truely optimal. This leads to an introduction of the the Swarm Intelligence Based (SIB) method proposed in [8] that works perfectly in a wide range of discrete optimization problems, such as the constructions of optimal experimental designs [9], the uniform distribution of testing points [10], target localization [11], supercomputing scheduling [12], hot spot determination [13], and many others. Details will be briefly reviewed in the next section.

Traditionally, a solution can be just a zero-dimensional number, an one-dimensional vector, or at most a two-dimensional matrix. A solution that has dimensions more than two is rare because the search becomes difficult in a huge solution domain, and the higher the dimensions, the larger the solution domain. As science and technology have advanced, practitioners attempt to perform complex optimization problems, and the computing power of searching high-dimensional solutions is available in the era of artificial intelligence. Not only do high-dimensional solutions test the feasibility of computation in both hardware and software, but additional cross-dimensional constraints, like some interactive quantities among several dimensions, create additional complexity in the optimization procedure. If cross-dimensional constraints do not exist, one may decompose the high-dimensional solution into layers of low-dimensional components and optimizes them one-by-one, but the existence of cross-dimensional quantities, which commonly exist in the real applications, break this layer independency assumption.

A real example of complex optimization with high-dimensional solution space exists in supply chain management, which was first termed in 1982 by Keith Oliver, a consultant at Booz Allen Hamilton. Supply chain management is the flow of goods and services management, and it includes all processes that transform raw materials into final products. It can also be considered as the connection of a business's supply-side from suppliers to customers. Good supply chain

management helps a company gain a competitive advantage in the marketplace. Readers who are interested in the basics of supply chain management are referred to [14]. It is of great interest to coordinate all parts of a supply chain from supplying raw materials to delivering products under the purpose of minimizing total costs and maximizing net profits in the process. Surprisingly, many optimization problems in supply chain management employ traditional methods llike linear programming and others. In the era of big data and artificial intelligence, there are many advanced optimization techniques that can greatly reduce the computational complexity and include many complicated problem constraints into consideration. This growing gap of optimization methods becomes an obstacle for researchers in supply chain management to develop large-scale data analysis techniques for better optimization schemes like the multi-channel selling scheme where products and services are sold and delivered to customers via different means.

In this work, we consider the optimization of the multi-channel selling scheme via a modern optimization technique in swarm intelligence. In Section 2, we briefly review optimization techniques in swarm intelligence. In Section 3, we implement the swarm intelligence based method for the optimization of the multi-channel selling scheme. Two real-life examples are used to demonstrate the efficiency of the proposed optimization method in Section 4. Some concluding remarks are stated in the last section.

2 A Brief Review of PSO and SIB

In this section, we review the basics of PSO and SIB, which are the two main algorithms used in this work. Readers who are interested in the details and theories of these algorithms are referred to [5] and [8].

2.1 Particle Swarm Optimization (PSO) Algorithm

Particle Swarm Optimization (PSO) is one of the most representative swarm intelligence algorithms in the past decades, and it has been widely applied in many industrial and scientific optimization problems. It is popular because of its easy implementation, and it is highly efficient in terms of memory and speed. In a PSO algorithm, we first initialize a swarm or a number of random particles, each of which represents a possible solution to the objective problem in the search space. The position of a particle is expressed as a vector consisting of values in every dimension. In addition to a position, each particle is given a velocity to determine its movement. At the end of every iteration, the position of every particle is updated based on its own velocity.

To make the swarm results in a good solution, an objective function has to be defined for evaluating the performance of a solution. With this definition, we are able to determine the Local Best (LB) particle and the Global Best (GB) particle. LB is the best solution a particle has encountered so far, and GB is the best one among all LB or the best solution that the whole swarm has encountered

so far. In each iteration, each particle's position is influenced by its LB and GB position through its velocity, which can be expressed as the following equations:

$$v_i^{t+1} \leftarrow \alpha v_i^t + b(x_i^{*,t} - x_i^t) + c(x_i^{+,t} - x_i^t)$$

$$x_i^{t+1} = x_i^t + v_i^t,$$

where i denotes the number of the iteration, v_i is the velocity of the particle, *alpha* is the inertia weight, b is the weight given to the cognitive/individual aspect, c is the weight given to the social aspect, x_i is the particle's position, x_i^* is the LB position of the particle, and x_i^+ is the GB position of the swarm. The determinations of b and c are generally user-defined or based on expert's experience.

