STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, January 2026, pp 407–422.
Published online in International Academic Press (www.IAPress.org)

An Energy Valley Optimizer Approach for Solving the Modified Quadratic Bounded Knapsack Problem with Multiple Constraints

Azka Huri 'Iin[1], Agustina Pradjaningsih[1,*], M. Ziaul Arif[1], Apriani Soepardi[2], Diva Rafifah Mutiara Muhammad [3]

[1]*Department of Mathematics, FMIPA, Universitas Jember, Indonesia*
[2]*Department of Industrial Engineering, Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia*
[3]*Embassy of Indonesia in Cairo, Egypt*

Abstract The Modified Quadratic Bounded Knapsack Problem with Multiple Constraints (MQBKMC) is an advanced variant of the traditional knapsack problem that incorporates interaction profits among items, quantity bounds, and multiple capacity constraints simultaneously. This study investigates the performance and parameter sensitivity of the Energy Valley Optimizer (EVO) algorithm in solving the MQBKMC using a dataset of 20 items with predefined interaction profits. Two critical parameters, namely the maximum number of function evaluations (MaxFes) and the number of particles (nParticle), were analyzed to assess their effects on solution quality, computational time, and stability. Experimental results show that increasing MaxFes consistently enhances solution quality, with the best objective value reaching Rp 6,286,994, whereas increasing nParticle beyond a moderate level provides only marginal improvement and increases computational time, indicating a trade-off between accuracy and efficiency. All experiments were conducted using MATLAB R2023a, and a comparison with the Particle Swarm Optimization (PSO) algorithm confirmed EVO's better convergence stability and profit performance. Overall, the EVO algorithm demonstrates robustness and efficiency in addressing complex, high-dimensional, and multi-constrained combinatorial optimization problems, offering practical insights for optimal parameter tuning in future implementations.

Keywords Knapsack Problem, Energy Valley Optimizer, Metaheuristic Algorithm, Optimization, Quadratic Bounded Knapsack Problem, Multiple Constraints, MQBKMC

AMS 2010 subject classifications 90B50, 90C20, 90C31

DOI: 10.19139/soic-2310-5070-2838

1. Introduction

The knapsack problem is recognized as one of the most fundamental and widely studied combinatorial optimization models, with far-reaching applications across computer science, operations research, industrial engineering, and logistics management [1]. At its essence, the knapsack problem involves selecting a subset of items from a candidate set to maximize total profit, subject to constraints on available resources or capacity. Depending on the nature of the decision variables, classical formulations include the 0-1 knapsack, bounded knapsack, and unbounded knapsack [2]. Over time, a broad spectrum of variants has emerged to address practical complexities, such as multi-objective optimization, multidimensional constraints, and item interdependencies, giving rise to the single-objective knapsack, multi-objective knapsack, multidimensional knapsack, multiple-constraints knapsack, and quadratic knapsack problems [3]. Recent research has explored various combinations of these variants, such as the multiple constraints 0-1 knapsack [4, 5, 6, 7, 8, 9, 10, 11], multidimensional 0-1 knapsack [12, 13, 14],

*Correspondence to: Agustina Pradjaningsih (Email: agustina.fmipa@unej.ac.id). Department of Mathematics, FMIPA, Universitas Jember. 37 Kalimantan Road, Jember, East Java Province, Indonesia (68121).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

multidimensional 0-1 knapsack with multiple constraints [15, 16], multidimensional bounded knapsack [17], multiple constraints bounded knapsack [18, 19], quadratic knapsack [20, 21, 22, 23, 24, 25, 26] and solving the knapsack problem using several variations of metaheuristic methods [27, 28, 29, 30, 31, 32]. Each of these variants introduces new dimensions of complexity to the solution space, posing significant challenges for the efficiency and scalability of existing optimization techniques and thus continuing to drive the development of more advanced solution methodologies in the field.

A particularly significant manifestation of the knapsack problem is found in inventory restocking and supply chain optimization, where decision-makers must determine not only which items to replenish but also in what quantities, all while seeking to maximize operational and economic gains. In practice, these optimization decisions are complicated by a host of realistic constraints, such as finite storage capacity, budget limitations, and fluctuating supply from vendors. The problem is further compounded by interactions among items: for example, the combined selection of certain products may enhance overall value due to bundling effects or complementarities, whereas other combinations may incur penalties or diminishing returns. These interdependencies naturally give rise to quadratic terms in the objective function, reflecting the complex synergies or trade-offs present in actual inventory systems. Moreover, managerial policies and market-driven considerations frequently impose upper and lower bounds on item quantities, adding yet another layer of complexity. Together, these factors culminate in advanced knapsack models characterized by nonlinear objectives, bounded item selection, and multiple, simultaneously active constraints. Such formulations capture the inherent multidimensionality and dynamism of modern distribution networks and highlight the persistent need for robust, adaptable optimization strategies capable of addressing real-world logistical challenges.

Addressing the complexity inherent in advanced knapsack models, such as those characterized by nonlinear objectives and multiple constraints, poses considerable challenges for exact optimization approaches like branch and bound or dynamic programming. The NP-hard nature of these problems often renders exact algorithms computationally infeasible for large-scale instances, as solution times can increase exponentially with the number of decision variables [3]. In response, metaheuristic algorithms have emerged as promising alternatives, offering a balance between solution quality and computational efficiency by intelligently exploring vast and complex search spaces. Inspired by natural phenomena—including biological evolution, collective animal behavior, and physical processes—metaheuristics have demonstrated notable flexibility in handling diverse objective functions and constraint structures without reliance on differentiability or linearity [33, 34, 35]. Among these, the Energy Valley Optimizer (EVO) represents a recent advancement, drawing on principles from particle decay and the tendency of systems to seek stable energy states [36]. In the EVO framework, candidate solutions are conceptualized as particles that traverse an energy landscape, iteratively moving toward optimal or near-optimal regions. Empirical studies have highlighted EVO's strong performance across various benchmark functions, frequently outperforming established algorithms such as Ant Colony Optimization, Firefly Algorithm, Harmony Search, and Cuckoo Search. Its efficacy has also been demonstrated in engineering applications, including speed reducer design, hydrostatic thrust bearing optimization, and ten-bar truss structures, where EVO has yielded competitive or optimal solutions [37, 38, 39]. Additionally, EVO has been successfully applied to complex scheduling problems [40] and several other engineering problems [41, 42, 43, 44]. Despite these promising results, its potential for addressing knapsack-type optimization challenges remains largely unexplored in the literature.

In light of these developments, this research addresses a notable gap by exploring the suitability of the Energy Valley Optimizer for solving sophisticated knapsack problems that involve nonlinear objectives, upper and lower bounds on item selection, and multiple simultaneous constraints. These problem characteristics are emblematic of many real-world scenarios, particularly in logistics and inventory management, where effective decision-making must accommodate complex interactions among items and stringent operational limits. Beyond implementing EVO for this class of problems, the study conducts a systematic evaluation of how variations in the algorithm's key parameters influence solution quality and computational efficiency. By conducting extensive numerical experiments, this work aims to establish evidence-based recommendations for parameter tuning and to demonstrate the practical value of EVO in addressing large-scale, multi-constrained combinatorial optimization tasks. The results are anticipated to advance both the theoretical understanding and practical application of

metaheuristic methods, reinforcing EVO's promise as a powerful tool for challenging optimization settings that defy conventional exact approaches.

