Improved Discrete Spider Monkey Optimization Using a Dynamic Penalty Function with Partial Constraint Violation Acceptance for Vending Machine Column Optimization Considering Sales and a Replenishment Cycle

Riko Hasegawa
School of Interdisciplinary Mathematical Sciences
Meiji University
Tokyo, Japan
ev221023@meiji.ac.jp

Yoshikazu Fukuyama
School of Interdisciplinary Mathematical Sciences
Meiji University
Tokyo, Japan
yfukuyam@meiji.ac.jp

Takuya Watanabe
Corporate R&D Headquarters
Fuji Electric Co., Ltd.
Tokyo, Japan
watanabe-takuya@fujielectric.com

Naoto Ishibashi
Corporate R&D Headquarters
Fuji Electric Co., Ltd.
Tokyo, Japan
ishibashi-naoto@fujielectric.com

Tatsuya Iizaka
Corporate R&D Headquarters
Fuji Electric Co., Ltd.
Tokyo, Japan
iizaka-tatsuya@fujielectric.com

***Abstract*—This paper proposes an improved discrete spider monkey optimization based method using a dynamic penalty function with partial constraint violation acceptance for vending machine column optimization considering sales and a replenishment cycle. The conventional method has a challenge that it only searches for feasible regions. To address the challenge, the proposed method added a dynamic penalty function in order to search for infeasible regions. Moreover, partial constraint violation acceptance rate is introduced to obtain high quality solutions. Effectiveness of the proposed method is verified by comparing with the conventional method, a method with only a static penalty function, a method with only a dynamic penalty function with actual vending machine data.**

Keywords—Improved discrete spider monkey optimization, dynamic penalty function, partial constraint violation acceptance, combinatorial optimization, vending machine column optimization, Industrial applications

I. INTRODUCTION

In Japan, there are approximately 2.19 million beverage vending machines (VMs), which have become an integral part of daily life [1]. Inside a VM, multiple storage spaces called columns are installed, and the capacity of each column is predetermined. Since demand varies for each product, allocating an insufficient number of columns to high-demand products can lead to premature stockouts and, consequently, lost sales opportunities. Therefore, the placement of products in various columns of VMs is directly linked to sales improvement. It can be formulated as a combinatorial optimization problem that considers both total sales values and a replenishment cycle. In VMs, regular replenishment is required. By allocating the larger-capacity columns to high-demand products, the replenishment cycle can be extended and the number of deliveries reduced. However, since there are various capacities of columns and various demands of products, the problem cannot be solved easily. Reducing delivery frequency offers three main advantages. First, it decreases carbon dioxide (CO_2) emissions associated with transportation. In 2022, the transportation sector accounted for 18.5% of Japan's energy-related CO_2 emissions [2]. Thus, reducing the number of deliveries is an effective measure against greenhouse gas emissions [3]. Second, it helps control transportation costs amid rising fuel prices. In recent years, fuel costs have placed a growing burden on companies, and reducing delivery frequency can mitigate this impact. Third, it helps alleviate the burden on workers. Since 2024, Japan has enforced legal regulations on overtime work [4], making improvements in work efficiency increasingly important. Taken together, reducing the number of deliveries is significant from environmental, economic, and labor perspectives.

Several approaches have been proposed for the VM column optimization problem (VMCOP). Miyamoto et al. regarded the problem as a combinatorial optimization problem and attempted optimization using Tabu Search (TS) and Multiple Start Local Search, confirming their effectiveness [5]. Takeuchi et al. formulated the problem with the objective of maximizing long-term profit margins and applied the Life Span Method, an extension of TS, to improve performance [6]. However, it has been pointed out that the solutions obtained by these methods remain limited, and that there is still room for further improvement [7-9]. Meanwhile, in the field of evolutionary computation, the usage of multi-point search algorithms with multi-population has been reported to improve solution quality [8-11]. Conventional multi-point search algorithms mainly employed static multi-populations in which the number of multi-populations was fixed and the number of search points in each population was fixed. The authors applied this framework to the VMCOP and proposed MP-IMA, a method based on static multi-populations, confirming its effectiveness [8]. More recently, methods that dynamically vary the number of search points and the population structure have emerged. It has been demonstrated that they can derive solutions superior to those obtained with traditional approaches [12-17]. Dynamic multi-population