2.2 Swarm Intelligence Based (SIB) Algorithm

Although PSO does not have any assumptions for the objective function, the search space is assumed to be continuous in the standard framework. There exist some variants of PSO to modify non-continuous domains, such as a simple round-off of velocities or using a probabilistic approach, but they may not be as good as in the Swarm Intelligence Based (SIB) method, which can be viewed as a hybrid method that some of SIB components can be viewed as a discretized version of PSO.

Similar to PSO, there is a swarm consists of several particles, LB particles, and a GB particle in the SIB algorithm. The objective function is also defined for evaluating the performance of particles. The main difference comes in the velocity update process. Rather than a linear combination formula of inertia and information from the two best particles in PSO, SIB extracts some important information from LB and GB particles by "mixing" particle units in the MIX operation. In addition, instead of only one choice of position updates in PSO, SIB picks the best of the three candidates to update in the MOVE operation. Below are the details of the two operations.

In the MIX operation, every particle has to be mixed with their own LB and GB, respectively, which returns two new positions called *mixwLB* and *mixwGB*. To mix a particle with a best particle, a given proportion, called q_{LB} and q_{GB} for mixing with LB and GB respectively, of entries in the particle is modified based on the corresponding values in the best particle. For example, we may simply replace the entry with the value in the best particle, or we may choose a random number in the range of two values to be the new value from the entry. Although there are no theoretical derivations to set the optimal values of q_{LB} and q_{GB}, our experience suggests setting $q_{LB} > q_{GB}$ to avoid premature convergence towards a relative good particle without an adequate exploration of the search space.

The MOVE operation is undertaken after all MIX operations are completed in an iteration. The performances of *mixwLB* and *mixwGB* are compared with the original position based on the objective function. If the best one among these three positions is one of the *mixwLB* and *mixwGB*, then it will be the new position of the particle. If none of the modified particles perform better than the

original particle, we randomly alter some units in the original particle to create a new particle, so the particle can escape from the trap of a local optimum and explore unknown search space near original positions. The algorithm is ceased at some pre-defined stopping criteria, such as reaching the assigned maximum number of iterations or achieving convergence towards a pre-defined threshold range of GB.

3 Method and Implementation

Multi-channel selling is a process of selling products on more than one sales channel. This sale strategy is popular in the E-commerce world nowadays, but the optimization of such a sale strategy can be very complicated with the existence of a middleman for product centralization between suppliers and customers. For example, a selling scheme of farm products may involve products gathering to a middleman company from multiple suppliers (farmers) and reselling to customers from the middleman company. On the other hands, a direct sale skips the middleman and connects the selling relationship between farmers and customers. It is obvious that if a direct sale is considered, the multi-channel selling scheme may not only increase the revenues of farmers and decrease the prices of products sold to customers, but also simplifies the complexity of the optimization and thus shortens the computational time.

Due to its high-dimensional and discrete natures of the selling scheme, we choose to use the SIB algorithm to tackle this important supply chain management problem. We consider a scenario that there are M suppliers supplying K types of products to N customers. The overall selling scheme is a three-dimensional matrix or a tensor with dimensions $N \times K \times M$. Before we propose the SIB algorithm for this problem, we state several underlying assumptions behind this scenario. First, we assume that all products are delivered directly from suppliers to customers, and there are no further complications on resale, buyback, or others. Second, we assume that the quantities of supply and demand are known in prior. This assumption is possibly valid nowadays as the selling information can be collected online and analyzed via big data analytics. Third, we assume that every product has a constant price for customers and a constant cost for buying from suppliers, and the transportation cost per mile for a specific product is constant. This assumption makes the optimization simpler, and it is not difficult to implement price and cost variations in the optimization.

We follow the standard framework in [8] and implement the SIB method to the optimization of the multi-channel selling scheme as in the following pseudocode:

3.1 Initialization

We define a particle in SIB as a three-dimensional tensor X with dimensions $N \times K \times M$ for N customers, K product types, and M suppliers. Each entry x_{nkm} in X stands for the number of the kth product suggested to be sold to the

Table 1. The SIB Algorithm.

