Multi-objective evolutionary product bundling

A case study

Tunali, Okan; Bayrak, Ahmet Turul; Sanchez-Anguix, Victor; Aydogan, Reyhan

DOI
10.1145/3449726.3463219

Publication date
2021

Document Version
Final published version

Published in
GECCO 2021 Companion - Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion

Multi-Objective Evolutionary Product Bundling: A Case Study

Okan Tunalı
Ata Technology Platforms
İstanbul, Turkey
okant@atp.com.tr

Ahmet Tuğrul Bayrak
Ata Technology Platforms
İstanbul, Turkey
tugrulb@atp.com.tr

Víctor Sanchez-Anguix
Universitat Politècnica de València
València, Spain
vicsana1@upv.es

Reyhan Aydoğan
Özyeğin University, Istanbul, Turkey
Delft University of Technology, The Netherlands
reyhan.aydogan@ozyegin.edu.tr

ABSTRACT

Product bundling is a strategy conducted by marketing decision-makers to combine items or services for targeted sales in today's competitive business environment. Targeted sales can be in various forms, like increasing the likelihood of a purchase, promoting some products among a specific customer segment, or improving user experience. In this study, we propose an evolutionary product bundle generation strategy that is based on the NSGA-II algorithm. The proposed approach is designed as a multi-objective optimization procedure where the objectives are designed in terms of desired bundle feature distributions. The designed genetic algorithm is flexible and allows decision-makers to specify objectives such as price, season, item similarity and association with bundle size constraints. In the experiments, we show that the evolutionary approach enables us to generate Pareto solutions compared to the initial population.

CCS CONCEPTS

• **Theory of computation** → **Evolutionary algorithms**; • **Computing methodologies** → **Genetic algorithms**; • **Information systems** → Association rules; • **Applied computing** → Marketing;

KEYWORDS

Bundle generation, Evolutionary algorithms, Genetic algorithm, Multi-objective optimization, Decision support systems

ACM Reference Format:
Okan Tunalı, Ahmet Tuğrul Bayrak, Víctor Sanchez-Anguix, and Reyhan Aydoğan. 2021. Multi-Objective Evolutionary Product Bundling: A Case Study. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3449726.3463219

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463219

1 INTRODUCTION

In today's marketing environment, for the sectors with high competition but low-profit margins, such as fast-food, retail and e-commerce, it is crucial to provide customer-centric services. Providing that kind of services is a real challenge for marketing decision-makers as it covers customer experience, relatedness, quality and personalization. Targeted and smart marketing decisions increase the chance to appeal to customers, increase sales and maintain customer loyalty.

One strategy employed by marketers is product bundling [1]. This strategy consists of offering two or more products, frequently at a discounted price. The objectives of product bundling are twofold. Firstly, it increases product visibility by introducing customers to new or unknown products. Hence, possibly increasing sales. Secondly, product bundling makes it possible to better cater to the tastes and needs of customers [2]. For this purpose, product bundling is one of the sector-independent strategies [1]. This strategy consists of offering two or more products, aiming to increase the product visibility by introducing new or untried products to customers. It also enables better catering based on tastes and needs of customers [2].

However, product bundling is not a trivial task for marketers. It is time-consuming, as many aspects may be considered when creating a bundle such as; pricing [3], product association [4], product heterogeneity [5], personalization [6], basket size, and so forth. Given the large catalog offered by many retailers and the vast amount of products available in some sectors like fast-food, the tasks of creating optimal product bundles make it cognitively difficult, or even not feasible, for human decision-makers. Considering the total number of all possible combinations, the search space is extremely large, and assessing how well the product bundles are generated without realizing them (i.e., offering the customers directly and observing whether or not they like it) is not straightforward. Computer-assisted bundle generation and optimization techniques may prove useful for this task, as they can evaluate far more options than human decision-makers in less time while also considering several criteria. The results from these processes may help marketers in their choices.

There are numerous studies related to product bundling in the market. Bai et al. propose a bundle generation network that aims at creating personal bundles considering both quality and diversity. The network uses a typical encoder-decoder framework with a feature-aware softmax to enhance the insufficient representation

of common softmax [7]. Beladev et al. introduce a bundle generation method that is based on collaborative filtering (CF) techniques, similarly used for one of our criteria. The model maximizes the expected revenue of generated bundles and deals with the associations between the products in a bundle [2]. Likewise, Agarwal and Chatterjee use content-based similarity to create bundles [8]. Apart from that, Birtolo et al. provide a generative system to find the product bundles that best meet users' needs and, while, the needs of the vendor, such as maximizing net income and minimizing dead stocks [9]. Besides, Pathak et al. try to understand the meaning of what makes a good bundle. To achieve this, they use a dataset from the Steam video game distribution platform and generate new bundles and score them by Bayesian Personalized Ranking (BPR) [10].