2. Experimental Section

2.1. Data and Variables

This research adopts an experimental approach, drawing primarily on the dataset established by [19]. Notably, the original dataset did not account for the interaction profits between items, a factor critical to the formulation of quadratic objective functions. To incorporate this dimension, interaction profit values were systematically generated at random within the interval [25, 50], ensuring the model captures the potential synergies or penalties arising from item combinations. The experimental analysis was restricted to 20 dataset instances to ensure computational feasibility and consistency across scenarios. A comprehensive summary of all variables utilized in this study is presented in Table 1.

Table 1. Definitions and measurement descriptions of variables used in the inventory 50 optimization model.

Variable	Definition	Measurement Description
Weight (w_j)	Weight of each item.	Measured in kilograms (kg).
Volume (v_j)	Space occupied by each item.	Calculated as length $\times$ width $\times$ height (all items are box-shaped) in cm^2.
Purchase Price (b_j)	Initial cost to acquire each item.	Based on supplier's listed price.
Selling Price (s_j)	Price at which each item is sold to customers.	Set by the business for each item.
Profit (p_j)	Net gain from selling each item.	Selling price minus purchase price.
Lower Bound (l_j)	Minimum quantity required for each item.	Determined by business demand.
Upper Bound (u_j)	Maximum quantity allowed for each item.	Based on item availability from the supplier.

The detailed dataset used in this study, including the parameters for all 20 items, is presented in Table 2. In addition, the numerical values of the total weight, volume, and budget constraints are 8.100 kg^2, 14.200 cm^2, and Rp. 8,870,000, respectively.

The interaction profit data utilized in this study are summarized in Table 3.

2.2. Methodology

The following procedure was implemented to solve the Modified Quadratic Bounded Knapsack Problem with Multiple Constraints (MQBKMC) using the Energy Valley Optimizer (EVO) [36]:

1. Parameter Initialization
   At the beginning of the optimization process, the algorithm requires the definition of several key parameters. These include n (number of particles), d (number of decision variables), $x^j_{(i,\min)}$ and $x^j_{(i,\max)}$ (lower and upper bounds for the j-th variable), and $\text{FES}_{\max}$ (maximum number of function evaluations allowed during the optimization process).
2. Generation of Initial Solutions
   An initial population of n particles is generated randomly to serve as candidate solutions. Each particle's position is represented as a vector in a matrix of size $n \times d$. The initial value for each particle is calculated using Equation 1:

$$x^j_i = x^j_{(i,\min)} + \text{rand}\left(x^j_{(i,\max)} - x^j_{(i,\min)}\right); \quad i = 1, 2, 3, \ldots, n; \quad j = 1, 2, 3, \ldots, d \tag{1}$$

Table 2. Parameter Definitions and Data Values for 20 MQBKMC Items

No	Barang	lb	ub	w	v	b	s
1	item 1	6	13	5	6720	344000	440000
2	item 2	6	15	17	7560	95000	132000
3	item 3	6	13	8	20150	70000	87000
4	item 4	3	15	9	5700	65000	69000
5	item 5	4	13	8	1188	165500	223000
6	item 6	6	17	15	155552	65000	72000
7	item 7	4	15	8	3525	60000	65000
8	item 8	5	14	7	480	50000	80000
9	item 9	8	17	9	157	22000	83000
10	item 10	11	15	15	155552	65000	72000
11	item 11	6	17	15	156	105000	88000
12	item 12	9	17	8	2900	71000	170000
13	item 13	5	12	10	11220	141000	120000
14	item 14	10	12	13	1370	10000	133000
15	item 15	11	6	12	4104	244000	22000
16	item 16	13	17	17	2736	20000	92000
17	item 17	7	9	9	1794	145000	220000
18	item 18	17	3	19	12852	110000	60000
19	item 19	3	7	8	4896	35000	46000
20	item 20	4	13	9	6090	95000	107000

Table 3. Interaction profits among items.

Item	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	30	44	47	34	39	42	36	43	49	43	50	43	37	45	50	43	38	33	50
2	0	0	42	37	36	27	31	35	40	40	32	38	48	30	42	48	37	31	25	47
3	0	0	0	27	47	34	27	32	47	49	49	25	40	26	45	29	43	30	35	36
4	0	0	0	0	47	42	25	28	32	25	27	27	39	37	46	27	25	45	49	28
5	0	0	0	0	0	34	44	41	26	38	28	38	36	26	49	38	35	25	48	44
6	0	0	0	0	0	0	25	36	49	42	42	49	50	30	45	38	41	39	31	42
7	0	0	0	0	0	0	0	33	48	37	40	50	46	33	46	35	35	37	41	41
8	0	0	0	0	0	0	0	0	32	33	37	45	36	37	49	31	36	44	36	41
9	0	0	0	0	0	0	0	0	0	50	33	37	29	27	35	46	31	36	40	43
10	0	0	0	0	0	0	0	0	0	0	37	36	46	48	34	48	50	34	44	37
11	0	0	0	0	0	0	0	0	0	0	0	28	44	29	42	38	42	45	40	48
12	0	0	0	0	0	0	0	0	0	0	0	0	32	49	50	46	30	41	25	46
13	0	0	0	0	0	0	0	0	0	0	0	0	0	27	37	32	45	32	34	44
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	26	41	30	30	39
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	32	44	25	31
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	26	43	26
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	50	43
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	28
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3. Mapping to Discrete Decision Variables
   The continuous particle positions x are then transformed into discrete decision variables y, which represent the number of each item selected. The transformation is done using Equation 2:

$$y_j = lb_j + \lfloor x_j \times (ub_j - lb_j) \rfloor ; \quad j = 1, 2, \ldots, n \tag{2}$$

   where y_j is quantity of item j to be selected, lb_j is minimum allowable quantity for item j, and ub_j is maximum allowable quantity for item j.
4. Constraint Verification
   Each solution vector y must satisfy the constraints imposed by the MQBKMC model, as in Equations 3 to 6:

$$\sum_{j=1}^{n} w_j y_j \leq C \tag{3}$$

$$\sum_{j=1}^{n} v_j y_j \leq S \tag{4}$$

$$\sum_{j=1}^{n} b_j y_j \leq M \tag{5}$$

$$y_j \in \mathbb{Z}; \quad lb_j \leq y_j \leq ub_j; \quad j = 1, 2, \ldots, n \tag{6}$$

   where w_j is weight of item j, v_j is volume of item j, b_j is purchase price of item j, C is total weight capacity of the container, S is total volume capacity of the container, and M is budget available for purchasing items.
5. Fitness Evaluation
   For each candidate solution, the fitness value is calculated based on the total profit, which comprises both the individual profits of selected items and the additional profits from item interactions. The objective function is as in Equation 7:

$$Z = \sum_{j=1}^{n} p_j y_j + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} p_{ij} y_i y_j \tag{7}$$

   where Z is total profit to be maximized, p_j is profit of item j, and p_{ij} is profit from the interaction between items i and j.
6. Best and Worst Solution Identification
   Within each iteration, the algorithm identifies the best solution (Best Solution, BS) and the worst solution (Worst Solution, WS) based on the evaluated fitness values. The fitness values are ranked from highest to lowest, with the top three fitness values further utilized in the decay calculations for the alpha, beta, and gamma parameters, which guide the particle updates in EVO.
7. Calculation of Average Evaluation and Stability Level
   The algorithm computes the average objective value (EB), and the stability level (SL_i) for each candidate solution in Equations 8 and 9 as follows:

$$EB = \frac{\sum_{i=1}^{n} \text{NEL}_i}{n} \tag{8}$$

$$SL_i = \frac{\text{NEL}_i - BS}{WS - BS} \tag{9}$$

   where NEL_i is objective value (total profit) for candidate solution i, BS is best (highest) objective value in the current population, WS is worst (lowest) objective value in the current population, and SL_i is stability level for candidate i, indicating its relative quality.

8. Position Update Mechanism
After evaluating the fitness of each solution NEL_i, the algorithm proceeds with different update strategies based on the relationship between the fitness value and the average objective value EB:

1) If $NEL_i > EB$:
The solution is of above-average quality. The algorithm then compares the stability level SL_i with a stability boundary (SB).

a) If $SL_i > SB$
Two types of decay, alpha and gamma, are performed simultaneously:

I. Alpha Decay:
It occurs when the particle satisfies $SL_i > SB$ and $NEL_i > EB$. At this stage, a portion of the solution variables is refined by replacing their components with those of the best particle (X_{BS}). Two parameters are employed: Alpha Index I determines the number of variables that may be refined $[1, d]$, while Alpha Index II specifies the number of variables that must be refined $[1, \text{Alpha Index II}]$. The selected variables are randomly replaced from X_{BS} to enhance the stability of the solution, and the new position of particle i is updated in Equation 10:

$$X_i^{\text{new1}} = X_i \left(X_{BS}(x_i^j) \right); \quad i = 1, 2, \ldots, n; \quad j = \text{Alpha Index II} \tag{10}$$

where $X_i$is the current position of particle i, X_{BS} is the best-known solution, and x_i^j is the value of the j-th decision variable.

II. Gamma Decay:
This phase takes place concurrently with the alpha decay process. In this stage, part of the solution variables is adjusted by replacing certain components with those of a neighboring particle (X_{NG}) that is spatially close to the current particle. Two parameters are defined: Gamma Index I specifies the range of variables that can potentially be modified $[1, d]$, while Gamma Index II indicates the subset of variables that must be modified $[1, \text{Gamma Index II}]$. The chosen variables are randomly substituted from X_{NG} to simulate particle–environment interaction and promote improved local exploration. This decay considers the distance to the nearest neighbor (X_{NG}), as shown in Equations 11 and 12:

$$D_i^k = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} \tag{11}$$

$$X_i^{\text{new2}} = X_i \left(X_{NG}(x_i^j) \right); \quad i = 1, 2, \ldots, n; \quad j = \text{Gamma Index II} \tag{12}$$

Here (x_i, y_i) and (x_k, y_k) are the coordinates of particle i and its neighbor k.

b) if $SL_i \leq SB$
Two types of beta decay are performed:

I. Beta Decay 1
The position is updated by considering both the best solution (X_{BS}) and the population center (X_{CP}), calculated in Equations 13 and 14 as follows:

$$X_{CP} = \frac{\sum_{i=1}^{n} X_i}{n}; \quad i = 1, 2, \ldots, n \tag{13}$$

$$X_i^{\text{new1}} = X_i + \frac{r_1 \times X_{BS} - r_2 \times X_{CP}}{SL_i}; \quad i = 1, 2, \ldots, n \tag{14}$$

where r_1 and r_2 are random numbers in the range $[0, 1]$.

II. Beta Decay 2
The position is also updated toward both the best solution and the nearest neighbor in Equation 15:

$$X_i^{\text{new2}} = X_i + (r_3 \times X_{BS} - r_4 \times X_{NG}); \quad i = 1, 2, \ldots, n \tag{15}$$

where r_3 and r_4 are random numbers in the range $[0, 1]$.

2) If $NEL_i \leq EB$
The particle is considered to have a lower N/Z ratio, simulating electron capture. Its position is updated in Equation 16 as:

$$X_i^{\text{new}} = X_i + r; \quad i = 1, 2, \ldots, n \tag{16}$$

where r is a random number in the range $[0, 1]$.

9. Update and Iteration Control
   a. Best Solution Update: After each iteration, update the current best solution (X_{BS}) identified within the population.
   b. Function Evaluation Increment: Increase the function evaluation counter (FES = FES + 1)
   c. Termination Condition: If FES = FES_{max}, terminate the algorithm; otherwise, proceed to step 4.
10. Parameter Initialization
At the beginning of the optimization process, the algorithm requires the definition of several key parameters. These include n (number of particles), d (number of decision variables), $x^j_{(i,\text{min})}$ and $x^j_{(i,\text{max})}$ (lower and upper bounds for the j-th variable), and FES_{max} (maximum number of function evaluations allowed during the optimization process).
11. Generation of Initial Solutions
An initial population of n particles is generated randomly to serve as candidate solutions. Each particle's position is represented as a vector in a matrix of size $n \times d$. The initial value for each particle is calculated using Equation 17:

$$x_i^j = x^j_{(i,\text{min})} + \text{rand}(x^j_{(i,\text{max})} - x^j_{(i,\text{min})}); \quad i = 1, 2, 3, \ldots, n; \quad j = 1, 2, 3, \ldots, d \tag{17}$$

12. Mapping to Discrete Decision Variables
The continuous particle positions x are then transformed into discrete decision variables y, which represent the number of each item selected. The transformation is done using Equation 18:

$$y_j = lb_j + \lfloor x_j \times (ub_j - lb_j) \rfloor; \quad j = 1, 2, \ldots, n \tag{18}$$

where y_j is quantity of item j to be selected, lb_j is minimum allowable quantity for item j, and ub_j is maximum allowable quantity for item j.
13. Constraint Verification
Each solution vector y must satisfy the constraints imposed by the MQBKMC model, as in Equations 19 to 22:

$$\sum_{j=1}^{n} w_j y_j \leq C \tag{19}$$

$$\sum_{j=1}^{n} v_j y_j \leq S \tag{20}$$

$$\sum_{j=1}^{n} b_j y_j \leq M \tag{21}$$

$$y_j \in \mathbb{Z}; \quad lb_j \leq y_j \leq ub_j; \quad j = 1, 2, \ldots, n \tag{22}$$

where w_j is weight of item j, v_j is volume of item j, $b_j$is purchase price of item j, C is total weight capacity of the container, $S$is total volume capacity of the container, and M is budget available for purchasing items.

