A Comparison Analysis of Constraint-Handling Techniques on Rule Selection Problem in Credit Risk Assessment: An Industrial View

Yongfeng Gu*, Hao Ding, Kecai Gu, Xiaoguang Huang, and Hua Wu
Ant Group, Hangzhou, Zhejiang, China
{guyongfeng.gy, luoyan.dh, gukecai.gkc, addie.hxg, wuhua.wh}@antgroup.com
*corresponding author

***Abstract*—The rule selection problem, aiming at selecting a subset of rule candidates, is important in credit risk assessment. In reality, we hope these selected rules can identify as many high-risk users as possible while fulfilling the business constraint simultaneously. In such cases, the rule selection problem can be seen as a typical constrained combinatorial optimization problem and can be solved by evolutionary algorithms (EAs). However, how to deal with the constraint dramatically affects the performance of our EAs since the feasible and infeasible solutions both affect the convergence and direction of the EAs. This paper conducts a comparative study to explore the performance of six constraint-handling techniques on the rule selection problem. Experimental results indicate the importance of the way we handle the constraints: 1) ϵ-Constrained Method finds the most high-risk users and Constrained-Domination Principle obtains the most diversified solutions. 2) dominance-based techniques outperform penalty function-based techniques in our real-world applications.**

Keywords*—*constraint-handling; dominance-based techniques; penalty function-based techniques; credit risk assessment

I. Introduction

In financial institutions or banking systems, credit risk (e.g., default and fraud) is one of the most vital risks and may induce economic loss as well as reputational damage to the enterprise [1][2][3]. To avoid such losses, companies must conduct the *Credit Risk Assessment* before making any critical decisions. Generally speaking, credit risk assessment is a task to identify the high-risk users from the whole users based on their historical behaviors and features. In industry, we usually construct a rule-based model to process this task. We first select a set of rules from the rule candidates and then use the selected rules to predict whether or not the user is risky.

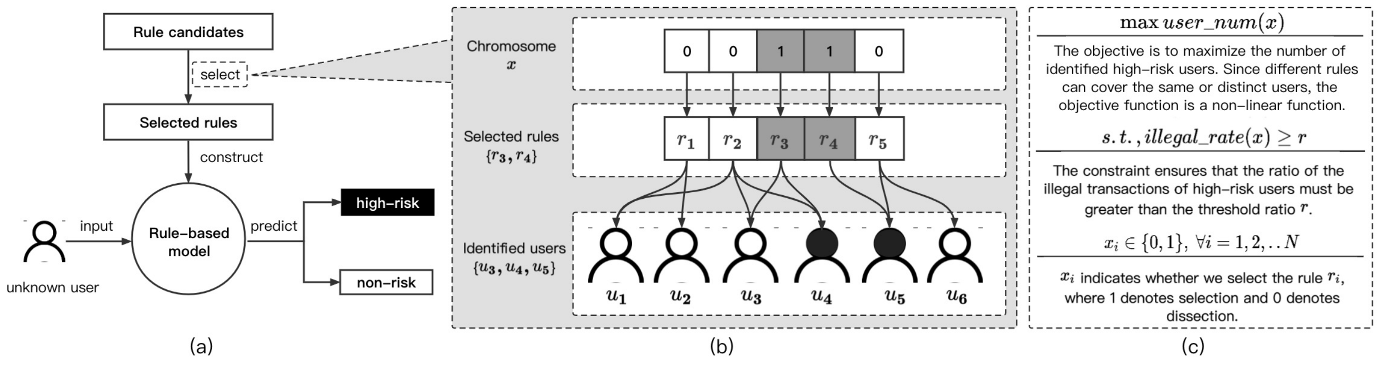
Consequently, choosing a suitable rules is important in credit risk assessment, which we call the *Rule Selection Problem*. Since our target is to identify the most high-risk users while fulfilling the business constraints. We consider the rule selection problem as a constrained combinatorial optimization problem and utilize the Evolutionary Algorithms (EAs) to solve it. Based on the Darwinian evolution theory, EAs can find a (near-)optimal solution by repeatedly updating the population with the crossover, mutation, and selection operations. Popular EAs includes GA (Genetic Algorithm [4]), DE (Differential Evolution [5]), SA (Simulated Annealing [6]), and PSO (Particle Swarm Optimization [7]). In industry, due to large-scale and non-convex features of the optimization problems, sometimes it is infeasible to directly use the exact algorithms to solve the problem. In such cases, we concern more about the feasibility than the optimality of solutions, GAs become a good choice and can often give an alternative near-optimal solution.