As a different approach, some studies use recency, frequency, and monetary (RFM) values. Beheshtian-Ardakani et al. develop a model where the product bundles are determined accordingly for each market segment by clustering algorithms where the clustering is executed based on recency, frequency, and monetary (RFM) values. Besides, the apriori algorithm is applied to detect the association rules for product bundles [11]. Similarly, Hung et al. suggest a method that understands the customer's buying habits for the product, chooses an appropriate and matching custom bundling strategy for the customer, based on the RFM scores [12]. Yang and Lai use a different data set as online shopping data and use apriori to find product association [13].

In this article, we present an evolutionary bundle generation approach applied to the fast-food domain. Since evolutionary algorithms have proved to be successful at providing near-optimal solutions in problems where finding the optimal solution by exact methods is computationally expensive and not feasible, especially on a large search space [14–16].

The proposed approach aims to create new product bundles that are later evaluated and selected by marketers in their campaigns. Therefore, our ultimate objective is to create a decision support tool to help marketers discover novel product bundles satisfying a flexible list of criteria. Particularly, we propose a pipeline where the statistics are based on item basket structure, pricing, product similarity, product association, and sales trend similarity. The statistics are used as features, and a genetic algorithm is applied to generate Pareto optimal, new, and unique product bundles in the fast-food domain.

As required by domain integration, our solution includes intense data processing and feature extraction procedures both for the targeted fitness function learning module and the individual selection process of the genetic algorithm. Eventually, we offer an evolutionary decision support system that provides flexible objective definition and selection capability. We also compare generated bundles for different occasions using the NSGA-II algorithm.

The remainder of this article is organized as follows. First, we give details about the data used in Section 2. Then we describe our proposed product bundling approach, evaluation criteria, and the details of the genetic algorithm in Section 3. Afterward, Section 4 explains the experiments we conducted to evaluate our genetic algorithm approach for product bundling and analyze our findings. Finally, Section 5 concludes our work with future research directions.

2 DATA

In our study, we use partitions of a sales transactions database of a worldwide known fast-food chain, which also is our internal customer. The dataset consists of more than 12.6 million purchase transactions by 3.9 million unique customer ids where the transactions are composed of various combinations of 270 unique products.

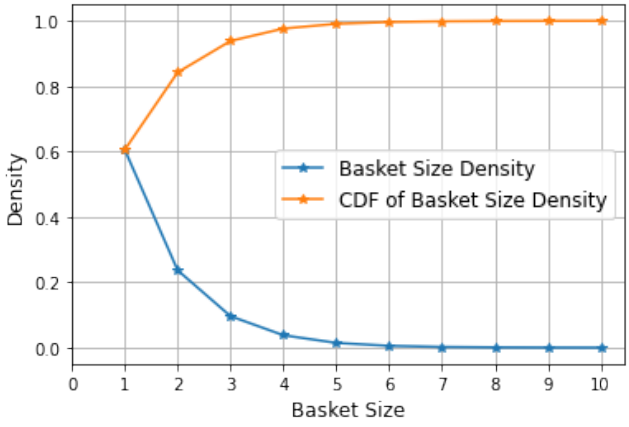

Figure 1: Basket size density and cumulative distribution function in the transactions of the dataset

As shown in Figure 1, in this domain, purchased items per transaction or basket sizes are in relatively small values. The cumulative distribution function (CDF) of basket sizes in the figure demonstrates that the customers prefer almost up to 6 items in their transactions. This distribution is one of the main motivations of this study since we see a great potential to increase sales by leading customers to prefer bigger and tailored baskets. Notice that basket size analysis is valuable to observe customer behavior and beneficial for strategic decision making. As will be explained in the following sections, basket sizes constitute decision boundaries searching for Pareto solutions. For instance, in our domain, the likelihood of a transaction having around ten items is relatively low. Since it is not rational to expect a significant change in customer behavior in the short run, a leading model to generate product bundles at this size would be impractical. Still, it can be used as a simulation tool by decision-makers, providing valuable insights.

3 PROPOSED BUNDLING APPROACH

In this section, we describe the feature extraction, fitness function learning procedure, and genetic algorithm integration that we designed for this problem. Initially, we explain the chromosome representation, population, and how to calculate fitness. Then we describe an adaptation of crossover and mutation for the product bundling approach.

3.1 Chromosome representation

In our problem, we have a set of available products denoted by $\mathcal{P}$=$\{p_1, p_2, .., p_m\}$ where m corresponds the number of products in our inventory. A bundle $B_i = \{p_1, .., p_n\}$ is a unbounded bag of

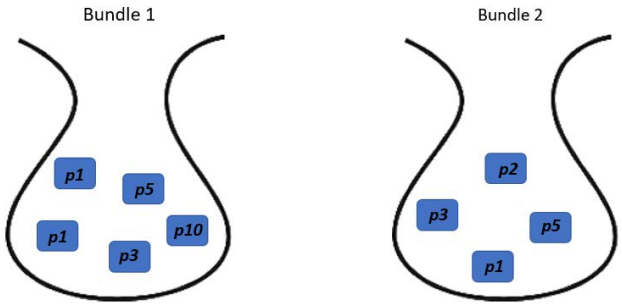

Figure 2: Two product bundles represented as bags. The first bundle contains two items of product 1, and one item of product 3, 5, and 10. The second bundle contains one item of products 1, 2, 3 and 5

products $p_i \in \mathcal{P}$, and $\mathcal{B}$ is the set of all possible bundles in the domain.