14. **Fitness Evaluation**
For each candidate solution, the fitness value is calculated based on the total profit, which comprises both the individual profits of selected items and the additional profits from item interactions. The objective function is as in Equation 23:

$$Z = \sum_{j=1}^{n} p_j y_j + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} p_{ij} y_i y_j \tag{23}$$

where Z is total profit to be maximized, p_j is profit of item j, $p_{ij}$is profit from the interaction between items i and j.

15. Best and Worst Solution Identification
Within each iteration, the algorithm identifies the best solution (Best Solution, BS) and the worst solution (Worst Solution, WS) based on the evaluated fitness values. The fitness values are ranked from highest to lowest, with the top three fitness values further utilized in the decay calculations for the alpha, beta, and gamma parameters, which guide the particle updates in EVO.
16. Calculation of Average Evaluation and Stability Level
The algorithm computes the average objective value (EB), and the stability level (SL_i) for each candidate solution in Equations 24 and 25 as follows:

$$EB = \frac{\sum_{i=1}^{n} \text{NEL}_i}{n} \tag{24}$$

$$SL_i = \frac{\text{NEL}_i - BS}{WS - BS} \tag{25}$$

where NEL_i is objective value (total profit) for candidate solution i, BS is best (highest) objective value in the current population, WS is worst (lowest) objective value in the current population, and SL_i is stability level for candidate i, indicating its relative quality.

17. Position Update Mechanism
After evaluating the fitness of each solution NEL_i, the algorithm proceeds with different update strategies based on the relationship between the fitness value and the average objective value EB:

1) If $\text{NEL}_i > EB$:
The solution is of above-average quality. The algorithm then compares the stability level SL_i with a stability boundary (SB).
   a) If $SL_i > SB$
   Two types of decay, alpha and gamma, are performed simultaneously:
      I. Alpha Decay:
      It occurs when the particle satisfies $SL_i > SB$ and $\text{NEL}_i > EB$. At this stage, a portion of the solution variables is refined by replacing their components with those of the best particle (X_{BS}). Two parameters are employed: Alpha Index I determines the number of variables that may be refined $[1, d]$, while Alpha Index II specifies the number of variables that must be refined $[1, \text{Alpha Index II}]$. The selected variables are randomly replaced from X_{BS} to enhance the stability of the solution, and the new position of particle i is updated in Equation 26:

$$X_i^{\text{new1}} = X_i \left(X_{BS}(x_i^j) \right); \quad i = 1, 2, \ldots, n; \quad j = \text{Alpha Index II} \tag{26}$$

      where X_i is the current position of particle i, X_{BS} is the best-known solution, x_i^j is the value of the j-th decision variable.
      II. Gamma Decay:
      This phase takes place concurrently with the alpha decay process. In this stage, part of the solution variables is adjusted by replacing certain components with those of a neighboring particle (X_{NG}) that is spatially close to the current particle. Two parameters are defined: Gamma Index I specifies the range of variables that can potentially be modified $[1, d]$, while Gamma Index II indicates the subset of variables that must be modified $[1, \text{Gamma Index II}]$. The chosen variables are randomly substituted from X_{NG} to simulate particle–environment interaction and promote improved local exploration. This decay considers the distance to the nearest neighbor (X_{NG}), as shown in Equations 27 and 28:

$$D_i^k = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} \tag{27}$$

$$X_i^{\text{new2}} = X_i \left(X_{NG}(x_i^j) \right); \quad i = 1, 2, \ldots, n; \quad j = \text{Gamma Index II} \tag{28}$$

      Here (x_i, y_i) and (x_k, y_k) are the coordinates of particle i and its neighbor k.
   b) If $SL_i \leq SB$
   Two types of beta decay are performed:
      I. Beta Decay 1
      The position is updated by considering both the best solution (X_{BS}) and the population center (X_{CP}), calculated in Equations 29 and 30 as follows:

$$X_{CP} = \frac{\sum_{i=1}^{n} X_i}{n}; \quad i = 1, 2, \ldots, n \tag{29}$$

$$X_i^{\text{new1}} = X_i + \frac{r_1 \times X_{BS} - r_2 \times X_{CP}}{SL_i}; \quad i = 1, 2, \ldots, n \tag{30}$$

      where r_1 and r_2 are random numbers in the range $[0, 1]$.
      II. Beta Decay 2
      The position is also updated toward both the best solution and the nearest neighbor in Equation 31:

$$X_i^{\text{new2}} = X_i + (r_3 \times X_{BS} - r_4 \times X_{NG}); \quad i = 1, 2, \ldots, n \tag{31}$$

      where r_3 and r_4 are random numbers in the range $[0, 1]$.

2) If $\text{NEL}_i \leq EB$
The particle is considered to have a lower N/Z ratio, simulating electron capture. Its position is updated in Equation 32 as:

$$X_i^{\text{new}} = X_i + r; \quad i = 1, 2, \ldots, n \tag{32}$$

where r is a random number in the range $[0, 1]$.

18. Update and Iteration Control
    a. Best Solution Update: After each iteration, update the current best solution (X_{BS}) identified within the population.
    b. Function Evaluation Increment: Increase the function evaluation counter (FES = FES + 1).
    c. Termination Condition: If FES = FES_{max}, terminate the algorithm; otherwise, proceed to step 4.

3. Results and Discussion

This section presents a comprehensive evaluation of the MQBKMC model utilizing a dataset comprising 20 item types. The analysis investigates the influence of two critical parameters of the Energy Valley Optimizer (EVO): the population size ($nParticle$) and the maximum number of function evaluations (FES_{max}) on solution quality, computational efficiency, and result stability. All computational experiments were performed using MATLAB 2023 [45].

3.1. Effect of Particle Number ($nParticle$)

To systematically assess the influence of population size on the performance of the Energy Valley Optimizer (EVO), first, we fixed the maximum number of function evaluations FES_{max} at 20,000, as the impact of this parameter was not yet established. For each $nParticle$ setting, 20 independent runs were conducted to ensure the statistical robustness of the results.