methods are characterized by the ability to vary the number of search points within each population, adjust the number of populations, and reconfigure the group structure. Among various approaches, only Discrete Spider Monkey Optimization (DSMO) possesses all three of these features. It is therefore expected to yield superior solutions. Based on this perspective, Koyama et al. improved DSMO for the VMCOP and proposed IDSMO, verifying its effectiveness [18-20]. However, in IDSMO, all solutions that violated constraints were excluded from the search space. Such infeasible solutions (ISs) were neither evaluated nor selected. As a result, the searchable region was restricted, and the flexibility of the search process was reduced. It made broad exploration difficult. Consequently, in the VMCOP addressed in the previous studies, penalty functions have not been introduced. Solutions that violated constraints were excluded from the search space [5-9, 18-23]. In contrast, the penalty function method originated from Courant's work on the calculus of variations in 1943 [24]. It has since been integrated with evolutionary algorithms, for example through the introduction of dynamic penalty functions for nonlinear constrained optimization problems by Joines and Houck [25]. A notable feature of this approach is that it allows the evaluation of solutions that do not strictly satisfy the constraints but nevertheless yield good objective function values. This property has been regarded as effective for maintaining diversity in the early stages of the search and for avoiding premature convergence to local optima [25][26].

Based on the above background, this paper proposes the following improvement to IDSMO aiming to expand the search space while balancing solution quality and search flexibility:

1) A dynamic penalty term is introduced into the objective function based on the number of unassigned products for the constraint that each product must be assigned to at least one column (Proposal 1).

The conventional update rule, which excluded all ISs, is revised. With this revision, all solutions can be evaluated and selected according to their penalized objective function values. However, introducing the penalty term alone often led to final best solutions that did not satisfy the constraints. This caused challenges with the stability of the search outcomes. Therefore, we propose further improvements. Namely, in addition to the penalty term, we incorporate a framework in which ISs are conditionally accepted during the solution update process. In this study, two strategies for controlling the degree of constraint violation acceptance are adopted and compared:

2) *A static acceptance scheme:* ISs are accepted with a fixed rate throughout the search process (Proposal 2).

3) *A dynamic acceptance scheme:* The acceptance rate changes over time as the search progresses. In the early stages, a high acceptance rate is applied to maintain diversity, while, in the later stages, the rate is gradually reduced to guide the search toward stricter constraint satisfaction (Proposal 3).

Using actual VM data, we compared the Proposed Method 1 (Proposal 1 + Proposal 2) and the Proposed Method 2 (Proposal 1 + Proposal 3) with the conventional IDSMO approach [20]. The results confirmed the effectiveness of the both proposed methods. Furthermore, these results were validated by the Friedman test and the Wilcoxon signed-rank test with the Holm correction as a post hoc analysis.

II. A Problem Formulation Of A Vending Machine Column Optimization Problem

A. Decision Variables

This study utilizes decision variables $s_{kp}^m (s_{kp}^m \in 0,1)$ to indicate whether product p is assigned to column k of VM m. Specifically, $s_{kp}^m = 1$ means that product p is placed in column k of VM m, while $s_{kp}^m = 0$ indicates that it is not.

B. The Proposed Objective Function

The objective function in [20] aims to maximize both the replenishment cycle of products in VMs and the total sales values. In this study, the objective function is further extended by introducing a dynamic penalty term (Proposal 1), which enables the evaluation of ISs.

$$\max\left(\alpha \times \min\left(z_{cp}^m, z_{hp}^m\right) + (1-\alpha) \times Rev - Penalty\right) \quad (1)$$

$$z_{cp}^m = \sum_{k \in K_c^m} \frac{C_k^m s_{kp}^m}{D_p} (p \in P_c^m) \quad (2)$$

$$z_{hp}^m = \sum_{k \in K_h^m} \frac{C_k^m s_{kp}^m}{D_p} (p \in P_h^m) \quad (3)$$

$$Rev = \sum_m \sum_{p \in P_c^m, P_h^m} \sum_{k \in K_c^m, K_h^m} \left(C_k^m s_{kp}^m \times F_p^m\right) \quad (4)$$