We define bundles as lists, the elements of which do not have to be unique. The decision to represent bundles as lists, but not sets, allows us to generate them with repeating items, which is required to meet domain requirements. For example, a fast-food menu or purchase transaction can include multiple and identical burgers. Following that, even the bundles are defined in unbounded size, and practically they are constrained by decision-makers and domain knowledge such as the diminishing demand versus basket size in Figure 1.

Figure 2 describes an example of how two product bundles are represented as bags. On the one hand, the first bag contains two items of product 1 and one item of product 3, 5, and 10. On the other hand, the second bundle includes one item of products 1, 2, 3, and 5. This representation matches the domain's needs (i.e., unbounded size, item repetition, irrelevant order) while offering a fast and efficient in-memory representation.

In our case, chromosomes correspond to product bundles where each item is a gene. To make it clear, we may list the representation properties as follows:

(1) Gene represents a single product
(2) Choromosome represents a product bundle
(3) Population represents a *set* of product bundles
(4) Chromosomes may be in various sizes and include genes with identical values

Following that, we explain the fitness of a chromosome followed by genetic modification operators.

3.2 Fitness definition

Considering multi-objective optimization procedure, the outcome of our evolutionary generation algorithm is expected to consist of Pareto solutions. Then, marketing decision-makers will be able to prioritize and select among solutions, i.e., item bundles, based on their business focus. To aim that goal, we propose four kinds of criteria to be utilized for guiding the generation of product bundles. Particularly, the price criterion, the association criterion, the content-based similarity criterion, the context-based similarity criterion. The first criteria evaluate the structure of a bundle according to its economical value, the second one is based on market basket analysis, and the last two criteria evaluate the content and context of a bundle attending to mechanisms inspired by well-known recommendation techniques [17].

In the following sections, we elaborately explain the fitness function definitions. Note that each function can be added to the optimization process multiple times. For example, the price criterion may focus on sales transactions occured in the summer and purchases of a target customer group. In that case, there would be two fitness functions based on price criterion that lead the search towards the target season's pattern and customers' purchasing pattern.

3.2.1 Basket Statistics. As mentioned earlier, item baskets are represented by related statistics to be used for fitness function learning. In other words, we calculate basket features, i.e., statistics, to be used in learning and evaluation of generated individuals during evolution.

- Price Statistics: Item price composition is one of the strongest characteristics indicators of an item basket. So, this feature is simply about the price distribution of items in a basket, which is represented as the price mean and standard deviation of the basket. The vector $[\mu^b_{pr}, \sigma^b_{pr}]$ denotes a single basket's price statistics.
- Associative Statistics: Item association is a non-personalized metric that establishes the basket relationship between two pairs of products according to co-occurrence in past transactions [18]. Therefore, item association is typically employed to identify items that are frequently bought together, and it aims to increase cross-selling. In our case, we use *lift* as the item association metric, which has a range from zero to infinity, and higher values mean stronger association. To utilize this metric, we calculate the pair-wise lift value mean and standard deviation of the basket. The vector $[\mu^b_{aso}, \sigma^b_{aso}]$ denotes a single basket's association statistics.
- Content Similarity Statistics: In our domain, we define content as the family or categorical membership of products, such as burgers or beverages. As a distinctive measure, we also calculate item similarities based on their name resemblance by the longest common string. Finally, we apply bagging to these similarity models and produce an element-wise cosine similarity matrix. It should be noted that family membership has a much higher weight for similarity, and the name is used to push variations of the same products closer. Similar to the association case, we calculate the pair-wise similarity of items and calculate the mean and standard deviation of similarities. The vector $[\mu^b_{fsim}, \sigma^b_{fsim}]$ denotes a single basket's content similarity statistics.
- Context Similarity Statistics: Purchase date or seasonality for products is another critical descriptive feature as we may have seasonal products (e.g., ice-creams, hot soups, etc.). As a contextual information, we aggregate weekly sales of the products such that for each product pr_i in the database we characterized a k dimensional vector $p_i =<$

$sw_{i,1}, sw_{i,2}, \ldots, sw_{i,k} >$, where $sw_{i,j}$ indicates the total number of sales of product i during the j-th week of the year. During processing, apply differencing used in time series analysis on a weekly basis which is used as feature vectors for similarity estimation. As in content similarity, we calculate the pair-wise context cosine similarity of items and calculate the mean and standard deviation of similarities. The vector $[\mu^b_{tsim}, \sigma^b_{tsim}]$ denotes a single basket's context similarity statistics.