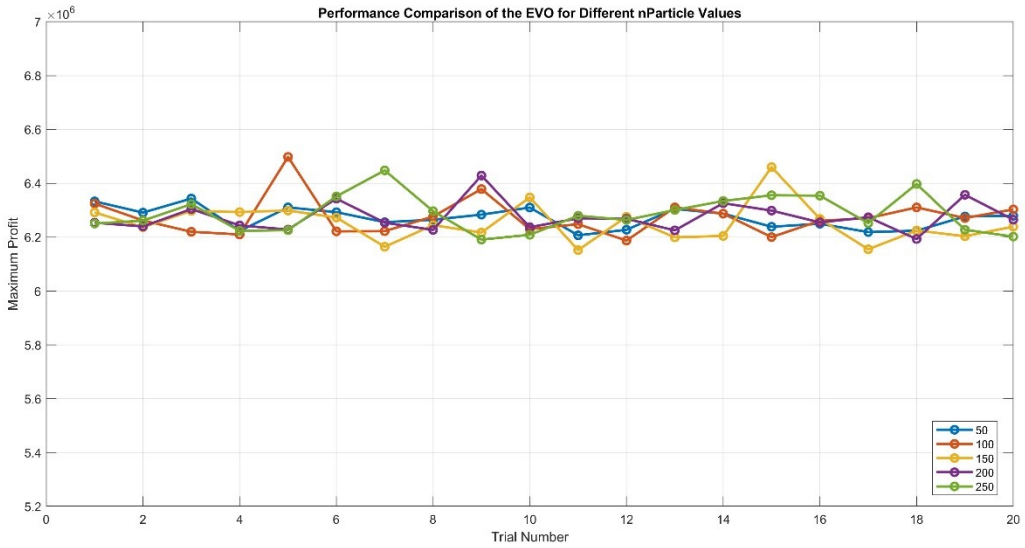

Figure 1. Maximum profit achieved across 20 independent trials for each particle population size ($nParticle$), illustrating the effect of population size on solution quality.

As illustrated in Figure 1, the maximum profit obtained across 20 trials for each $nParticle$ configuration exhibits only minor fluctuations and remains within a relatively narrow and stable range. The absence of a consistent upward or downward trend indicates that increasing the number of particles does not significantly influence the solution quality produced by the Energy Valley Optimizer (EVO). This stability suggests that EVO can maintain reliable performance regardless of population size.

This observation is further supported by the summarized results in Table 4, where the average profit values range closely from Rp6,251,559 to Rp6,286,994. Such a narrow interval reinforces the conclusion that varying the number of particles has minimal impact on the optimizer's ability to generate high-quality solutions for the MQBKMC problem. Despite the differences in population sizes, the algorithm consistently converges to solutions of comparable quality.

In terms of computational effort, a moderate increase in computation time is observed as the number of particles is raised—from 70.140 seconds at $nParticle = 50$ to 75.727 seconds at $nParticle = 250$. Although this represents a gradual increase, the overall difference remains relatively small. This indicates that while larger populations do introduce additional computational workload, the impact on runtime remains manageable within the tested range.

Moreover, the deviation percentages across 20 trials remain low for all configurations, consistently below 1%. This low variability demonstrates the algorithm's robustness and its ability to produce repeatable results. The smallest deviation occurs at $nParticle = 100$ (0.575%), whereas the largest is observed at $nParticle = 250$ (0.956%). Even so, these differences are minimal and do not indicate any instability in the algorithm's behavior.

Although the profit values slightly increase with larger particle populations, the improvements are relatively small across all configurations. The setting with $nParticle = 250$ achieves the highest profit, though it also shows a slightly higher deviation percentage. Overall, $nParticle = 250$ remains a reasonable choice for subsequent experiments, as it provides the best profit with only a minor increase in computational time, despite not being the most statistically stable configuration.

Table 4. Effect of particle number ($nParticle$) on total profit, computation time, and deviation percentage.

($nParticle$)	**Profit**	**Computation Time (s)**	**Deviation Percentage**
50	Rp6,274,137	70.140	0.603%
100	Rp6,251,559	71.729	0.575%
150	Rp6,274,572	73.083	0.712%
200	Rp6,273,559	75.053	0.828%
250	Rp6,286,994	75.727	0.956%

3.2. Effect of Maximum Function Evaluation FES_{max}

To evaluate the influence of the maximum number of function evaluations (FES_{max}) on the performance of the Energy Valley Optimizer (EVO), a series of experiments was conducted using a fixed particle population of 250, as established in the previous analysis. The outcomes of these experiments are comprehensively presented in Figure 2 and Table 5, which summarize the algorithm's performance across five different FES_{max} configurations, each assessed over 20 independent trials. Together, these visual and numerical results offer clear insights into how increasing the evaluation budget affects both solution quality and computational effort.

As shown in Figure 2, increasing the FES_{max} leads to a consistently upward trend in the maximum profit achieved across trials. This pattern is further supported by the quantitative results in Table 5, where the average profit increases substantially from Rp5,893,437 at $FES_{max} = 1,000$ to Rp6,285,271 at $FES_{max} = 200,000$. This monotonic improvement reflects the benefits of allocating a larger number of function evaluations, enabling the EVO to explore the search space more thoroughly, avoid premature convergence, and ultimately reach higher-quality solutions. The clear separation among the profit curves in Figure 2 reinforces this observation, illustrating how larger evaluation thresholds consistently yield better optimization outcomes.

However, this improvement in solution quality is accompanied by a considerable increase in computation time. According to Table 5, computation time rises sharply with increasing FES_{max}, from only 1.493 seconds

at $\text{FES}_{\max} = 1,000$ to 77.884 seconds at $\text{FES}_{\max} = 200,000$. Despite this increase, the deviation percentage across trials remains low and stable, with all values staying below 1.3%. The relatively narrow spread of profit values in Figure 2 further confirms this consistency, indicating that even at high evaluation limits, the EVO produces reliable and reproducible results.

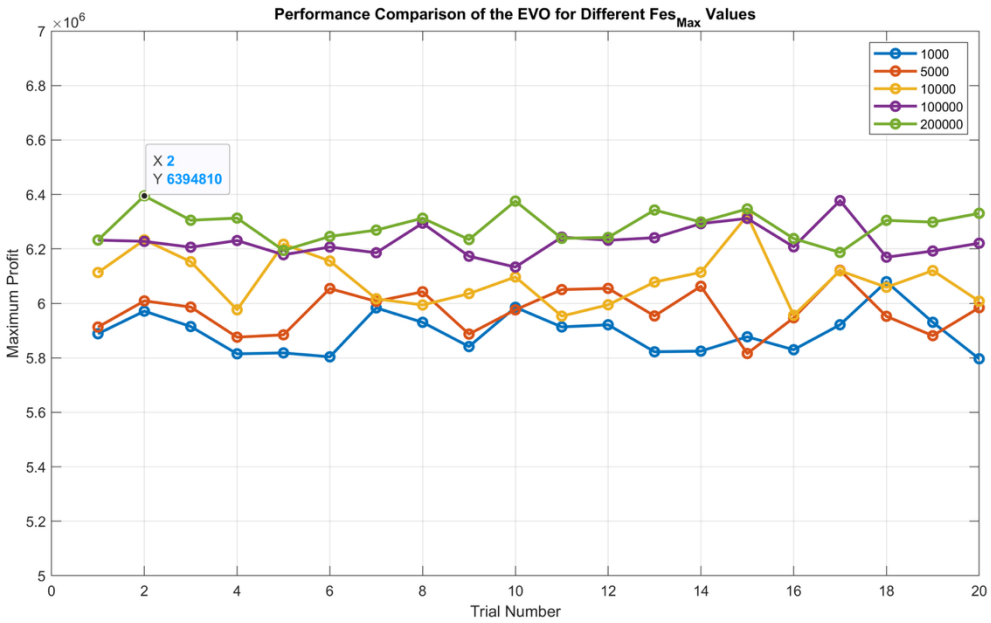

Figure 2. Maximum profit obtained across 20 independent trials for varying values of maximum function evaluations $\text{FES}_{\max}$, highlighting the impact of evaluation limits on optimization performance.