$$Penalty = r_0 * \left(\frac{t}{T}\right)^e * \left(\sum_p^{N_p} \sigma_p\right)^2 \quad (5)$$

where z_{cp}^m is the time until the p-th cold product in the m-th VM sells out, z_{hp}^m is the time until the p-th hot product in the m-th VM sells out, Rev is a total sales value, α is a weighting coefficient representing the relative importance of replenishment cycle, K_c^m is the set of columns allocated to cold products in the m-th VM, D_p is the demand for product p, C_k^m is the capacity of the k-th column in the m-th VM, P_c^m is the set of cold products allocated to the m-th VM, K_h^m is the set of columns allocated to hot products in the m-th VM, P_h^m is the set of hot products allocated to the m-th VM, F_p^m is the unit price of product p in the m-th VM, r_0 is the penalty coefficient, t is the current number of objective function evaluations, T is the maximum number of objective function evaluations, e is the coefficient controlling the growth rate of the penalty term, σ_p is the indicator that takes the value 1 if product p is unassigned in all VMs, and 0 otherwise, and N_p is the number of products.

Equation (2) calculates the sales duration of cold product p in the m-th VM. It divides the total assigned column capacity by the demand for that product. Equation (3) gives the sales duration for hot product p in the same way. Equation (4) computes the total sales value. It sums the product of the assigned quantity and the unit price for all products in a VM. Equation (5) represents a dynamic penalty function for the constraint that each product must be assigned to at least one column (Proposal 1). The magnitude of the penalty changes dynamically according to the number of unassigned products and the progress of objective function evaluations. This design allows a certain degree of freedom for ISs in the early stages of the search, while gradually guiding the search toward constraint satisfaction as the search progresses. In other words, solutions with many unassigned products or ISs in the later

stages of the search receive larger penalties. Namely, solutions with larger penalties are less likely to be evaluated as favorable in terms of objective function value.

C. Constraints

$$\sum_{p \in P_c^m, P_h^m} s_{kp}^m = 1(k \in K_c^m, K_h^m) \quad (6)$$

$$\sum_{k \in K_c^m} s_{kp}^m \geq 1(p \in P_c^m) \quad (7)$$

$$\sum_{k \in K_h^m} s_{kp}^m \geq 1(p \in P_h^m) \quad (8)$$

$$s_{kp}^m = 0(p \in P_c^m, k \in K_h^m) \quad (9)$$

$$s_{kp}^m = 0(p \in P_h^m, k \in K_c^m) \quad (10)$$

Equation (6) defines the allocation constraint that every column in VM m must be assigned exactly one product. Equation (7) and (8) state that each cold product and each hot product, respectively, must be allocated to at least one column in VM m. Equation (9) prevents cold products from being assigned to columns designated for hot products. Similarly, Equation (10) prohibits hot products from being assigned to columns designated for cold products.

III. Overview of Discrete Spider Monkey Optimization with Dynamic Multi-Populations

The IDSMO utilized in this study is based on Spider Monkey Optimization (SMO) developed by Bansal et al. [15] and DSMO by Akhand et al. [27], and was further improved by Koyama et al. for the VMCOP [20]. SMO is an algorithm inspired by the social behavior of spider monkeys, known as a fission-fusion social system. It divides the population into multiple subgroups, conducts local searches within each subgroup, and then shares information at the global level. DSMO extends SMO so that it can be applied to combinatorial optimization problems. Koyama et al. incorporated operators designed for the VMCOP into this framework, thereby achieving efficient search [20]. In particular, the update rules include the Swap Operator (SO), originally proposed in DSMO, and Swap Product (SP), introduced by Koyama et al. for this problem. A solution is represented as a sequence of product numbers assigned to each column in a VM. Among all solutions generated during initialization, the one with the highest objective function value is defined as the Global Leader (GL). The best solution within each subgroup is defined as the Local Leader (LL). The hyperparameters are as follows. M_{iter} is the maximum number of iterations. MN_g is the maximum number of groups. NSM is the number of search points. pr_LL is the probability threshold for applying updates. MLL is the maximum number of consecutive iterations in which the LL can maintain the same solution. Similarly, MGL is the maximum number of consecutive iterations in which the GL can maintain the same solution. The meanings of the operators "+" and "*" in (11)-(13) follows the original DSMO formulation [27]. The procedure of IDSMO for a the VM column optimization is presented below [20].

