An R-base program to build e-commerce product discount portfolio using spiral dynamic optimization algorithm

Mohammad Rizka Fadhli[a,*]

[a]*Magister of Computational Sciences Program Faculty of Mathematics and Natural Sciences Institut Teknologi Bandung*

Abstract

The development of the digital economy market in Indonesia is growing every year. As a result, marketplaces have to compete with each other to get consumers. One of the marketplace startup in Indonesia has strategy to provide additional discounts on their listed products. Product discount portfolio problem is a Binary Linear Programming (BLP) problem with a decision variable in the form of a binary number that states whether a product is eligible for a discount or not. With many decision variables involved, we developed metaheuristic approaches Spiral Dynamic Optimization Algorithm (SDOA) to solve the optimization problem. The SDOA solution is proven to be more optimal (generates higher revenur) than the existing startup solution by average 9.8%.

Keywords: binary linear programming, spiral dynamic optimization algorithm, R programming, e-commerce

1. INTRODUCTION

Currently, 30 million Indonesians are used to buying and selling online, creating a market of 8 trillion rupiahs. This market can continue to grow to 40 trillion in the next five years. Online transactions are divided into social commerce (trading via social media platforms) and e-commerce (trading via marketplace platforms). By 2022, the revenue projection from the e-commerce market will exceed US$62 million. It was highly contributed by the increasing number of MSMEs that listed their products in e-commerce. The total number of MSMEs that market their products in e-commerce is currently 14.5 million MSMEs. The number has not reached half of the target set in 2023, which is 30 million MSMEs.

One of the top marketplace in Indonesia use a product discount strategy to win the competition. Each listed product on the marketplace has an initial price defined by the seller. Marketplace can intervene in its price indirectly by providing discounts, so consumers pay lower than the initial price. A study at an online retailer in China shows an influence between product discounts and consumer purchase behavior such as purchase incident, purchase quantity, and spending. Especially in specific discount percentage ranges [1]. The pricing and discount aim to attract consumers to buy products in the marketplace for a certain period. One of the mathematical models that can be used to determine prices is the Price Elasticity Model, a causality model between price and demand. A study comparing the price elasticity model with linear, polynomial, and exponential bases found that the linear model is the most commonly used, and it is easier to interpret [2].

The problem is determining which products need to be given an additional discount. This problem is one of the binary linear programming (BLP) forms: choosing the right product to maximize sales with a constraint limited discount budget in a certain period. Binary programming is a form of an optimization method in which the variables involved are binary numbers. The main characteristic of BLP is the matching process between indexes [3]. One of the studies related to BLP is the matching process to determine the placement of phasor measuring units in a power system [4]. Another study used BLP to optimize the

*Corresponding author
Email address: 20921004@mahasiswa.itb.ac.id (Mohammad Rizka Fadhli)

selection of 100 marketing channels and activities against millions of customers [5]. The goal is to optimize the marketing messages delivery so that sales increase. Business-related and consumer-centric optimization problems have not been widely published in journals.

At least we can use two methods to solve BLP: the exact method and the meta-heuristic method. Meta-heuristic methods are developing rapidly that several algorithms can quickly solve optimization problems. The meta-heuristic algorithm such as SDOA can solve BLP in a small number of decision variables involved. SDOA is an algorithm inspired by spiral motion in natural events. The rotation matrix is an algebraic principle closely related to SDOA [6]. SDOA is proven to be used to solve BLP problems by modifying the existing objective function and constraints [7]. The basic idea is to turn a constrained problem into an unconstrained problem by creating a penalty function.

In another study, researchers made an adaptive linear SDOA by changing the value of the contraction constant into a specific function that depends on the function value of each candidate solution [8]. The latest study in 2022 conducted a thorough review, including making several improvements to SDOA [9], such as:

- Changing the value of the contraction constant,
- Utilizing alternative functions (linear, quadratic, fuzzy, and exponential), and
- Crossing between SDOA and PSO into PSO with a spiral movement.

Currently, the marketplace determines which product to give a discount by trial and error from several possible combinations - (this method is similar to Monte Carlo simulation but on a simpler computational scale). In this research, we develop a computational model based on SDOA to solve the discount product portfolio optimization problem. Then we will compare the solution with the existing product portfolio.