1: Initialize a swarm of particles.
2: Evaluate the objective function values of each particle.
3: Determine the Local Best (LB) and the Global Best (GB) for each particles.
4: while STOPPING CRITERIA NOT FULFILLED
5: Do MIX operation.
6: Do MOVE operation.
7: Update all LB particles and the GB particle.
8: Check the conditions of convergence.

nth customer from the mth supplier. Each column with K entries represents a selling scheme between a supplier and a customer, and each slice of size $M \times K$ represents a selling scheme towards a specific customer of interest.

In a supply chain optimization problem, the objective function is generally the profit that a selling scheme is able to earn. The profit is the difference between sales and costs. To simplify our problems, we only consider the cost of packaging from suppliers and the cost of delivery to customers as the only two costs in the objective function, and the sale is simply product prices and quantities that a customer purchases. Mathematically speaking, we have

$$Profit = Sale - Cost, \tag{1}$$

where $Cost = \Sigma(Delivery + Package)$ and $Sale = \Sigma(Price \times Quantity)$.

There are constraints in supply and demand in this problem, and we implement these constraints in the particle generation step. In specific, we generate each column separately and combine these columns into a particle. To make sure the availability of products in both supply and demand, we record the remaining supply and demand after generating each slice of the particle. We randomly choose integers from 0 to the minimum between the remaining supply and demand to the available entry. In case of no remaining supply or demand, the entry will set to be zero. Moreover, we shuffle the generating order in both column level and slice level to increase variations among particles. Once the particle initialization is done, then their objective function values are evaluated, and the best particles are defined accordingly.

3.2 Iteration and Update

Every particle is mixed with its own LB and GB of the swarm respectively in an iteration step and results in two outcomes denoted as mixwLB and mixwGB. The MIX operation is done via mixing two particles in a column-by-column fashion and shuffling the order of columns in each slice. We deal with a pair of columns in each MIX operation, one from the original particle and another from either the LB or the GB particle. To ensure the availability of output positions, we calculate the remaining demand and supply and update them after every MIX operation of a pair of columns is completed. For each pair of columns,

we first identify entry indices that their values are larger in the best particle, and both demand and supply constraints are still fulfilled. In other words, we do not do anything if there are no remaining counts in the demand or supply constraints. Then, we randomly choose a specific proportion of those selected indices and replace values in the original particle with the values corresponding best particle, where the proportion is equal to q_{LB} or q_{GB} ; we set q_{LB} to be 0.6 and q_{GB} to be 0.4. Since we only select entries with larger values in the best particle, the objective function values will only increase in this process, and we determine to add this condition for achieving convergence faster.

After the MIX operation, we have three tensor particles: the original particle, the mixwLB particle, and the mixwGB particle. The MOVE operation in this algorithm is the same as the standard SIB algorithm. If either the mixwLB or the mixwGB particle has a better objective function value, then the original particle will be updated with a better choice. If the original particle still has the best objective function value, we first count the number of elements that corresponds to the non-zero demand in each column, then we randomly choose half of them and assign new integers that are randomly generated from the range between 0 and the minimum within the remaining demand and supply. This step ensures that the particle is out of the local-attractive trap while fulfills demand and supply constraints.

The procedure continues until the stopping criteria are reached. The criteria can be the maximum number of iterations, the achievement of a pre-defined profit value, or a convergence of a large proportion of tensor particles towards GB. Once the procedure is completed, the GB particle is the outcome, which is the optimal multi-channel selling scheme suggested by our proposed SIB algorithm.

3.3 Acceleration by CPU Parallelization

The computation among tensors is time-consuming, so we use the CPU parallelization technique to accelerate the whole process. Using the python package Multiprocessing, the data of the global best particle is stored in the shared memory while pairs of particles and their local best particles are stored in different CPUs. The MIX and MOVE operations for every particle are run in different CPUs, and the new positions are compared with the GB particle separately. If an output particle performs better than the GB particle, we modify the GB particle in the shared memory. To keep the process synchronous, Barriers are used to hold the complete sub-processes until all particles are completed. Moreover, a Lock is used to protect a shared resource from being modified by two or more concurrently running processes at the same time to make sure that the correct information is recorded.