Step 1 Initiation: The number of iterations is set as $iter = 1$, and the number of groups is set as $N_g = 1$. The number of search points in the first group is set as $NSM_1 = NSM$. All search points are generated randomly. The initial solutions are evaluated using the objective function ((1)-(5)), and the LL for each group and the GL are determined. The counters sll_1 and sgl, which record the number of times the LL and GL maintain the same solutions, are initialized to 1.

Step 2 Update search points: For each search point SM_{ia} (the a-th search point in group i), the update is performed according to the following equations.

$$SS_{ia} = U(0,1) * (GLorLL_i - SM_{ia}) + U(0,1) * (RSM - SM_{ia}) \quad (11)$$

$$SM_{newia} = SM_{ia} + SS_{ia} \quad (12)$$

where SS_{ia} is the update value for the a-th search point in group i. $U(0,1)$ is a uniform random number in the range [0, 1], $GLorLL_i$ is either the GL or the LL of group i, RSM is a search point different from SM_{ia} randomly chosen from all search points or from all other search points within group i.

In (11), when the update is directed toward the GL, $GLorLL_i$ is set to the GL, and RSM is chosen randomly from all search points except SM_{ia}. When the update is directed toward the LL, $GLorLL_i$ is set to the LL, and RSM is chosen randomly from all search points in group i except SM_{ia}. The difference operation in (11) such as $(GLorLL_i - SM_{ia})$ is transformed into a sequence of discrete operations. These operations are expressed by the SP and the SO. The SP adjusts shortages or surpluses in the number of product allocations to columns, while the SO exchanges products between different columns [20]. For the SO and the SP, the parameters pr_{SO} and pr_{SP} determine whether the operation is applied. From the first term in (11), all SO and SP operations are obtained. Similarly, all SO and SP are also obtained from the second term. In total, four objective function values are calculated. These four evaluation results are compared with the objective function value obtained before the update, and the solution with the best value is selected. According to (12), the sequence of operations SS_{ia}, which combines these results, is then applied to SM_{ia}. This produces the updated solution SM_{newia}. If the candidate solution violates the constraints ((6)-(10)), the update is not applied. Further details can be found in [20].

Step 3 Update of LL and GL: For each group, the best updated solution at Step 2 is compared with the current LL. If the best updated solution is better, the LL is updated. All LLs are then compared with the current GL. If the updated LL is better than the GL, the GL is replaced by the LL.

Step 4 Replacement of LL: If $sll_g > MLL$, a decision is made based on the probability pr_LL. The search point is either reinitialized randomly or the LL is updated according to (13).

$$SM_{newia} = U(0,1) * (GL - SM_{newia}) + U(0,1) * (SM_{newia} - LL_i) \quad (13)$$

After the LL is replaced, sll_g is reset to 1.

Step 5 Reconfiguration of groups: If $sgl > MGL$, the number of groups is increased by one, i.e., $N_g = N_g + 1$. If $N_g = MN_g$, then N_g is reset to 1. If $N_g < MN_g$, all

search points are divided into N_g groups based on the method described in [20]. This division considers both the objective function values and the Hamming distance, which represents the similarity between solutions. After this operation, the LLs of all groups and the GL are updated, and sgl is reset to 1.

Step 6 Termination check: If $iter = M_{iter}$, proceed to Step 7. Otherwise, set $iter = iter + 1$ and return to Step 2.

Step 7 Output of the solution: The final GL is output as the best solution, and the algorithm terminates.