2. METHOD

The marketplace has created the following portfolio:

Table 1: Sample Portfolio Data

product_code	budget	expected_revenue
6000094-0002	240.0	112.800
6000100-0003	70350.0	78289.500
6000301-0004	15300.0	7191.000
6000307-0005	2700.0	2079.000
6000348-0007	50460.0	55036.200
6000378-0010	1425.0	1097.250
6000514-0014	28774200.0	15355284.000
6000134-0026	3072.0	1443.840
6000149-0029	2052.0	964.440
6000245-0036	5670.0	1449.900
6000014-0038	65016.0	41897.520
6000027-0040	5625.0	4331.250
6000030-0041	21168.0	22548.960
6000042-0044	6747.3	1725.381
6000062-0048	6349.5	2424.015

Each portfolio contains 100 product candidates. The marketplace has the following problem: choosing the right product from the portfolio to generate the maximal revenue with a budget limit of 5 million rupiahs.

The problem can be written as follows:

$$\begin{array}{c} \text{find } x_i \in [0,1] \text{so that} \max r_i x_i, i \in \{1,2,..,100\} \\ \text{with constraint} \sum_{i=1}^{100} x_i b_i \leq 5000000 \end{array} \tag{1}$$

whereas r_i is expected revenue if the product i is given a discount and b_i is the amount of budget that needs to be spent to give a discount on a product i.

To solve using SDOA, we need to convert the constrained problem to an unconstrained problem. The general form is as follows: for any MILP or MIP:

$$\begin{array}{c} \min_{x \in \mathbb{R}} f(x) \\ \text{subject to } h(x) \leq 0 \\ x = (x_1, x_2, .., x_n)^T \in \mathbb{R} \end{array} \tag{2}$$

The general unconstrained form is as follows:

$$F(x, \beta) = f(x) + \beta \max(h(x), 0)^2 \tag{3}$$

where β is penalty constant (defined by large number). Finding x so that $F(x, \beta)$ is minimum.

This is general form of spiral dynamic optimization algorithm:

```
INPUT
  N number of candidate solutions
  theta rotation degree
    A(theta) is rotation matrix
  r contraction constant
  iter_max iteration limit
PROCESS:
  Step 1: Generate N random candidate solutions: xi in feasible region.
  Step 2: set k = 0.
  Step 3: Evaluate all F(xi). Find x0 which makes F(x0) the minimum. Set x0 as rotation center.
  Step 4: Update xi(k+1) = x0 + r (A(theta) (xi(k+1) - x0)).
  Step 5: Update k = k + 1.
  Step 6: Repeat step 3 - 5 until k reach iter_max.
OUTPUT:
  The last x0 is the optimal solution which makes F(X0) the minimum.
```

One of the crucial processes in SDOA is to rotate candidate solutions using a rotation matrix. For instance, this is R program to create roration matrix using input n dimension and θ as rotation angle.

```
rot_mat_creator = function(theta,n){
  # initiate zero matrix n x n
  temp_mat          = matrix(0,ncol = n,nrow = n)
  # put '1' into matrix diagonal
  diag(temp_mat)    = 1
  # initiate matrix rotation
  mat_rot           = temp_mat
  # iterate to calculate matrikx element
  for(i in 1:(n-1)){
    for(j in 1:i){
      temp          = temp_mat
      idx           = n-i
```

```
      idy           = n+1-j
      temp[idx,idx] = cos(theta)
      temp[idx,idy] = -sin(theta)
      temp[idy,idx] = sin(theta)
      temp[idy,idy] = cos(theta)
      mat_rot       = mat_rot %*% temp
    }
  }
  return(mat_rot)
}
```

In this case, one candidate solution is a vector of size 1×100 while the rotation matrix is of size 100×100. This process is quite computationally burdensome, especially since the number of candidate solutions used is relatively large. Therefore, we need to modify this rotation process by using matrix multiplication partitioning in linear algebra.

For instance: vector b is $1 \times$ n, and matrix A is n $\times$ n. Both can be divided into two, namely:

- Vector b1 is $1 \times \frac{n}{2}$ and vector b2 is $1 \times \frac{n}{2}$. Vector b1 contains the 1st to $\frac{n}{2}$-th elements of vector b. Vector b2 contains elements $\frac{n}{2} + 1$-th to n-th of vector b.
- Matrix A is divided into 4 for the same size matrix $\frac{n}{2} \times \frac{n}{2}$.

$$A = \begin{pmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ a_{31} & a_{32} & .. & a_{3n} \\ .. & .. & .. & \\ a_{n1} & a_{n2} & .. & a_{nn} \end{pmatrix} \tag{4}$$