3.2.2 Fitness Function Learning. In this section, we briefly explain the usage of basket statistics to learn data distribution or the patterns to be used as fitness functions that will play a role in evolution's selection step.

Reminding that our eventual target is to generate unique item baskets that show characteristic patterns provided by marketing decision-makers. For this purpose, we use a method to learn purchasing behaviors in terms of item basket features. As a result, given a set of item baskets, we first extract features or statistics in the previous section and use them for training multidimensional Gaussian mixture models [19, 20]. This approach is quite flexible, expandable, and sector-independent since the models learn from statistics but not directly from items. Finally, depending on the case scenario, we can pre-process and extract statistics from past transactions and train Gaussians. The reason behind using Gaussians is that because they can be used as complex clustering methods, learning multiple data centers with different covariance of distribution. Equations 1, 2, 3, 4 represents that given the Gaussian mixture models, we learn the fitness from target basket statistics to produce the likelihood of any item basket to come from that distribution.

$$f_{pr}(B) = \mathcal{L}(B|\mu_{pr}, \sigma_{pr}) \tag{1}$$

$$f_{aso}(B) = \mathcal{L}(B|\mu_{aso}, \sigma_{aso}) \tag{2}$$

$$f_{fsim}(B) = \mathcal{L}(B|\mu_{fsim}, \sigma_{fsim}) \tag{3}$$

$$f_{tsim}(B) = \mathcal{L}(B|\mu_{tsim}, \sigma_{tsim}) \tag{4}$$

3.3 Crossover

The crossover operator is designed to provide meaningful operations between bundles. In this case, the crossover operator mixes genetic material between parents by performing swap operations of the product. The crossover operator takes two parent bundles b_i and b_j and generates combinations of item swappings. Each swapping generates precisely two children, and a crossover is only eligible if target items are different. This filter eliminates ineffective operations.

Figure 3 shows an example crossover candidate generation with eligibility filter. Notice that the first items, burgers, of parents, are identical, so that swap operation is skipped. In the row of Children 1, we see the swap of a burger of the left parent with chips of the right parent. Here, we see the example of a duplicate item case. The rest of the candidate children are the eligible combinations of

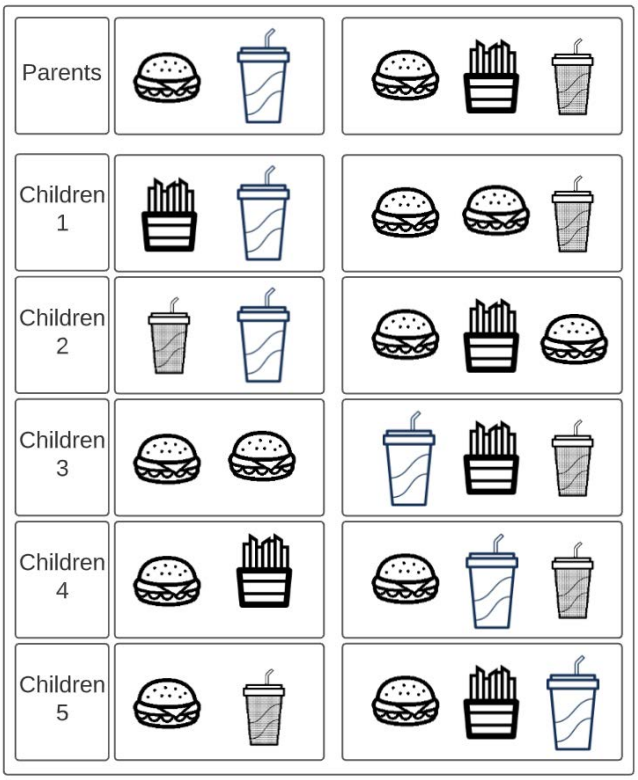

Figure 3: Sample crossover operation outcomes

item swaps. Given those candidate children, the crossover operator randomly picks a pair and passes them to the mutation operator. Note that the crossover operator does not test whether the child bundles already exist in the population. The reason for this is that there may be no swapping possibilities to generate a new child. So, this control is left to the mutation operator to ensure.

3.4 Mutation

In our setup, the mutation operator has a strong effect on the evolution process. As noted in the previous section, children generated by the crossover operator are passed for mutation. The mutation operator is ensured to change and generate a unique bundle. When a bundle is selected for mutation, the mutation operator randomly selects one of three possible actions: add a product to the bundle, remove an item from the bundle, and modify the bundle by swapping an item from the bundle with any different product in the database. In the following subsections, we explain how each of the mutation operators is carried out in detail. Then, we provide the pseudo-algorithm for the mutation operator.