The combined insights from Figure 2 and Table 4 highlight the importance of selecting an appropriate evaluation threshold when applying metaheuristic algorithms to complex optimization problems. While larger $\text{FES}_{\max}$ values significantly enhance the effectiveness and robustness of the EVO, they also require substantially greater computational resources. Therefore, in practical applications, it is crucial to balance the trade-off between solution quality and computational cost. Excessively large evaluation thresholds may yield diminishing returns relative to their computational expense, making the choice of $\text{FES}_{\max}$ a key consideration in the practical deployment of the EVO.

Table 5. Effect of maximum function evaluations ($\text{FES}_{\max}$) on total profit, computation time, and deviation percentage.

$\text{FES}_{\max}$	Profit	Computation Time (s)	Deviation Percentage
1000	Rp5,893,437	1.493	1.048%
5000	Rp5,973,054	0.824	1.082%
10000	Rp6,085,901	3.915	1.293%
100000	Rp6,227,851	38.978	0.650%
200000	Rp6,285,271	77.884	0.660%

Overall, the parameter analysis demonstrates that increasing the number of particles beyond a moderate threshold does not yield significant improvements in solution quality, while higher values of $\text{FES}_{\max}$ generally lead to higher profits, but with diminishing returns and greater computational cost. Importantly, after a certain point, further increasing $\text{FES}_{\max}$ results in only marginal improvements, suggesting that excessively large evaluation limits may not be necessary. The algorithm's performance remains stable and consistent across multiple independent trials, as

indicated by the low deviation percentages observed in the results. The convergence behavior of EVO for example in trial 2, as illustrated in Figure 3, further confirms the algorithm's efficiency: the best fitness values are typically achieved early in the search, with later function evaluations contributing little additional improvement. This rapid and stable convergence highlights EVO's robustness and reliability for solving the MQBKMC under the tested parameter settings.

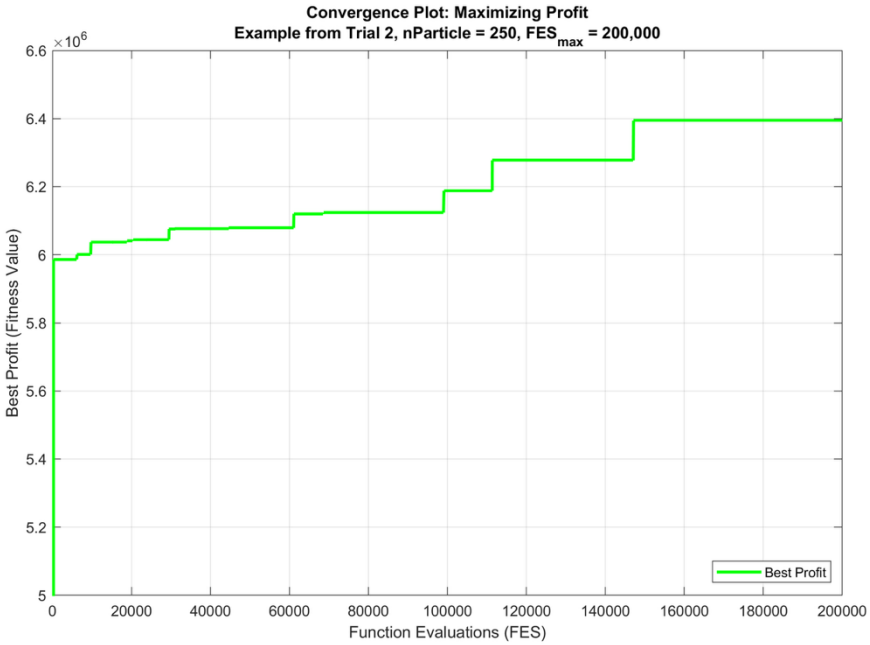

Figure 3. Convergence plot of the Energy Valley Optimizer (EVO) for the MQBKMC in trial 2 ($nParticle = 250$, $\text{FES}_{\max} = 200,000$), showing rapid achievement of best profit with minimal improvement in later evaluations.

For benchmarking purposes, the performance of the EVO was further compared with that of the Particle Swarm Optimization (PSO) algorithm under identical settings. The results, summarized in Table 5, indicate that EVO consistently achieved higher average profits and lower deviations across all test instances, while maintaining comparable computation times. This confirms the better exploration–exploitation balance of the EVO compared to PSO.

As illustrated in Figure 4, the comparison between EVO and PSO across different particle sizes highlights a clear performance distinction between the two algorithms. The results show that EVO consistently achieves higher maximum profit values than PSO for every tested population size, demonstrating its better capability in handling the MQBKMC optimization problem.

The performance patterns of both algorithms exhibit a similar trend: as the number of particles increases from 50 to 200, the maximum profit steadily improves. EVO shows its peak performance at a particle size of 250, whereas PSO reaches its highest value at 200 particles. Despite these variations, EVO maintains a consistently higher profit curve across all configurations, indicating its stronger balance between exploration and exploitation during the search process.

An additional observation from Figure 4 is the magnitude of separation between the two curves. This consistent gap suggests that EVO is more effective at navigating complex landscapes and avoiding premature convergence compared to PSO. While PSO demonstrates gradual improvement as the population grows, its maximum profit remains below that of EVO, signaling limitations in its ability to exploit promising regions of the search space as effectively.

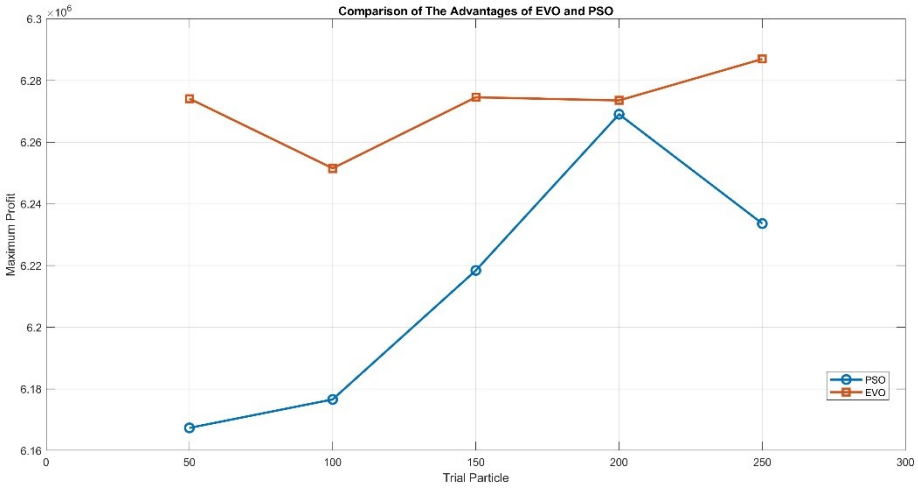

Figure 4. Comparison of the Maximum Profit between EVO and PSO under Different $nParticle$.