The traditional GAs are designed for unconstrained optimization problems and cannot be directly applied on Constrained Optimization Problem. To make GAs more general and flexible on constrained problems, many popular constraint-handling techniques are proposed [8][9][10][11][12]. The core problem of constraint-handling techniques is how to balance the trade-offs between optimality and feasibility during the evolution [13][14]. Since too strict constraint penalty results in the deterioration of objectives while too loose constraint penalty may lead to no feasible solutions.

For instances, Kramer et al. discussed the *Death Penalty Function* (DPF [15]), a straightforward method to give the maximum penalty to the infeasible solutions. Joines and Houch designed the *Dynamic Penalty Function* (DyPF [16]), a dynamic way to tune the punishment of infeasible solutions through iterations. Woldesenbet proposed the *Self-adaptive Penalty Functions* (SPF [17]), which flexibly tunes the punishment according to the feasibility information of the current populations. Deb *Constrained-domination Principle* gave the *Constrained-domination Principle* (CDP) to determine which solution is better when comparing infeasible and feasible solutions in the same generations. Runarsson and Yao proposed the *Stochastic Ranking* (SR [18]) to assist to rank the populations based on both constraint violation and objective values. Takahama and Sakai designed the *ϵ-Constrained Method* (ECM [19]), which induces a constraint relaxation parameter ϵ to divide the whole space into regions and then uses different domination principles to compare individuals.

DPF, DyPF, and SPF are penalty function-based techniques, which try to construct a penalty function consisting of the objective and constraint violation together, transforming the constrained problem into an unconstrained problem. CDP, SR, and ECM are dominance-based techniques, which design the comparison rules between two individuals in the population. These rules determines which individual is better in the selection process. However, the above techniques are widely compared in experimental environments (usually the artificial competition functions) and rarely analyzed in real-world datasets.

2693-9371/22/$31.00 ©2022 IEEE
DOI 10.1109/QRS-C57518.2022.00099



$\max user\_num(x)$

The objective is to maximize the number of identified high-risk users. Since different rules can cover the same or distinct users, the objective function is a non-linear function.

$s.t., illegal\_rate(x) \geq r$

The constraint ensures that the ratio of the illegal transactions of high-risk users must be greater than the threshold ratio r.

$x_i \in \{0,1\}, \forall i = 1, 2, ..N$

x_i indicates whether we select the rule r_i, where 1 denotes selection and 0 denotes dissection.

Figure 1. The details in solving the credit risk assessment in real-world applications. (a) depicts the overview of the credit risk assessment. (b) shows the chromosome encoding in the genetic algorithm. (c) presents the rule selection optimization problem in a mathematical format.

In this paper, to explore the impact of constraint-handling techniques on the credit risk assessment scenario, we conduct a comparative analysis of six techniques, namely the DPF, DyPF, SPF, CDP, SR, and ECM. We hope this preliminary study can assist to find a suitable constraint-handling technique for our real-world applications and further provides inspiration for how to use such techniques to improve the objectives.

Overall, this paper makes the following contributions:

- We conducted a comparative analysis of six typical constraint-handling techniques on a real-world application, which gave an industrial view on how to effectively handle the constraints.
- Experiments showed that ϵ-Constrained Method found the most high-risk users and Constrained-domination Principle achieved the most diversified solutions. Besides, dominance-based techniques outperforms the penalty function-based methods in our experiments.