3.4.1 Mutation type I: Add item to bundle. This mutation action aims to increase the size of the bundle by including a new item. When adding new items to the bundle, there are as many options as items in the product database. Here, a new item does not necessarily mean a non-existing item since we let item repetition. On the other hand, unlike crossover, the item added must generate a unique child to the population. This operation is not also eligible if the

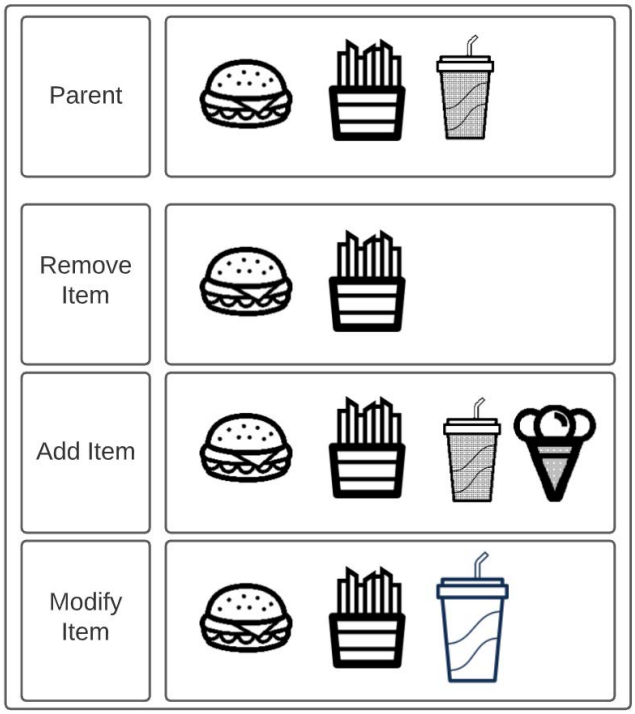

Figure 4: Sample mutation operations

child's item count exceeds the pre-defined maximum bundle size constraint.

3.4.2 Mutation type II: Remove item from bundle. Similarly, this mutation aims to decrease the size of the bundle. In this case, there are as many removal options as items in the bundle, as one can remove any item in the bundle. Similar to the previous case, item removal is not eligible if a child is not unique in population or its size is below the preset minimum basket size. This option lets decision-makers force the model to generate unique item baskets with a minimum size constraint.

3.4.3 Mutation type III: Modify item. The other type of mutation aims to introduce new products in the bundle to be mutated by swapping one of the products in the bundle with a different product taken from the item pool. Considering that each item has an equal chance of mutation in three different ways, it would be infeasible to produce each possibility, especially for large-item pools. Hence, given a bundle to be mutated, the mutation operation samples up to max_{mut} eligible mutations. Among those candidate children, the operator makes a random choice. Like other mutation types, the generated child must be a new member of the population.

3.5 Selection and Genetic Scheme

In this study, we integrate and adapt our fast-food domain business problem to a well-studied evolutionary algorithm NSGA-II. As the algorithm is known for its elitist and non-dominated sorting selection mechanism, we left its functionality unchanged. However, the objective integration module requires further explanation.

The algorithm below demonstrates fitness estimation for any list of bundles. In our setting, this method is used to prepare population individuals for the selection process. The algorithm starts with an iteration of a list of items in Line 1. In Line 2, the statistics generator processes a target item bundle and the outcome vector assigned to the bundle. Next, statistics of the target bundle are rescaled in Line 3 by the data scaler, which is used to z-normalize training statistics. This step is essential to get proper outcomes from trained models. Then, we iterate each fitness function to calculate the corresponding objective given by statistics. Line 9 can be evaluated as the core process of this study. Here, we pass features of baskets to the corresponding trained density estimators, Gaussian Mixtures, and receive likelihood signals in different magnitudes and directions. A strong positive signal implies a high likelihood that the target bundle features resemble ones in the train set of the given estimator. In our case, we assume higher likelihood means greater fitness. As a result, we aim to find Pareto bundles by defining dominance as having a greater likelihood for the given objective.

Data: B : List of item bundles (individuals), DS: Data scaler calculated from training data, F : Fitness functions to estimate bundle likelihood, BSG : Bundle statistics generator function, $b.stats$: vector of statistics of bundle b, $b.objectives$: Likelihood value for bundle b given its statistics to Gaussians
Result: B : Individuals having calculated objectives

```
1  foreach b ∈ B do
2      b.stats ← BSG(b);
3      b.scaled_stats ← DS(b.stats);
5      foreach F_i ∈ F do
7          f ← F_i;
9          b.objectives_i ← f(b.scaled_stats_i);
10     end
11 end
```

Algorithm 1: Bundle objective calculation procedure

In the Algorithm 2, child generation steps are demonstrated. As you might see, we follow a well-known flow. Initially, in Line 1 we initialize an empty children set to be filled with generated children. Line 3 and Line 6 shows parent selection by tournament with NSGA-II algorithm's default settings, ensuring parents are not the same individuals. In Line 8, we apply crossover to selected parents as elaborately explained in the Section 3.3. Next, we pass the crossover product children to mutation operator in Line 10. Note that, as explained in Section 3.4, mutation operation generates a unique child, and to do so, the operator gets the population and the existing children set that includes generated individuals up to that moment. Finally, in Line 14 we pass generated children and the fitness functions to calculate individual objectives as shown in Algorithm 1.