Overall, Figure 4 reinforces the advantage of EVO over PSO in terms of solution quality and robustness. The consistently higher profit values obtained by EVO confirm its enhanced capability to escape local optima and converge toward more optimal solutions for the MQBKMC problem.

4. Conclusion

This study demonstrates that the Energy Valley Optimizer (EVO) is an effective and reliable metaheuristic approach for solving the Modified Quadratic Bounded Knapsack Problem with Multiple Constraints (MQBKMC). The parameter analysis reveals that increasing the particle population size may yield slightly better profit values. In contrast, increasing the maximum number of function evaluations (FES_{max}) consistently enhances solution quality, with the best objective value reaching Rp6,286,994, although with diminishing returns at higher computational costs. EVO achieves most of its optimal profits early in the search process, confirming its rapid convergence and strong exploration–exploitation balance.

Comparative benchmarking with the Particle Swarm Optimization (PSO) algorithm further indicates that EVO tends to perform better than PSO, achieving higher average profits and lower deviation percentages under identical experimental settings. Overall, EVO provides a stable, efficient, and scalable framework for tackling complex combinatorial optimization problems, particularly when parameters are carefully tuned to balance accuracy and efficiency. Future work will focus on evaluating EVO's scalability using larger benchmark instances and comparing its performance with other advanced metaheuristic algorithms to further validate its robustness.

Acknowledgment

We want to express our deepest gratitude to the Lembaga Penelitian dan Pengabdian (LP2M) of the University of Jember, through the 2025 Internal Grant scheme of the Postgraduate Program, for the support, resources, and facilities provided in implementing this research under the Research Contract Number: 2840/UN25.3.1/LT/2025. Thanks to this assistance, this research was able to proceed smoothly. We also appreciate the invaluable guidance and contributions from the researchers and staff involved, especially our colleagues especially our colleagues at the Keris-Dimas, Mathematical Optimization and Computations (MOCo), Department of Mathematics, who provided

valuable insights and suggestions throughout the research process., Department of Mathematics, who provided valuable insights and advice throughout the research process.

Source Code Availability Statement

The source code presented in this study are available upon reasonable request from the authors.

REFERENCES

1. D. Connolly, *Knapsack Problems: Algorithms and Computer Implementations*, Journal of the Operational Research Society, vol. 42, no. 6, pp. 513, 1991.
2. D. Pisinger, *A Minimal Algorithm For The Multiple-Choice Knapsack Problem Author Links Open Overlay Paneldavid Pisinger*, Eur. J. Oper. Res., vol. 83, no. 2, pp. 394—410, 1995.
3. H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problem*, 2004.
4. Y. D. Regita, K. Santoso, and A. Kamsyakawuni, *Elephant Herding Optimization Algorithm: Multiple Constraints Knapsack 0-1 Problems*, Maj. Ilm. Mat. dan Stat., vol. 18, no. 1, pp. 13–22, 2018.
5. R. A. Abdullah, A. Riski, A. Kamsyakawuni, and J. Matematika, *Implementation Of Penguins Search Optimization (PESOA) And Migrating Birds Optimization (MBO) Algorithms On Knapsack*, Maj. Ilm. Mat. dan Stat., vol. 19, pp. 75–84, 2019.
6. B. Sun, A. Zeynali, T. Li, M. Hajiesmaili, A. Wierman, and D. H. K. Tsang, *Competitive Algorithms for the Online Multiple Knapsack Problem with Application to Electric Vehicle Charging*, Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 3, pp. 1-–32, 2021.
7. D. Patel, A. Khan, and A. Louis, *Aspects of total variation regularized ℓ_1 function approximation*, SIAM Journal on Applied Mathematics, vol. 65, pp. 1817–1837, 2005.
8. A. Zeynali, B. Sun, M. Hajiesmaili, and A. Wierman, *Data-driven Competitive Algorithms for Online Knapsack and Set Cover*, Proc. AAAI Conf. Artif. Intell., vol. 35, no. 12, pp. 10833–10841, 2021.
9. F. B. Ozsoydan and İ. Gölcük, *A Reinforcement Learning Based Computational Intelligence Approach For Binary Optimization Problems: The Case Of The Set-Union Knapsack Problem*, Eng. Appl. Artif. Intell., vol. 118, p. 105688, 2023.
10. C. Jin, *0-1 Knapsack in Nearly Quadratic Time*, Proc. Annu. ACM Symp. Theory Comput., pp. 271–282, 2024.
11. B. Wu, W. Bao, and B. B. Zhou, *Augmenting Online Algorithms for Knapsack Problem with Total Weight Information*, Proc. AAAI Conf. Artif. Intell., vol. 39, no. 25, pp. 26725–26732, 2025.
12. Wilkening, S., Lefterovici, A.-I., Binkowski, L., Funck, M., Perk, M., Karimov, R., Fekete, S., & Osborne, T. J., *A quantum search method for quadratic and multidimensional knapsack problems*, 2025.http://arxiv.org/abs/2503.22325.
13. L. F. M. López, N. G. Blas, and A. A. Albert, *Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations*, Soft Comput., vol. 22, no. 8, pp. 2567–2582, 2018.
14. J. García, I. Cattarinich, P. Moraga, and H. Pinto, *Exploring the Impact of Local Operator Configurations in the Multi-Demand Multidimensional Knapsack Problem*, Appl. Sci., vol. 15, no. 4, p. 2059, 2025.
15. M. Vasquez and J. Hao, *A hybrid approach for the 0 – 1 multidimensional knapsack problem*, in Proceeding of the 17th International Joint Conference on Artificial Intelligence, 2001, pp. 328–333.
16. M. Abdel-Basset, R. Mohamed, K. M. Sallam, I. Alrashdi, and I. A. Hameed, *An efficient binary spider wasp optimizer for multi-dimensional knapsack instances: experimental validation and analysis*, J. Big Data, vol. 12, no. 1, 2025.
17. A. Ambarwati, S. Abusini, and V. H. Krisnawati, *Optimizing Knapsack Allocation: The Preemptive Multiple Bounded Knapsack Problem*, in Proceedings of the First International Conference on Applied Mathematics, Statistics, and Computing (ICAMSAC 2023), Atlantis Press International BV, 2024, pp. 214–220.
18. K. A. Santoso, M. B. Kurniawan, A. Kamsyakawuni, and A. Riski, *Hybrid Cat-Particle Swarm Optimization Algorithm on Bounded Knapsack Problem with Multiple Constraints*, Proc. Int. Conf. Math. Geom. Stat. Comput. (IC-MaGeStiC 2021), vol. 96, 2022.
19. I. Maris, A. Pradjaningsih, and K. A. Santoso, *Application Of Combined GSA& sCSO Algorithm To Modified Bounded Knapsack With Multiple Constraints Problem Against Uncertain Coefficient*, in The 3rd International Conference On Science, Mathematics, Environment, And Education, 2023, p. 080011.
20. N. Dogan, K. Bilgiçer, and T. Saraç, *Quadratic Multiple Knapsack Problem with Setups and a Solution Approach*, no. 2006, pp. 613–622, 2012.
21. T. Saraç and A. Sipahioglu, *Generalized Quadratic Multiple Knapsack Problem and Two Solution Approaches*, Comput. Oper. Res., vol. 43, no. 1, pp. 78–89, 2014.
22. V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, *Knapsack Problems — An Overview of Recent Advances. Part II: Multiple, Multidimensional, and Quadratic Knapsack Problems*, Comput. Oper. Res., vol. 143, p. 105693, 2022.
23. F. D. Fomeni and A. N. Letchford, *A Dynamic Programming Heuristic For The Quadratic Knapsack Problem*, INFORMS J. Comput., vol. 26, no. 1, pp. 173–182, 2014.
24. D. S. Hochbaum, P. Baumann, O. Goldschmidt, and Y. Zhang, *A Fast and Effective Breakpoints Heuristic Algorithm For The Quadratic Knapsack Problem*, Eur. J. Oper. Res., vol. 323, no. 2, pp. 425–440, 2025.
25. Y. E. Pratiwi, F. Ubaidillah, and M. Fatekurohman, *Quadratic Bounded Knapsack Problem Solving with Particle Swarm Optimization and Golden Eagle Optimization*, Int. J. Adv. Eng. Res. Sci., vol. 9, no. 7, 2022.
26. L. Galli, S. Martello, and P. Toth, *The quadratic knapsack problem*, Eur. J. Oper. Res., vol. 326, no. 1, pp. 1–12, 2025.