This remaining paper is organized as follows, Section II presents the basic backgrounds about the credit risk assessment. Section III introduces the six constraint-handling techniques we used in our experiment. Experimental setup and results are discussed in Section IV. Finally, Section V concludes the paper and gives future work.

II. Backgrounds

This section presents the basic information about the credit risk assessment and then formulates the optimization problem we solve in this paper.

A. Credit Risk Assessment

Credit risk assessment can be seen as a classification problem that aims to categorize all users into high-risk and non-risk ones. In industry, we prefer to build a rule-based model (see Figure I (a)) to solve this problem due to its explainability and comprehensibility. To build such a model, a set of rules are selected from the rule candidates. Such rule candidates are exacted from the historical data and can assist to identify high-risk users in certain aspects. The rules perform in the "IF-THEN" format. An example rule is "IF Debt=True and Fraud=True THEN high-risk", which means a user will be predicted as a high-risk user if this user owns some debt and also has credit fraud records. After that, we use the selected rules to build the model, i.e., the rule subset, to predict whether an unknown user is a high-risk user or a non-risk one.

B. Rule Selection Problem

Rule selection problem is a core part of credit risk assessment, which studies how to select suitable rules from the rule candidates. In this paper, we concentrate on using evolutionary algorithm to solve the rule selection problem. In our study, we use the Elite Genetic Algorithms (EGA [20]) to find the optimal solutions. EGA records the global optimal solution in every iteration and updates it if there exists a better local optimal solution in the current generation.

The chromosome encoding is depicted in Figure I (b). Suppose that there are 5 rules $(r_1, r_2, ..., r_5)$ in the rule candidates, then we construct a 5-bit binary chromosome x to display the selection result. For instance, the chromosome $x = \{0, 0, 1, 1, 0\}$ indicates that we select rules $r3$ and $r4$ to build the rule-based model. Furthermore, the above selected rules identify three users u_3, u_4, u_5 , where u_4, u_5 are high-risk users and u_3 is a non-risk user. In this case, the selected rule can identify 2 high-risk users. The transactions evolved with a high-risk user are considered as the illegal transactions, otherwise, the transactions are called the legal transactions.

Rule select problem is considered as a optimization problem which aims to maximize the number of identified high-risk users ($user\_num(x)$) while the ratio of illegal transactions ($illegal\_rate(x)$) is greater than the threshold r. The inner implementation and structure of $user\_num(x)$ and $illegal\_rate(x)$ are high-order and time-consuming functions Here we omit their implementations due to the enterprise's secret and data security. We only consider these two functions as complex, non-convex, black-box functions. Let objective $f(x) = -user\_num(x)$ and $g(x) = -illegal\_rate(x) + r$, the rule selection problem can be formulated as,

$$\begin{aligned} &\min\ f(x) \\ &s.t.,\ g(x) \leq 0, \\ &\quad x_i \in \{0,1\},\ \forall i = 1, 2, ..., N \end{aligned} \tag{1}$$

III. Constraint-Handling Techniques

This section introduces six constraint-handling techniques discussed in this paper. Different techniques reveal various insights on dealing with constraints in the evolution process. DPF, DyPF, and SPF are penalty function-based methods, which aim to construct a fitness value by individuals' objective value and constraint violation. CDP, SR, and ECM are dominance-based methods, which prefer to design the comparison criteria among individuals.

A. Death Penalty Function (DPF)

Death Penalty Function [8] is the most naive method for handling the constraints and often serves as the baseline method. The core idea is to let the fitness value $\hat{f}(x)$ of the feasible individual be its objective value $f(x)$ and let that of the infeasible individual be infinity ($+\infty$).

$$\hat{f}(x) = \begin{cases} f(x), \text{ if } g(x) \leq 0 \\ +\infty, \text{ otherwise} \end{cases} \tag{2}$$

where $g(x) \leq 0$ denotes that x is feasible, and otherwise that x is infeasible. DPF assigns the wort fitness value ($+\infty$) for the infeasible solution, which ignores the potential information of the infeasible solution in the evolution.