IV. Application of a Modified Discrete Spider Monkey Optimization Using a Dynamic Penalty Function with Partial Constraint Violation Acceptance to Vending Machine Column Optimization

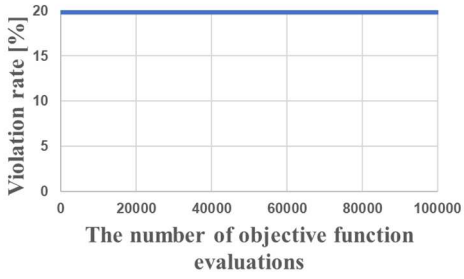
A. Acceptance Schemes for Infeasible Solutions

In general, when a penalty function is introduced to allow the search to explore ISs, all such solutions are accepted throughout the entire search process. However, in the proposed method, not all ISs are accepted during the search. Instead, acceptance schemes are proposed in which ISs are accepted only at a certain rate. Specifically, two schemes are proposed: the static acceptance scheme (Proposal 2), in which ISs are accepted at a fixed rate throughout the search, and the dynamic acceptance scheme (Proposal 3), in which the acceptance rate changes dynamically as the search progresses (See Fig. 1).

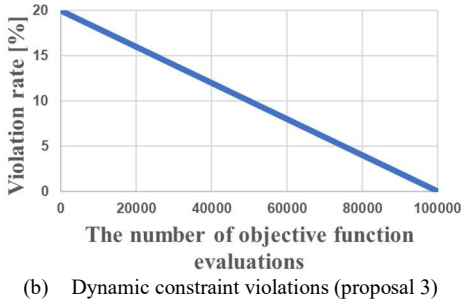
1) The static acceptance scheme (Proposal 2): In this scheme, during Step 2 in Section 3, even if a candidate solution is infeasible, it is evaluated by calculating the objective function when the condition in (14) is satisfied:

$$U[0,1] < violation_{rate} \tag{14}$$

where $violation_{rate}$ is the rate of accepting an infeasible solution, and $U[0,1]$ is a uniform random number in the range [0, 1].



(a) Static constraint violations (proposal 2)



(b) Dynamic constraint violations (proposal 3)

Fig. 1. Images of constraint violation acceptance by the proposed methods (the proposal 2 and 3).

Algorithm 1: Step 2: The solution update process using the Proposal 2

Input: Search point SM_ia, Violation tolerance rate $violation\_rate$
Output: Updated solution SM_ia

```
for n ← 0 to 3 do
    check ← 0;
    if n mod 2 = 0 then
        // Apply the Swap Product (SP)
        foreach target column do
            Replace products in SM_ia according to the SP operation;
    else
        // Apply the Swap Operator (SO)
        foreach pair of target columns do
            Exchange products in SM_ia according to the SO operation;
    // Check whether the constraint is satisfied or not
    foreach product p do
        if product p is not assigned to at least one column then
            check ← check + 1;
    // Apply the static acceptance scheme
    if check = 0 or U[0,1] < violation_rate then
        Calculate the objective function value according to Eqs. (1)–(5);
    else
        Set the objective function value to 0;
Select the best solution and update SM_ia;
```

Fig. 2. The proposed algorithm for static constraint violation acceptance.

With this mechanism, ISs can remain in the search space at a fixed rate from the beginning to the end of the search. This helps maintain solution diversity. Fig. 2 shows the algorithm of Step 2 in Section 3 under this scheme.

2) The dynamic acceptance scheme (Proposal 3): In this scheme, the acceptance rate changes according to the progress of the search. The dynamic acceptance rate, $V_{dynrate}$, is defined by the following equation:

$$V_{dynrate} = V_{max} - \frac{V_{max} - V_{min}}{N_{maxeval}} \times N_{eval} \tag{15}$$

where V_{max} is the maximum acceptance rate at the beginning of the search, V_{min} is the minimum acceptance rate at the end of the search, N_{eval} is the current number of objective function evaluations, and $N_{maxeval}$ is the maximum number of objective function evaluations.

With this mechanism, more ISs are accepted in the early stages of the search to maintain solution diversity, while stricter constraint satisfaction is gradually enforced in the later stages. Fig. 3 shows the algorithm of Step 2 in Section 3 under this scheme.

V. Simulations

A. Simulation Conditions

Simulations were conducted for the following five methods to verify the effectiveness of the proposed methods using the column capacities, demand data, and product price data from five actual vending machines:

1) The compared method 1: The conventional IDSMO based method without a penalty term in the objective function [20].