4 EXPERIMENTS

Before explaining the experimental setup and results, we exemplify the following case scenario and evaluate the results. We want to generate product bundles that are more likely to be sold in summer and purchased by customers who order during lunch breaks. Our

Data: $population$: A set of item bundles, $tournament$: NSGA-II's selection by tournament, F : Fitness functions to estimate bundle likelihood
Result: B : Individuals having calculated objectives

$children \leftarrow \emptyset$;
while $children.size < population.size$ **do**
  $parent1 \leftarrow tournament(population)$;
  $parent2 \leftarrow parent1$;
  while $parent1 = parent2$ **do**
    $parent2 \leftarrow tournament(population)$;
  end
  $child1, child2 \leftarrow crossover(parent1, parent2)$;
  foreach $child \in (child1, child2)$ **do**
    $child \leftarrow mutate(child, (population \cup children))$;
    $children \leftarrow (children \cup child)$;
  end
end
$calculate\_objectives(children, F)$;

Algorithm 2: Child generation scheme

Data: $N_{individuals}$: Number of individuals in population, $N_{transactions}$: Number of samples from queried transaction subset, $Quarter$: 3 monthly quarters of the year.

foreach $n_{indv} \in N_{individuals}$ **do**
  foreach $n_{sample} \in N_{transactions}$ **do**
    foreach $i_{instance} \in N_{instance}$ **do**
      foreach $i_q \in Quarters$ **do**
        $tr \leftarrow query\_db(target\_transactions, i_q)$;
        $tr_s \leftarrow sample(tr, n\_sample)$;
        $F \leftarrow train\_models(tr_s)$;
        $evolution\_model \leftarrow model(tr_s, F, n_{indv})$;
        $P_{init} \leftarrow evolution\_model.population$;
        foreach $r \in N_{repeat}$ **do**
          $evolution\_model.run()$;
          $log()$;
        end
      end
    end
  end
end

Algorithm 3: Experimental setup

first step is to query the database to get *transactions occurred in summer* and *transactions between 12:00 and 14:00*. As it might be noticed, we have two subsets of the dataset that might contain common transactions. Next, for each subset and its transactions, we calculate basket statistics as explained in the previous sections. Please note that the decision-maker may choose to leave any of the statistics out, such that the only statistics of interest for *lunch break* group may be the basket price distribution. Following that, for each of the selected statistics, we train a Gaussian mixture. In this case, there would be four Gaussians to represent summer transactions $f_{pr}(B^s)$, $f_{aso}(B^s)$, $f_{fsim}(B^s)$, $f_{tsim}(B^s)$ where B^s represents baskets in summer dataset. In addition, lunch break group would have $f_{pr}(B^{lb})$ where B^{lb} represents basket in lunch break dataset. Eventually, for this scenario, we train five independent fitness functions that output a likelihood value for any given itemset created during the evolution.

4.1 Experimental setup

Algorithm 3 shows our experimental setup. In this setup, $N_{individuals}$ is the list of (64, 128) individuals limit for the population. $N_{transactions}$ are the number of samples to get from queried transactions which, in our case, it is the list of (1 million, 2 million) samples. $N_{instance}$ is the list of instance numbers to repeat, and we picked in size of 10. Then we query the dataset based on the target quarter of the year. In Line 6 we get a sample from the database query result that have target transactions as explained in the previous section, and then, they are used to train Gaussian Mixtures in Line 7. Next, we prepare the evolution model by passing sampled transactions, fitness models, and the number of individuals constraint for the population in Line 8. Then, to track the progress, we save the initial population in Line 9. Finally, we set N_{repeat} to repeat this setup ten times and log the results for each run.

4.2 Evaluation

To estimate the success of multi-objective optimization, we use Lebesgue measure or S-metric differences compared to the initial population. According to this metric, each individual fitness vector is treated as a d-dimensional point to create a hyperspace. Here, S-metric produces a smaller value if the hyperspace is closer to the given reference point.

In order to keep the order of the objective values the same and scale them to a measurable range, we applied the sigmoid activation function known from machine learning applications. Sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$ maps the data between (0,1) and keeps the ordering same which makes it a good fit. Given that, we pickled our reference point as the upper boundary of the sigmoid, which is 1 for each dimension.

In Table 1 and Table 2 we see results for S-metric differences. The first table shows a smaller population and fewer generations, and the second table has two folds population size and a larger number of generations.

Both of the tables show us that our solution always improves population in a way that, eventually, we end up with unique item bundles close to the Pareto frontier compared to the initial population, as shown by the consistent negative mean value for S-metric differences.

Since we have four objectives in our experiments, hyperspace visualization requires a dimension reduction procedure. For this purpose, we used the elaborate study on visualization of Pareto front approximations [21], which formulates 4D to 3D dimension mapping properly. The mapping in 5 shows that the first two dimensions are calculated into a single value given the angle φ. In our case, we selected φ as 45° to stay aligned with that study.