B. Dynamic Penalty Function (DyPF)

Dynamic Penalty Function [16] designs the constraint violation $G(x)$ of each individual and lets the penalty factor $(C \cdot t)^{\alpha}$ vary from the iteration time. DyPF relaxes the punishment of infeasible individual in the early generations and then gradually increase the punishment of infeasible individuals in the subsequent generations.

$$\hat{f}(x) = f(x) + (C \cdot t)^{\alpha} \cdot G(x) \tag{3}$$

where t is the number of iterations. $C = 0.5$ and $\alpha = 1$ are the two hyper-parameters. The constraint violations function $G(x)$ is defined as $G(x) = \max\{0, g(x)\}$. The bigger the $G(x)$ is, the more severely the constraint will be violated.

C. Self-adaptive Penalty Function (SPF)

Self-adaptive Penalty Function [17] can dynamically tune the penalty factor according to the feasible ratio rf of the current population. Specifically, the fitness value consists of the distance $d(x)$ and penalty $p(x)$.

$$\hat{f}(x) = d(x) + p(x) \tag{4}$$

The calculation of $d(x)$ and $p(x)$ both rely on the ratio rf. Note that $rf = 0$ denotes all individuals are infeasible in the current population, while $rf \neq 0$ indicates some of the individuals are feasible.

$$d(x) = \begin{cases} G(x), \text{ if } rf = 0 \\ \sqrt{f(x)^2 + G(x)^2}, \text{ if } rf \neq 0 \end{cases} \tag{5}$$

$$p(x) = (1 - rf) \times X(x) + rf \times Y(x) \tag{6}$$

where

$$X(x) = \begin{cases} 0, \text{ if } rf = 0 \\ G(x), \text{ if } rf \neq 0 \end{cases} \tag{7}$$

$$Y(x) = \begin{cases} 0, \text{ if } x \text{ is feasible} \\ f(x), \text{ if } x \text{ is infeasible} \end{cases} \tag{8}$$

D. Constrained-Domination Principle (CDP)

Constrained-Domination principle [21] defines the comparing criteria between two individuals x_i and x_j. We consider that x_i dominates x_j, which are marked as $x_i \prec x_j$, in the following three conditions,

$$x_i \prec x_j \Leftrightarrow \begin{cases} \text{if } x_i, x_j \text{ are both feasible} \wedge f(x_i) < f(x_j), \\ \text{if } x_i, x_j \text{ are both infeasible} \wedge g(x_i) < g(x_j), \\ \text{if } x_i \text{ is feasible and } x_j \text{ is infeasible} \end{cases} \tag{9}$$

CDP is a standard constraint-handling technique, which is widely employed in several genetic algorithm toolboxes, such as GeatPy[1]. Compare with the penalty function-based techniques, CDP performs more flexibly in the comparison between feasible and infeasible individuals.

E. Stochastic Ranking (SR)

Stochastic Ranking [18] combines the Bubble Sort with its comparison criteria to rank individuals. Specifically, when comparing two individuals we generate a random number $u \in [0, 1]$. If two individuals are both feasible or the random number u is less than the given threshold pf, the domination relationship relies on their objectives, otherwise relies on their constraint violations. We set $pf = 0.475$ in this paper. The details of SR are listed in the following pseudo-code,

Algorithm 1: Stochastic Ranking

```
Input: Population $P = \{x_1, x_2, ..., x_{NP}\}$, probability $pf$,
       objective f, constraint violation G
Output: Sorted Population $P$
1 for i in range(NP) do
2     for j in range(NP-i-1) do
3         u = randn(0, 1);
4         if ($x_j$ and $x_{j+1}$ are feasible) or ($u \leq pf$) then
5             if $f(x_j) > f(x_{j+1})$ then
6                 swap($x_j, x_{j+1}$);
7         else
8             if $G(x_j) > G(x_{j+1})$ then
9                 swap($x_j, x_{j+1}$);
```