Algorithm 2: Step 2: The solution update process using the Proposal 3

Input: Search point SM_ia, Maximum constraint violation tolerance rate $v_{\max}$, Minimum constraint violation tolerance rate $v_{\min}$, Current number of objective function evaluations N_{eval}, Maximum number of objective function evaluations $N_{\text{eval}}^{\max}$, Violation tolerance rate $v_{\text{rate}}^{\text{dyn}}$

Output: Updated solution SM_ia

```
for n ← 0 to 3 do
    check ← 0;
    if n mod 2 = 0 then
        // Apply the Swap Product (SP)
        foreach target column do
            Replace products in SM_ia according to the SP operation;
    else
        // Apply the Swap Operator (SO)
        foreach pair of target columns do
            Exchange products in SM_ia according to the SO operation;
    // Check whether the constraint is satisfied or not
    foreach product p do
        if product p is not assigned to at least one column then
            check ← check + 1;
    // Apply the dynamic acceptance scheme
    v_rate^dyn ← v_max × (1 − N_eval / N_eval^max);
    if v_rate^dyn < v_min then
        v_rate^dyn ← v_min;
    if check = 0 or U[0,1] < v_rate^dyn then
        Calculate the objective function value according to Eqs. (1)–(5);
    else
        Set the objective function value to 0;
Select the best solution and update SM_ia;
```

Fig. 3. The proposed algorithm for dynamic constraint violation acceptance.

2) The compared method 2: The IDSMO based method with only a static penalty term in the objective function.

3) The compared method 3: The IDSMO based method with only a dynamic penalty term in the objective function

4) The proposed method 1: The proposed IDSMO based method with a dynamic penalty term and static acceptance of ISs during the update process (Proposal 1 + Proposal 2).

5) The proposed method 2: The proposed IDSMO based method with a dynamic penalty term and dynamic acceptance of ISs during the update process (Proposal 1 + Proposal 3).

The static penalty term utilized in the compared method 2 is defined as follows:

$$Penalty = r_0 * \left(\sum_p \sigma_p \right)^2 \tag{16}$$

The simulation parameters are as follows:

1) Common parameters of all methods: The number of objective function evaluations: 380000, the number of trials: 30, MN_g: 5, MLL: 1, MGL: 5, NSM: 20, C: 8, pr_LL: 0.5, pr_SO: 0.5, pr_SP: 0.5, α: 0.5

2) A common parameter excluding the compared method 1: r_0: 20000

3) A common parameter of the compared method 3, the proposed method 1, and the proposed method 2: e: 1.5

4) A parameter of the proposed method 1: $\text{violation}_{\text{rate}}$: 0.2

5) Parameters of the proposed method 2: $V_{\min}$: 0.0, $V_{\max}$: 0.1

A simulation environment on a PC (Intel®Core™i9-14900K), Linux Ubuntu Desktop 22.04 LTS using gcc (ver. 11. 4. 0) is utilized.

B. Simulation Results

Table 1 shows average, the minimum, the maximum, and standard deviation of the objective function values obtained over 30 trials by the compared method 1-3, and the proposed method 1 and 2. The bold numbers in the table show the best value in each evaluation metric. From Table 1, the proposed method 1 and 2 achieve higher-quality results than the compared methods in terms of average, the minimum, and the maximum objective function values. Furthermore, the proposed method 2 outperforms the proposed method 1 in terms of average, the minimum, and the maximum objective function values.

As observed in Table 1, since the proposed method 2 is superior to the proposed method 1, the statistical test is applied to the methods except the proposed method 1. Therefore, the table reports the p-value of the Friedman test for comparisons among the four methods excluding the proposed method 1. For selection of the statistical test, normality is assessed using the Anderson-Darling test and the D'Agostino and Pearson test. As a result, normality cannot be assumed. Since the initial values of the four methods are identical, the data are paired. Therefore, the Friedman test, a nonparametric test, was applied. Since the p-value of the Friedman test is below 0.05 significant level, it is verified that significant differences exist among the four methods. Table 2 shows corrected p-values obtained using the Wilcoxon signed-rank test with Holm correction as a post hoc test. The results confirmed that the proposed method 2 differed

TABLE I. COMPARISON OF AVERAGE, THE MINIMUM, THE MAXIMUM VALUES, STANDARD DEVIATIONS OF OBJECTIVE FUNCTION VALUES AMONG THE COMPARED METHOD 1-3, THE PROPOSED METHOD1, AND THE PROPOSED METHOD2, AND A P-VALUE OF THE FRIEDMAN TEST.