		$S_{evolve} - S_{init}$		
N_{sample} (million)	*quarter*	count	μ	σ
1	1	100	-0.254	0.102
1	2	100	-0.169	0.072
1	3	100	-0.197	0.07
1	4	100	-0.212	0.069
2	1	100	-0.205	0.085
2	2	100	-0.179	0.064
2	3	100	-0.226	0.084
2	4	100	-0.207	0.08

Table 1: S-Metric difference statistics for the number of individuals is 64 and number of generations is 100

		$S_{evolve} - S_{init}$		
N_{sample} (million)	*quarter*	count	μ	σ
1	1	100	-0.294	0.066
1	2	100	-0.276	0.063
1	3	100	-0.289	0.06
1	4	100	-0.317	0.053
2	1	100	-0.327	0.101
2	2	100	-0.247	0.05
2	3	100	-0.286	0.077
2	4	100	-0.297	0.063

Table 2: S-Metric difference statistics for number of individuals is 128 and number of generations is 200

$$(f_1, f_2, f_3, f_4) \rightarrow (f_1 \cos\varphi + f_2 \sin\varphi, f_3, f_4) \tag{5}$$

Figure 5 and 6 show the 4D to 3D approximated hyperspaces. The experiment numbers stand for quarters, i.e., experiment 1 is a sample run that uses data from the first quarter of the year, having N_{sample} value of 2 million. Note that each sub-figure represents a single evolution, and as shown in the legend, initial population objective value distribution is represented by orange-colored points and evolved population in blue. Please notice that figures do not only show the points on the Pareto frontier but all individuals of the population. Considering the reference point, i.e. red star, in each sub-figure, evolved populations form closer hyperspaces and Pareto surfaces as shown by S-metric results in Table 1 and 2.

5 CONCLUSION

In today's marketing environment, competition is fierce. This is especially true in markets having a huge variety of products with a wide range of prices for every budget. To stand out of that competition, smart product bundling is an assertive candidate strategy. In this study, we propose an evolutionary product bundle generation approach for that situation. We define and adapt our business problem as a multi-objective optimization procedure; then integrate it into the NSGA-II algorithm. Eventually, we build up an evolutionary engine that generates on-demand product bundles. The engine is capable of operating with any transactional datasets with item

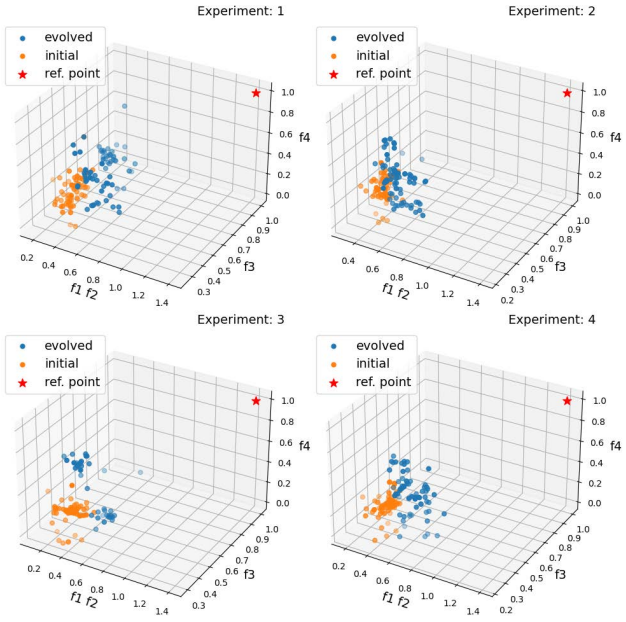

Figure 5: Population objective hyperspace approximations from experiments in Table 1

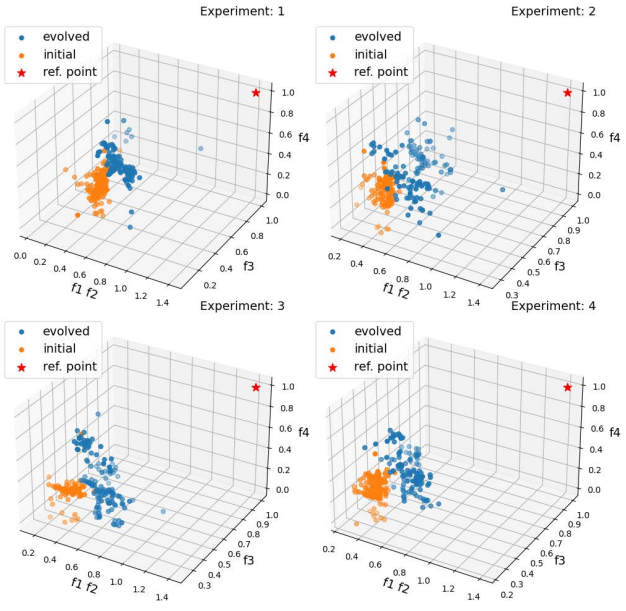

Figure 6: Population objective hyperspace approximations from experiments in Table 2