- Matrix A11 is the top-left element of matrix A.
- Matrix A12 is the top-right element of matrix A.
- Matrix A21 is the lower-left element of matrix A.
- Matrix A22 is the lower-right element of matrix A.

$$r = Ab \tag{5}$$

So we get the result vector r with $1 \times$ n elements:

1. Elements 1 to $\frac{n}{2}$ = b1 * A11 + b2 * A12
2. Elements $\frac{n}{2} + 1$ to n = b1 * A21 + b2 * A22

The algorithm for solving optimization problems using modified SDOA was made in R programming language only using base library and dplyr [10] library for data preparation and data wrangling. As a form of evaluation of the algorithm, we compared the SDOA solution with:

1. The existing solution from the marketplace.
2. The exact solution uses the simplex method created using the R programming language with `library(ompr)` [11].

This comparison uses portfolio data from as many as ten portfolios with 100 products per portfolio each. Two factors that will be compared are the total budget issued and the total expected revenue.

3. RESULTS AND DISCUSSION

SDOA is created using the following parameters:

1. N candidate solutions are made up of 200 candidates in the form of a binary vector measuring 1×100.
2. θ is $\frac{50}{2\pi}$.
3. Contraction constant of 0.85.
4. The iteration limit is 70.

The following are some of the results of SDOA solutions in ten portfolios and their comparison to existing solutions and exact solutions.

3.1. Budget Comparison

First, we compare the budget from the exact method (`exact_budget` using simplex) with the budget using the SDOA and the existing budget from the marketplace.

Table 2: Budget Fullfilment: What percentage of the budget is spent on the solution?

#portfolio	exact_budget	existing_budget	spiral_budget
1	100.00	93.96	99.98
2	99.99	92.19	99.99
3	100.00	89.16	99.99
4	100.00	98.62	99.99
5	100.00	95.91	99.96
6	100.00	97.65	99.74
7	100.00	89.16	99.51
8	100.00	97.01	99.80
9	96.85	72.53	95.97
10	100.00	80.77	100.00

The total discounted budget per portfolio is five million rupiahs. If we look at the table above, each method produces a solution that does not violate the total budget limit. However, the exact solution always consumes all of the total budget, while the SDOA solution still has a smaller budget remaining than the existing solution. These findings found that the SDOA solution was better in using the total budget than the existing solution.

3.2. Total Expected Revenue Comparison

Table 3: Total Expected Revenue: What is the percentage of revenue achieved compared to the revenue of the exact solution?

#portfolio	eksak_revenue	existing_revenue	spiral_revenue
1	4.778 mil IDR	81.67%	90.19%
2	7.194 mil IDR	74.56%	80.52%
3	7.428 mil IDR	34.93%	50.24%
4	5.568 mil IDR	88.26%	92.53%
5	2.606 mil IDR	81.92%	90.65%
6	6.203 mil IDR	94.74%	94.57%
7	7.4 mil IDR	74.04%	88.4%
8	4.146 mil IDR	91.13%	99.41%

#portfolio	eksak_revenue	existing_revenue	spiral_revenue
9	5.097 mil IDR	82.27%	99.34%
10	5.525 mil IDR	80.41%	97.02%

The following comparison shows how much total expected revenue can be achieved. If the revenue from the exact solution is used as a benchmark value (the highest revenue that can be achieved from each portfolio), the percentage in the table is calculated by dividing the revenue from the existing solution or SDOA by the revenue from the exact solution. The revenue generated by the SDOA solution is 9.8% higher (on average) than the existing solution. However, the SDOA solution revenue is still 11.7% lower (on average) than the exact solution.

3.3. Similarity Between Exact and SDOA Solution

The exact solution is the most optimal, while SDOA has not been able to achieve that optimality. What do the two have in common? Suppose we count how many products are discounted or not discounted in both portfolios (exact and SDOA); here are the results:

Table 4: Similarity Product Portfolio

#portfolio	similar_product	dissimilar_product
1	63	37
2	36	64
3	55	45
4	59	41
5	74	26
6	48	52
7	63	37
8	85	15
9	91	9
10	68	32

On average, the similarity between the two portfolios is 64.2%. Now we see what proportion of products are given and not discounted on the exact method and SDOA.

Table 5: Product Proportion: How many products are given and not discounted?