F. ϵ-constrained (ECM)

ϵ-Constrained Method [19] induces the violation tolerance ϵ when comparing two individuals, where ϵ is varying from iterations in Equation (10).

$$\epsilon(t) = \begin{cases} \epsilon(0)\ (1 - t/T_c)^{cp}, \ 0 < t < T_c \\ 0, \ t \geq T_c \end{cases} \tag{10}$$

[1]Geatpy, http://geatpy.com/index.php/home/

TABLE I
BASIC INFORMATION OF THE USED DATASETS

Dataset	# Rule candidates	# Total users
A	1,001	100,000
B	1,001	200,000
C	1,001	500,000
D	1,001	1,000,000
E	1,001	2,000,000

where $\epsilon(t)$ denotes the value of ϵ in the t-th iteration. We set $\epsilon(0) = G(x^{\theta})$. x^{θ} is the θ-th individual according to constraints violations in ascending order. We set $\theta = 0.05NP$, $T_c = 0.1T_{max}$, anc $cp = 2$ in this paper. NP is the population size and T_{max} is the maximum iteration times.

After that, the ϵ-Constrained Method compares two individuals by the following criteria,

$$x_i \prec x_j \Leftrightarrow \begin{cases} f(x_i) < f(x_j), & \text{if } g(x_i), g(x_j) < \epsilon \\ f(x_i) < f(x_j), & \text{if } g(x_i) = g(x_j) \\ g(x_i) < g(x_j), & \text{otherwise} \end{cases} \tag{11}$$

IV. EXPERIMENTS

This sections list the experimental setup and the preliminary results on the real-world applications.

A. Experimental Setup

1) Dataset: We collect four datasets with different scales sampled from the real-world transactions in our company (see Table I). Column '# Rule candidates' presents the number of rule candidates whom we select from. Column '# Total users' presents the number of total users (high-risk users and non-risk users). More number of users indicates more complex mapping relationships, making the optimization problem more sophisticated Note that the data set is only used for academic research, it does not represent any real business situation.

2) Implementations: For the Elite Genetic Algorithm, we set the population size to 100. The ratio of crossover and mutation operations are 0.9 and 1/100, respectively. All experiments are developed on the basis of an Evolutionary Algorithm toolkit GeatPy.

B. Experimental Results

To explore the impact of constraint-handling techniques on the credit risk assessment task, we combine the above six techniques with the standard Genetic Algorithms to solve the rule selection problem. We use the identified high-risk users, i.e., the $user\_num(x)$ in Section II, as the objective and try to answer the following two research questions.

RQ-1: Which technique identifies the most high-risk users?

We run each technique 20 times and record their mean value, standard deviation, the best and worst values. The number of identified high-risk users on six constraint-handling techniques are listed in Table II.

As we can see, ECM performs the best and achieves the highest mean value in datasets B and E with relatively small standard deviations, which indicates that ECM is effective and robust. In addition, ECM can find the highest high-risk users in datasets A, B, and E. CDP and SR perform well, obtaining the highest mean value in datasets C and D, respectively. SR can find the highest high-risk users in datasets B and D. CDP can find the highest high-risk users in dataset C.

As for the penalty function-based method, DPF finds the least high-risk users and also has the most unstable performance, obtaining the largest standard deviations in datasets A, C, D, and E. SPF performs well in dataset A, which has relatively good results compared to DPF and DyPF.

Overall, the dominance-based methods (CDP, SR, and ECM) perform better than the penalty function-based methods (DPF, DyPf, and SPF). This result reveals the fact that we should concern more about the feasible and infeasible solutions during evolution and adjust the comparison criteria according to the current population automatically.

Answer to RQ-1: ECM performs the best and identifies the most high-risk users. Dominance-based methods achieve better results than penalty function-based methods.

RQ-2: Which technique finds more diversified solutions?

Besides the number of identified users, we also concern about the diversity of solutions provided by different techniques. In reality, we not only expect to identify more high-risk users but also hope the algorithms can provide us with more diversified solutions, that is, the rule combinations. Thus, we can select the most suitable one from the outputted solutions based on the factual business requirements. In other words, diversified solutions make it more flexible for decision-makers to select rules.