Since our SIB algorithm contains a lot of for-loop and basic numerical functions, it is essential for the success of our program to implement efficiently. Numba [15], which translates Python functions to optimize machine code at runtime using the industry-stand LLVM compiler library, is a suitable choice for our SIB algorithm. Numba-compiled numerical algorithms in Python is claimed

to approach the speeds of C or FORTRAN. The source code of Numba can be found in Github at github.com/numba/numba.

Numba translates Python functions to optimize machine code at runtime using the industry-standard LLVM compiler library. Numba-compiled numerical algorithms in Python can approach the speeds of C or FORTRAN. Our program contains a lot of for-loop and basic numerical functions, which makes it suitable for applying Numba.

4 Applications

In this section, we apply our SIB algorithm in two examples. The first example is the real data from the layer industry in Taiwan with one dealer playing the only supplier in the supply chain. We can neither find any data with multiple suppliers (without dealer) nor find other data with a larger number of customers and products, so we randomly generate data as an extended example with additional suppliers based on this real data in order to demonstrate the capability of handling high-dimensional data of our SIB algorithm.

4.1 A Real Example in Layer Industry in Taiwan

In this example, we have 30 farmers, 82 customers, 78 products, and one dealer. We consider the dealer as the only supplier in this supply chain. Our data consists of the supply amount of each product, the cost of purchasing eggs from farmers, distances between customers and the dealer, the transport cost per mile for each product, and product prices that are different among customers due to quantities. To compare the performance of the SIB method, we also implement the PSO algorithm with feasible initial particles and a random back strategy to this data. The swarm size is set at 40 particles, and the stopping criterion is fulfilled after 50 iterations.

Figures 1 and 2 are the profit trends of the resulting selling scheme suggested by SIB and PSO algorithms, respectively. We observe that the SIB algorithm converges faster and achieves a better position than the PSO algorithm. The final profit of the selling scheme by the SIB algorithm is $ 2,001,256.66, and that of the PSO algorithm is only $ 1,609,522.59.

4.2 An Example of Multi-channel Selling

The first example is not "multi-channel" because there is only one dealer on the supply side of the selling scheme. In addition, the numbers of customers and products are quite small when we consider standard e-commerce. Therefore, we artificially generate extended data based on the real data in the first example, with 1000 customers, 100 kinds of products, and 100 different suppliers. We set the swarm size at 20, and the maximum number of iterations are set at 100 steps. Since each particle in this example is a tensor of size $1000 \times 100 \times 100$, and the computation is time-consuming, thus CPU parallelization is employed. As

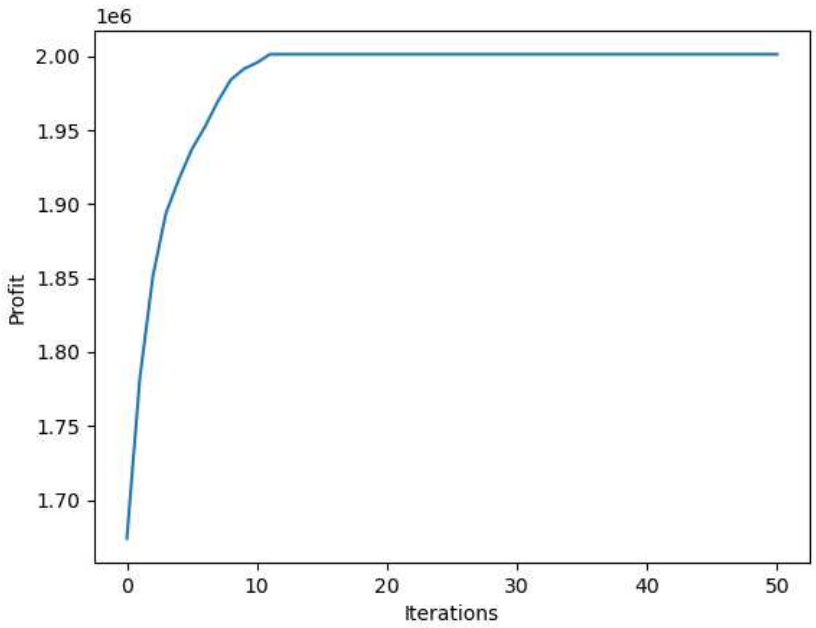

Fig. 1. The Profit Trend of GB via the SIB Algorithm.