27. Z. Li, Y. He, H. Li, Y. Li, and X. Guo, *A novel discrete grey wolf optimizer for solving the bounded knapsack problem*, Commun. Comput. Inf. Sci., vol. 986, pp. 101–114, 2019.
28. J. Tang, X. Tang, A. Lim, K. Han, C. Li, and J. Yuan, *Revisiting Modified Greedy Algorithm for Monotone Submodular Maximization with a Knapsack Constraint*, Proc. ACM Meas. Anal. Comput. Syst., vol. 5, no. 1, pp. 1–22, 2021.
29. N. Moradi, V. Kayvanfar, and M. Rafiee, *An efficient population-based simulated annealing algorithm for 0–1 knapsack problem*, Eng. Comput., vol. 38, no. 3, pp. 2771–2790, 2022.
30. M. Anditta, N. Amartya, L. S. Warnars, H. L. H. S. Warnars, A. Ramadhan, and T. Siswanto, *Dynamic Programming Algorithm using Furniture Industry 0/1 Knapsack Problem*, 2023 5th Int. Conf. Cybern. Intell. Syst., 2023.
31. K. A. Santoso, I. M. Ilmiyah, and A. Pradjaningsih, *Optimizing the Arrangement of Goods in Box Van Using the Tabu Search Algorithm*, Stat. Optim. Inf. Comput., vol. x, pp. 0–7, 2024.
32. A. Ambarwati, S. Abusini, and V. H. Krisnawati, *The Branch And Bound Approach To A Bounded Knapsack Problem (Case Study: Optimizing Of Pencak Silat Match Sessions)*, BAREKENG J. Math. Its Appl., vol. 18, no. 4, pp. 2449–2458, 2024.
33. I. Boussaïd, J. Lepagnot, and P. Siarry, *A Survey On Optimization Metaheuristics*, Inf. Sci. (Ny)., vol. 237, no. February, pp. 82–117, 2013.
34. A. H. Gandomi, X. S. Yang, S. Talatahari, and A. Alavi, *Metaheuristic Applications in Structures and Infrastructures*, London: Elsevier, 2013.
35. A. Pradjaningsih and L. H. R. Romadhoni, *Firefly Algorithm-Optimized SVR Framework for Accurate Stock Price Forecasting*, Stat. Optim. Inf. Comput., vol. 14, no. 3, pp. 1403–1418, 2025.
36. M. Azizi, U. Aickelin, H. A. Khorshidi, and M. Baghalzadeh Shishehgarkhaneh, *Energy Valley Optimizer: A Novel Metaheuristic Algorithm For Global And Engineering Optimization*, Sci. Rep., vol. 13, no. 1, pp. 1–23, 2023.
37. Azad, M. A., Sajid, I., Lu, S. Der, Sarwar, A., Tariq, M., Ahmad, S., Liu, H. D., Lin, C. H., & Mahmoud, H. A., *Energy Valley Optimizer (EVO) for Tracking the Global Maximum Power Point in a Solar PV System under Shading*, Processes, vol. 11, no. 10, 2023. https://doi.org/10.3390/pr11102986
38. S. M. K. Sarkhi and H. Koyuncu, *Optimization Strategies for Atari Game Environments: Integrating Snake Optimization Algorithm and Energy Valley Optimization in Reinforcement Learning Models*, Ai, vol. 5, no. 3, pp. 1172–1191, 2024.
39. B. E. Elnaghi, A. M. Ismaiel, F. El Sayed Abdel-Kader, M. N. Abelwhab, and R. H. Mohammed, *Validation of energy valley optimization for adaptive fuzzy logic controller of DFIG-based wind turbines*, Sci. Rep., vol. 15, no. 1, pp. 1–25, 2025.
40. M. Toghani, S. Maleki, A. Younesi, S. Safari, and S. Hessabi, *ARGENT: Energy-Aware Scheduling in Edge Computing Using Energy Valley Optimizer*, 2025 29th Int. Comput. Conf. Comput. Soc. Iran, pp. 1–6, 2025.
41. Y. Liu, P. Su, P. Qiu, T. Luo, C. Yang, and X. Lu, *Research On Gas Tunnel Prediction In Central Sichuan Using Energy Valley Optimizer And Support Vector Machine*, Bull Eng Geol Env., vol. 84, no. 50, 2025.
42. C. Li, *Landscape Ecological Design Using Elman Neural Networks And Improved Energy Valley Optimizer Algorithm*, Int. J. Low-Carbon Technol., vol. 20, no. August 2024, pp. 973–989, 2025.
43. N. C. Chinh and N. N. Tung, *Determining Optimal Installation Of Renewable Distributed Generators In The Radial Distribution System Considering Technical-Economic Aspects By Using Energy Valley Optimizer Algorithm*, COMPEL, vol. 44, no. 4, pp. 411–425, 2025.
44. Q. Zhou, Y. Pan, S. Zhai, and Z. Ye, *Wind Turbine Bearing Fault Diagnosis Method Based On Energy Valley Optimization Algorithm Optimization Multi Kernel Extreme Learning Machine*, in 2025 7th International Conference on Information Science, Electrical and Automation Engineering (ISEAE), Harbin, China: IEEE, 2025, pp. 1180–1184.
45. P. Kumar, *Energy Valley Optimizer*, 2023, MATLAB Central File Exchange. Available: https://www.mathworks.com/matlabcentral/fileexchange/163406-energy-valleyoptimizer-evo.