For each constraint-handling technique, we record the number of distinct output solutions which identify the most high-risk users. Higher numbers mean more choices to select rules. The results are listed in Table III, it is obvious that CDP has more diversified solutions than others, which obtain 62, 57, and 13 solutions in datasets A, B, and E. Dominance-based methods can obtain more diversified solutions. On contrary, DPF, DyPF, and SPF have the least diversified solutions, which indicates we can have only fewer choices to select rules. The main reason may be the penalty function fail to make use of the potential information of infeasible individuals and only focus on the feasible individuals. However, some 'good' bits or patterns in the infeasible individuals can also be used in evolution.

Answer to RQ-2: CDP obtains the most diverse solutions to achieve the same optimal objective. Dominance-based methods can provide more diversified solutions than penalty function-based methods.

V. CONCLUSION AND FUTURE WORK

In this paper, we consider the rule selection problem in credit risk assessment as a constrained optimization problem.

TABLE II

NUMBER OF IDENTIFIED HIGH-RISK USERS OF SIX CONSTRAINT-HANDLING TECHNIQUES ON FIVE REAL-WORLD DATASETS. THE 'MEAN', 'STD', 'BEST', AND 'WORST' DENOTES THE MEAN VALUE, STANDARD DEVIATION, BEST OBJECTIVE, AND WORST OBJECTIVES IN 20 RUNS.

Dataset	DPF				DyPF				SPF				CDP				SR				ECM			
	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst
A	55,291.61	12.65	55,310	55,269	56,334.48	11.56	56,362	56,311	**56,386.47**	10.18	56,400	56,371	56,379.49	12.29	56,400	56,355	56,383.90	5.14	56,397	56,371	56,382.01	10.53	56,403	56,351
B	93,672.99	24.67	93,721	93,635	93,855.39	33.59	93,923	93,796	93,859.04	36.70	93,920	93,785	93,930.20	23.92	93,986	93,874	93,774.30	86.48	94,043	93,180	**93,978.27**	31.30	94,043	93,897
C	232,089.31	70.36	232,224	231,979	227,974.68	58.76	228,133	227,889	233,024.94	66.03	233,120	232,902	**233,096.12**	24.10	233,136	233,051	233,093.98	15.39	233,127	233,071	232,791.64	47.70	232,924	232,679
D	447,055.33	75.19	447,420	446,993	447,072.04	54.86	447,151	446,983	446,870.27	40.84	446,951	446,794	445,848.29	40.06	445,912	445,783	**447,772.57**	37.28	447,823	447,699	447,721.77	34.41	447,821	447,725
E	758,089.69	165.33	758,421	757,782	762,101.69	134.43	762,322	761,893	762,184.26	100.57	763,307	762,942	763,310.69	83.26	763,415	763,122	763,280.48	86.03	763,412	763,099	**763,319.63**	48.53	763,418	763,235

TABLE III

NUMBER OF DIVERSITY SOLUTIONS OBTAINED BY EACH CONSTRAINT-HANDLING TECHNIQUE

Dataset	DPF	DyPF	SPF	CDP	SR	ECM
A	1	1	6	**62**	1	61
B	36	1	1	**57**	1	54
C	1	1	1	32	1	**33**
D	1	1	1	19	**51**	2
E	1	1	1	**13**	2	11

To solve it, we try six specific constraint-handling techniques with the standard Genetic Algorithm. Our work is based on real-world datasets and can bridge the gap between academics and industry. Experiments show that different constraint-handling techniques affect the results to varying degrees. ϵ-Constrained Method identifies the most high-risk users and Constrained-Domination Principle obtains the most diversified solutions. Dominance-based techniques outperform the penalty function-based methods in our real-world situations. Since this is preliminary work about the rule selection problem, promising future work involves adding more constraint-handling techniques and using more evolutionary algorithms.