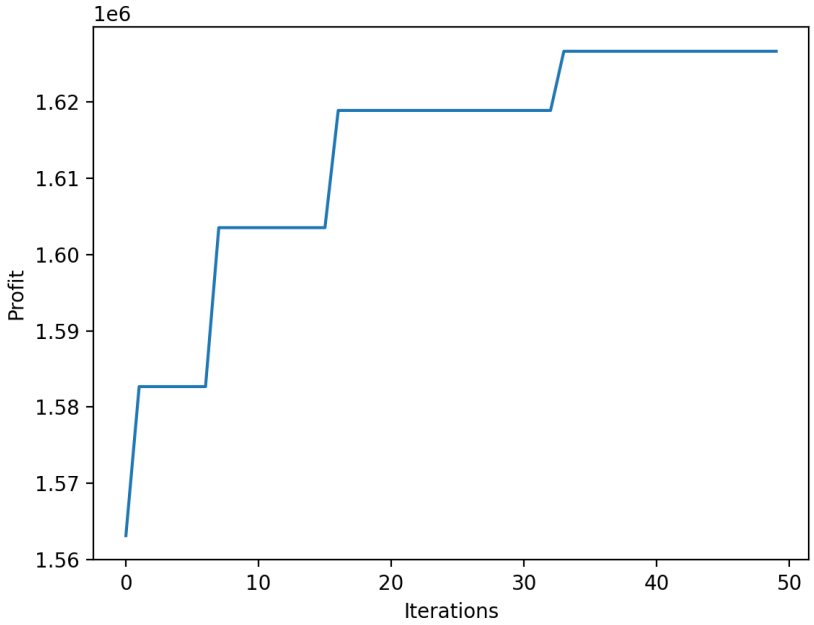

Fig. 2. The Profit Trend of GB via the PSO Algorithm.

a comparison, the whole procedure is completed in roughly an hour with CPU parallelization instead of 20 hours without CPU parallelization.

The simulation is performed about 100 times for each of two algorithms. Figure 3 shows the average profit comparison of the multi-channel selling scheme suggested by the initial GB and the final GB of the SIB algorithm. The selling scheme of the initial GB can be viewed as the best scheme among 20 random schemes, and the final GB is the optimal selling scheme suggested by the SIB algorithm. If a user conducts a multi-channel selling scheme without optimization or simply by experience, the performance of their schemes may not have a big difference from that of the initial GB. The significant increase in the profit from the initial GB to the final GB suggests that the SIB algorithm has greatly improved the multi-channel selling scheme. In specific, it modifies the scheme by creating a better selling configuration between the sellers and buyers, so that the selling scheme produces a very good profit as a result. Notice that the highest profit in the initial GB is still smaller than the lowest profit in the final GB in Figure 3, meaning that the worst-case scenario among all optimized selling scheme via SIB can still generate higher profit than the best-case scenario of a random scheme.