	The compared method 1 [20]	The compared method 2	The compared method 3	The proposed method 1	The proposed method 2
Ave	0. 62431	0. 651479	0. 654436	0. 66975	**0. 67070**
Max	0. 63914	0. 678944	0. 682278	0. 68550	**0. 68694**
Min	0. 61594	0. 626167	0. 628889	0. 65571	**0. 65904**
Std	**0. 00548**	0. 013887	0. 0131	0. 00660	0. 00741
p-value	6. 46E−15				6. 46E−15

TABLE II. CORRECTED P-VALUES BY THE WILCOXON TEST WITH HOLM CORRECTION AS A POST HOC TEST.

	The compared method 1 [20]	The compared method 1	The compared method 2	The proposed method 2
The compared method 1 [20]		S	S	S
The compared method 2	2. 15E−10		NS	S
The compared method 3	3. 30E−15	0. 3403		S
The proposed Method 2	2. 20E−16	2. 68E−06	4. 22E−07	

significantly from all the compared methods. These findings demonstrate the effectiveness of the proposed methods for the VMCOP.

VI. Conclusions

This paper proposes improved discrete spider monkey optimization using a dynamic penalty function with partial constraint violation acceptance for a vending machine column optimization problem considering sales and a replenishment cycle. The proposed methods are verified to obtain higher-quality solutions compared with the compared methods. In addition, with respect to the acceptance schemes for Infeasible solutions, the results are demonstrated that applying the dynamic acceptance scheme yields better solutions than the static acceptance scheme.

As future works, we will conduct comparative evaluations against recent state-of-the-art approaches in constrained combinatorial optimization and evolutionary computation to more clearly position the contribution of this study. In addition, new evolutionary computation techniques will be investigated to improve solution quality.

References

[1] Japan Vending System Machinery Manufacturers Association, Vending machine data, 2024. [Online]. Available: https://www.jvma.or.jp/information/information_3.html, (in Japanese)

[2] Climate Action Tracker, Japan: Policies and action, 2024. [Online]. Available: https://climateactiontracker.org/countries/japan/policies-action/

[3] The Coca-Cola Company, "Sustainability," 2025. [Online]. Available: https://www.coca-colacompany.com/about-us/sustainability

[4] R. Hosokawa, "Issues for Regulating Working Hours in the Post-Work Style Reform", Japan Labor Issues, Vol.9, No.51, pp.1-6, 2025. [Online]. Available: https://www.jil.go.jp/english/jli/documents/2025/051-01.pdf

[5] Y. Miyamoto, et al., "Algorithms for the Item Assortment Problem: An Application to Vending Machine Products," Japan J. of Industrial and Applied Mathematics, vol. 20, pp. 87–100, Feb. 2003 (in Japanese).

[6] T. Takeuchi, et al., "A Vending Machine Column Allocation Optimization Problem with Demand Following a Poisson Process," The journal of the Department of Economics of Gakushuin Univ., vol. 49, no. 1, pp. 47–52, Apr. 2012 (in Japanese).

[7] R. Masahira, Y. Fukuyama, et al., "Vending Machine Column Optimization by Tabu Search," Proc. of IEEJ Annual Conf., no. 3-032, Mar. 2022 (in Japanese).

[8] R. Masahira, Y. Fukuyama, et al., "Vending Machine Column Optimization by an Improved Monkey Algorithm with Multiple Populations," Proc. of Annual Conf. of Industry Applications Society of IEEJ, no. 5-19, Aug. 2022 (in Japanese).

[9] R. Masahira, Y. Fukuyama, et al., "Vending Machine Column Optimization by Multi-population evolutionary computation – Analysis of effective search by a statical multi-population strategy –," Proc. of IEEJ Annual Conf., no. 3-035, Mar. 2023 (in Japanese).