ACKNOWLEDGMENT

This work was supported by Ant Group through Ant Research Program. We thank our participants Jun Zhou, Yulin Kang, and Jinghua Feng for their contributions to this paper.

REFERENCES

[1] Makram Soui, Ines Gasmi, Salima Smiti, and Khaled Ghédira. Rule-based credit risk assessment model using multi-objective evolutionary algorithms. *Expert Syst. Appl.*, 126:144–157, 2019.

[2] Kevin Buehler, Andrew Freeman, , and Ron Hulme. The new arsenal of risk management. *Harvard Business Review*, 86(9):92–100, 2008.

[3] Yongfeng Gu, Hao Ding, Kecai Gu, Runsheng Gan, Xiaoguang Huang, Yanming Fang, Zhigang Hua, Hua Wu, Jifeng Xuan, and Jun Zhou. Heuristic strategies for solving the combinatorial optimization problem in real-world credit risk assessment. In *GECCO '22: Genetic and Evolutionary Computation Conference, Companion Volume, 2022*, pages 715–718. ACM, 2022.

[4] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091 – 8126, 2021.

[5] Manolis Georgioudakis and Vagelis Plevris. A comparative study of differential evolution variants in constrained structural optimization. In *Frontiers in Built Environment*, 2020.

[6] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson. The theory and practice of simulated annealing. In *Handbook of Metaheuristics*, 2003.

[7] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22:387–408, 2018.

[8] Oliver Kramer. A review of constraint-handling techniques for evolution strategies. *Appl. Comput. Intell. Soft Comput.*, 2010:185063:1–185063:11, 2010.

[9] Cheng gang Cui, Yan jun Li, and Tie-Jun Wu. A relative feasibility degree based approach for constrained optimization problems. *Journal of Zhejiang University SCIENCE C*, 11:249–260, 2009.

[10] Yong Wang, Bing chuan Wang, Han-Xiong Li, and Gary G. Yen. Incorporating objective function information into the feasibility rule for constrained evolutionary optimization. *IEEE Transactions on Cybernetics*, 46:2938–2952, 2016.

[11] Carlos A. Coello Coello. Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization*, 32:275 – 308, 2000.

[12] Hemant Kumar Singh, Tapabrata Ray, and Warren F. Smith. Performance of infeasibility empowered memetic algorithm for cec 2010 constrained optimization problems. *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[13] Josiel Neumann Kuk, Richard Aderbal Gonçalves, Lucas Marcondes Pavelski, Sandra Mara Guse Scós Venske, Carolina Paula de Almeida, and Aurora Trinidad Ramirez Pozo. An empirical analysis of constraint handling on evolutionary multi-objective algorithms for the environmental/economic load dispatch problem. *Expert Syst. Appl.*, 165:113774, 2021.

[14] Jia-Peng Li, Yong Wang, Shengxiang Yang, and Zixing Cai. A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization. In *IEEE Congress on Evolutionary Computation, CEC 2016*, pages 4175–4182. IEEE, 2016.

[15] Oliver Kramer and Hans-Paul Schwefel. On three new approaches to handle constraints within evolution strategies. *Natural Computing*, 5:363–385, 2006.

[16] Jeffrey A. Joines and Christopher R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 579–584. IEEE, 1994.

[17] Yonas G. Woldesenbet, Gary G. Yen, and Biruk G. Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Trans. Evol. Comput.*, 13(3):514–525, 2009.

[18] Thomas Philip Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evol. Comput.*, 4(3):284–294, 2000.

[19] Tetsuyuki Takahama and Setsuko Sakai. Constrained optimization by the ϵ constrained differential evolution with gradient-based mutation and feasible elites. In *IEEE International Conference on Evolutionary Computation, CEC 2006*, pages 1–8. IEEE, 2006.

[20] H. Adeli and K. C. Sarma. *Evolutionary Computing and the Genetic Algorithm.* Cost Optimization of Structures: Fuzzy Logic, Genetic Algorithms, and Parallel Computing, 2006.

[21] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2):311–338, 2000.