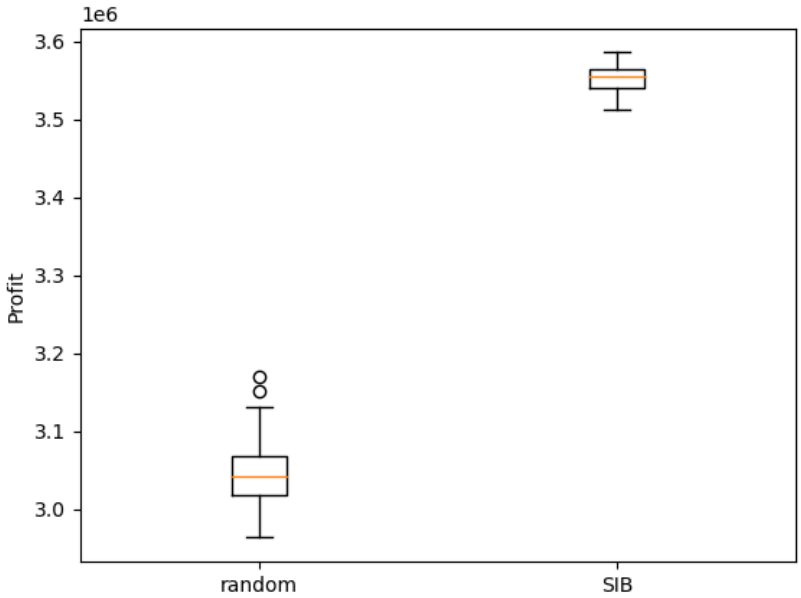

Fig. 3. Initial GB vs Final GB.

5 Conclusion

In the era of big data and artificial intelligence, a multi-channel selling scheme is an important advancement in supply chain management and e-commerce. However, computations on the optimization of the multi-channel selling scheme are infeasible if one employs traditional optimization techniques instead of parallelization. In this work, we propose the use of the SIB method to tackle this highly computational intensive problem. We introduce the high-dimensional tensor particle to be a solution particle in the SIB method with the consideration of demand and supply constraints and a new MIX operation to handle the information exchange between two particles with the preservation of the particle validity under these constraints. The simulation shows that the SIB method helps to increase the profits of these multi-channel selling schemes.

In reality, there are many other considerations in multi-channel selling that we simplified in our assumptions. Some of them are easy to implement as additional constraints, and they can be handled similarly to supply and demand constraints in our SIB method. If one considers the variations of price and cost due to the change of demand and supply, which is valid in the common sense of microeconomics, we may need advanced marketing models to perform predictions prior to the optimization. If one considers any buy-back or resale actions, we may need to consider the sale dynamics rather than a static model.

References

1. Mifjalili, S.: The art lion optimizer. Advances in Engineering Software **83**, 80–98 (2015)
2. Ab Wahab, M.N., Nefti-Meziani, S., Atyabi, A.: A comprehensive review of swarm optimization algorithms. PLoS ONE **10**(5), e0122827 (2015)
3. Goldberg, D.: Genetic Algorithms in Optimization, Search and Machine Learning. Addison Wesley, New York (2003)
4. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: arti
cial bee colony (ABC) algorithm. Journal of Global Optimization **39**(3), 459–471 (2007)
5. Kennedy, J.: Particle swarm optimization. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, vol. 10, pp.760–766. Springer (2010)
6. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, pp. 1942-1948 (1995).
7. Kim, T.H., Maruta, I., Sugie, T.: A simple and efficient constrained particle swarm optimization and its application to engineering design problems. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, pp. 389-400 (2010)
8. Phoa, F.K.H.: A Swarm Intelligence Based (SIB) method for optimization in designs of experiments. Natural Computing **16**(4), 597–605 (2017)
9. Phoa, F.K.H., Chen, R.B., Wang, W.C., Wong, W.K.: Optimizing two-level supersaturated designs using swarm intelligence techniques. Technometrics **58**(1), 43–49 (2016)

10. Phoa, F.K.H., Chang, L.L.N.: A multi-objective implementation in swarm intelligence and its applications in designs of computer experiments. In: 12th Intnerational Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD) 2016 on Proceedings, pp. 253–258. IEEE (2016)
11. Lin, F.P.C., Phoa, F.K.H.: An efficient construction of confidence regions via swarm intelligence and its application in target localization. IEEE Access **6**, 8610–8618 (2017)
12. Lin, F.P.C., Phoa, F.K.H.: Runtime estimation and scheduling on parallel processing super-computers via instance-based learning and swarm intelligence. International Journal of Ma-chine Learning and Computations **9**, 592–598 (2019)
13. Hsu, T.C., Phoa, F.K.H.: A smart initialization on the Swarm Intelligence Based method for efficient search of optimal minimum energy design. In: International Conference on Swarm Intelligence (ICSI) 2018 on Proceedings, pp. 78–87. Springer (2018)
14. Fredendall, L.D., Hill, E.: Basics of Supply Chain Management. 1nd edn. CRC Press, USA (2000)
15. Numba Homepage, https://numba.pydata.org/. Copyright at 2018 Anaconda.