[10] J. P. Cohoon, et al., "Punctuated Equilibria: A Parallel Genetic Algorithm," Proc. of the 2nd Int. Conf. on Genetic Algorithms, pp. 148–154, Oct. 1987.

[11] D. Azuma, Y. Fukuyama, et al., "Distribution State Estimation Using Multiple Stages considering Asynchronous Measurement Data by Dependable Parallel Multi-Population Global-best Brain Storm Optimization with Differential Evolution strategies," Proc. of IEEJ Trans. PE, vol. 141, no. 6, pp. 426–439, June 2021 (in Japanese).

[12] X. Zhang, et al., "Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization," Int. J. of Electrical Power & Energy Systems, vol. 32, Issue. 5, pp. 351–357, June 2010.

[13] S. Z. Zhao, et al., "Dynamic Multi-Swarm Particle Swarm Optimizer with Subregional Harmony Search," Proc. of IEEE Congress on Evolutionary Computation, July 2010.

[14] M. F. Han, et al., "Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems," Applied Intelligence, vol. 39, pp. 41–56, Nov. 2013.

[15] J. C. Bansal, M. Clerc, et al., "Spider Monkey Optimization Algorithm for Numerical Optimization," Memetic Computing, vol.6, pp.31-47, Jan. 2014.

[16] G. Duarte, A. Lemonge, and L. Goliatt, "A dynamic migration policy to the Island Model," Proc. of IEEE CEC 2017, June 2017.

[17] M. M. Fouad, et al., "Dynamic Group-Based Cooperative Optimization Algorithm," IEEE Access, vol. 8, pp. 148378–148403, Aug. 2020.

[18] S. Koyama, Y. Fukuyama, et al., "A New Grouping Method for Improved Discrete Spider Monkey Optimization Using Dynamic Multi-population for Vending Machine Column Optimization," Proc. of IEEJ Electronics, Information and Systems Society Conference, GS11-1, Sep. 2023 (in Japanese).

[19] S. Koyama, Y. Fukuyama, et al., "Improved Discrete Spider Monkey Optimization for Vending Machine Column Optimization Considering Sales and a Replenishment Cycle," Proc. of IEEE 2024 Conference on AI, Science, Engineering and Technology (AIxSET), pp. 212–213, Sep. 2024.

[20] S. Koyama, Y. Fukuyama, et al., "Improved Discrete Spider Monkey Optimization Using Dynamic Multi-population for a Vending Machine Column Optimization Problem," Proc. of IEEJ Trans. on Electronics, Information and Systems, vol. 144, no. 12, pp.1217-1229, Dec. 2024 (in Japanese).

[21] Y. Hara, Y. Fukuyama, et al., "Vending Machine Column Optimization by an Improved Discrete Spider Monkey Optimization Using Dynamic Multi-population," Proc. of IEEJ Technical Meeting on Systems/Smart Facility, ST-22-026, SMF-22-033, Nov. 2022 (in Japanese).

[22] Y. Hara, Y. Fuluyama, et al., "Application of Multi-population Evolutionary Computation for Vending Machine Column Optimization ― Parameter Sensitivity Analysis of Improved Discrete Spider Monkey Optimization Using Dynamic Multi-populations ―," Proc. of IEEJ Annual Conf., no. 3-036, Mar. 2023 (in Japanese).

[23] H. Grzybowska, et al., "A simulation-optimisation genetic algorithm approach to product allocation in vending machine systems," Expert Systems with Applications, vol. 145, May 2020.

[24] R. Courant, "Variational Methods for the Solution of Problems of Equilibrium and Vibrations," Bull. Amer. Math. Soc., vol. 49, no. 1, pp. 1–23, Jan. 1943.

[25] J. A. Joines, and C. R. Houck, "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's," Proc. IEEE Int. Conf. Evolutionary Computation, pp. 579–584, Jun. 1994.

[26] C. A. Coello Coello, "Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art," Comput. Methods Appl. Mech. Engrg., vol. 191, no. 11–12, pp. 1245–1287, Jan. 2002.

[27] M. A. H. Akhand, et al., "Discrete Spider Monkey Optimization for Travelling Salesman Problem," Applied Soft Computing, vol. 86, pp. 1-13, Jan. 2020