definitions and time information. Yet, we selected fast-food domain as a case study and showed that our engine can generate unique, targeted, and Pareto efficient item bundles despite the huge solution space. So, with our application, decision-makers can generate and also simulate the outcomes based on business requirements or for any kind of the desired scenario. Finally, we showed that

evolution ends up with individuals with significantly higher fitness values. In future work, we aim to extend the engine in a way that lets marketers generate bundles considering the trade-off between personalization, novelty, and utility by applying selective elitism to crossover and mutation operators.

REFERENCES

[1] Vithala R Rao, Gary J Russell, Hemant Bhargava, Alan Cooke, Tim Derdenger, Hwang Kim, Nanda Kumar, Irwin Levin, Yu Ma, Nitin Mehta, et al. Emerging trends in product bundling: Investigating consumer choice and firm behavior. *Customer Needs and Solutions*, 5(1):107–120, 2018.

[2] Moran Beladev, Lior Rokach, and Bracha Shapira. Recommender systems for product bundling. *Knowledge-Based Systems*, 111:193 – 206, 2016. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2016.08.013.

[3] Xiaogang Lin, Yong-Wu Zhou, Wei Xie, Yuanguang Zhong, and Bin Cao. Pricing and product-bundling strategies for e-commerce platforms with competition. *European Journal of Operational Research*, 283(3):1026–1039, 2020.

[4] Anthony Karageorgos and Elli Rapti. Dynamic generation of personalized product bundles in enterprise networks. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 208–217. Springer, 2013.

[5] Mehdi Sheikhzadeh and Ehsan Elahi. Product bundling: Impacts of product heterogeneity and risk considerations. *International Journal of Production Economics*, 144(1):209–222, 2013.

[6] Tzyy-Ching Yang and Hsiangchu Lai. Comparison of product bundling strategies on different online shopping behaviors. *Electronic Commerce Research and Applications*, 5(4):295–304, 2006.

[7] Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. Personalized bundle list recommendation. In *The World Wide Web Conference*, WWW '19, page 60–71, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313568.

[8] Agarwal Manoj K. and Chatterjee Subimal. Complexity, uniqueness, and similarity in between-bundle choice. *Journal of Product & Brand Management*, 12(6): 358–376, Jan 2003. doi: 10.1108/10610420310498795.

[9] C. Birtolo, D. De Chiara, S. Losito, P. Ritrovato, and M. Veniero. Searching optimal product bundles by means of ga-based engine and market basket analysis. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pages 448–453, 2013. doi: 10.1109/IFSA-NAFIPS.2013.6608442.

[10] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. Generating and personalizing bundle recommendations on steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1073–1076, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350228. doi: 10.1145/3077136.3080724.

[11] Arash Beheshtian-Ardakani, M. Fathian, and Mohammad Reza Gholamian. A novel model for product bundling and direct marketing in e-commerce based on market segmentation. *Decision Science Letters*, 7:39–54, 2018.

[12] Shao-Shin Hung, Li-Hua Li, Rong-Wang Hsu, and Pei-Jung Tsai. The personalized recommendation with bundling strategy based on product consuming period. CIS'09, page 461–469, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS). ISBN 9789604740710.

[13] Tzyy-Ching Yang and Hsiangchu Lai. Comparison of product bundling strategies on different online shopping behaviors. *Electronic Commerce Research and Applications*, 5(4):295–304, 2006. ISSN 1567-4223. doi: https://doi.org/10.1016/j.elerap.2006.04.006.

[14] Rafael de Paula Garcia, Beatriz Souza Leite Pires de Lima, Afonso Celso de Castro Lemonge, and Breno Pinheiro Jacob. A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. *Computers & Structures*, 187:77–87, 2017.

[15] Mitsuo Gen, Wenqiang Zhang, Lin Lin, and YoungSu Yun. Recent advances in hybrid evolutionary algorithms for multiobjective manufacturing scheduling. *Computers & Industrial Engineering*, 112:616–633, 2017.

[16] Victor Sanchez-Anguix, Rithin Chalumuri, Reyhan Aydoğan, and Vicente Julian. A near pareto optimal approach to student–supervisor allocation with two sided preferences and workload balance. *Applied Soft Computing*, 76:1–15, 2019.

[17] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[18] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Citeseer, 1994.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[20] Hagai Attias. A variational bayesian framework for graphical models. *Advances in neural information processing systems*, 12(1-2):209–215, 2000.

[21] Tea Tušar and Bogdan Filipič. Visualizing 4d approximation sets of multiobjective optimizers with prosections. GECCO '11, page 737–744, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305570. doi: 10.1145/2001576.2001677. URL https://doi.org/10.1145/2001576.2001677.