#portfolio	portofolio_eksak	portofolio_spiral
1	Discount: 69; No discount: 31	Discount: 62; No Discount: 38
2	Discount: 14; No discount: 86	Discount: 72; No Discount: 28
3	Discount: 33; No discount: 67	Discount: 56; No Discount: 44
4	Discount: 54; No discount: 46	Discount: 69; No Discount: 31
5	Discount: 87; No discount: 13	Discount: 73; No Discount: 27
6	Discount: 42; No discount: 58	Discount: 72; No Discount: 28
7	Discount: 62; No discount: 38	Discount: 83; No Discount: 17
8	Discount: 86; No discount: 14	Discount: 89; No Discount: 11
9	Discount: 99; No discount: 1	Discount: 90; No Discount: 10
10	Discount: 82; No discount: 18	Discount: 74; No Discount: 26

Although there is a 64.2% similarity, the differences in the range of selected products in the two portfolios are apparent. The SDOA solution can still not explore the possibility of the optimal solution like an exact solution in the feasible area.

4. CONCLUSION

SDOA is proven to provide better solutions than existing solutions, namely: spending better budgets and generating more revenue. However, to achieve an optimal solution, such as an exact solution, further modification is needed to make SDOA explore more potential candidates in a feasible area. Some alternatives to consider are:

1. Using a pseudo-random algorithm in generating candidate solutions at an early stage.
2. Multiply N candidate solutions made.

5. DECLARATIONS

5.1. Study Limitations

In this study, the portfolio used is derived from the calculation of the price elasticity study previously conducted by the marketplace data analytic team.

- The basic assumption of this portfolio is that changes in sales only come from changes in product prices. The interaction between products in the same category (competitor effect) is not considered in this study.
- The budget and expected revenue per product are assumed to be fixed values (not dynamic or probabilistic).
- The total revenue calculation achieved from the solution of each method in the previous section is derived from the value in the portfolio.

5.2. Acknowledgements

The author would like to thank the marketplace data analytic team for entrusting its product portfolio data to be used as a case study in developing an SDOA-based optimization algorithm.

5.3. Declaration of Competing Interest

The author stated that there was no conflict of interest during this research.

REFERENCES

1. Liu, H., Lobschat, L., Verhoef, P., & Zhao, H. (2020). The effect of permanent product discounts and order coupons on purchase incidence, purchase quantity, and spending. *Journal of Retailing*. https://doi.org/10.1016/j.jretai.2020.11.007
2. Hajdinjak, M. (2020). Functions with linear price elasticity for forecasting demand and supply. *The B.E. Journal of Theoretical Economics*. https://doi.org/10.1515/bejte-2017-0015
3. Hillier, F. S., & Lieberman, G. J. (2001). *Introduction to operations research* (7th ed.). New York, US: McGraw Hill.
4. Billakanti, S., & Venkaiah, Ch. (2014). An effective binary integer linear programmed approach for optimal placement of PUMs in power systems. *IEEE: International Conference on Smart ELectric Grid*. https://doi.org/10.1109/ISEG.2014.7005593
5. Bigler, T., Baumann, P., & Kammermann, M. (2019). Optimizing customer assignments to direct marketing activities: A binary linear programming formulation. *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. https://doi.org/10.1109/IEEM44572.2019.8978863
6. Tamura, K., & Yasuda, K. (2011). Spiral dynamics inspired optimization. *Journal of Advanced Computational Intelligence and Intelligent Informatics*. https://doi.org/10.20965/jaciii.2011.p1116

7. Kania, A., & Sidarto, K. A. (2016). Solving mixed integer nonlinear programming problems using spiral dynamics optimization algorithm. *AIP Publishing*. https://doi.org/10.1063/1.4942987
8. Nasir, A. N. K., Ismail, R. M. T. R., & Tokhi, M. O. (2016). Adaptive spiral dynamics metaheuristic algorithm for global optimisation with application to modelling of a flexible system. *Applied Mathematical Modelling*. https://doi.org/10.1016/j.apm.2016.01.002
9. Omar, M. B., Bingi, K., Prusty, B. R., & Ibrahim, R. (2022). Recent advances and applications of spiral dynamics optimization algorithm: A review. *MDPI, Basel, Switzerland*. https://doi.org/10.3390/fractalfract6010027
10. Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). *Dplyr: A grammar of data manipulation*.
11. Schumacher, D. (2022). *Ompr: Model and solve mixed integer linear programs*. Retrieved from https://github.com/dirkschumacher/ompr