Analysis of the Strategies in Heuristic Techniques for Solving Constrained Optimisation Problems

Ahmad Rezaee Jordehi [1] , Nouradin Hashemi [1] , Hamid Nilsaz Dezfouli[2]

[1]Department of Electrical Engineering, University Putra Malaysia
[2]Department of mathematics, Mahshahr Branch, Islamic Azad University, Mahshahr, Iran

Abstract: Many real-world optimisation problems are constrained. Solving such problems is somehow challenging for optimisation techniques. Among various optimisation techniques, heuristics have demonstrated more efficiency in tackling constrained problems. In this paper, different strategies in heuristics for handling constrained optimisation problems are analyzed in details and the advantages and disadvantages of each strategy is discussed. The paper can provide a broad view to researchers in related area and help them to recognize the best constraint handling strategy for their certain problem.
[Jordehi AR, Hashemi N, Nilsaz Dezfouli H. **Analysis of the Strategies in Heuristic Techniques for Solving Constrained Optimisation Problems.** *J Am Sci* 2012;8(10):345-350]. (ISSN: 1545-1003).
http://www.jofamericanscience.org. 51

Keywords: optimisation; heuristics; constraint handling; classic optimisation techniques.

1. Introduction

There are so many optimisation problems in various areas of science and engineering. For solving them, there exist twofold approaches; classical approaches and heuristic approaches. Classical approaches are not efficient enough in solving optimisation problems. Since they suffer from curse of dimensionality and also require preconditions such as continuity and differentiability of objective function that usually are not met.

Heuristic approaches which are usually bio-inspired include a lot of approaches such as genetic algorithms, evolution strategies, differential evolution, ant colony algorithm and so on. Heuristics do not expose most of the drawbacks of classical and technical approaches. Therefore, they are very commonplace in solving optimisation problems.

However, many real-world optimisation problems are constrained while typical heuristic variants are supposed to handle unconstrained problems. Therefore, for enhancing them to be able to handle constrained problems, appropriate modifications should be applied to them. In this paper existing strategies in heuristics for tackling constraints are analysed deeply. Moreover, in order to provide a better understanding for reader, the capability of classic optimisation techniques in constraint handling is discussed concisely. The paper is organised as follows; in section 2, general framework of a constrained optimisation problem is presented and also classic optimisation techniques are investigated from constraint-handling capability point of view. In section 3, strategies in heuristic techniques for dealing with constrained problems are analysed. Finally, drawing conclusions and proposing some directions for future research is implemented in section 4.

2. Generalities of Constrained Optimisation

Most of real-world optimization problems are constrained. General form of a constrained optimization problem (COP) is:

Minimize $f(X)$
Subject to (1)

$$h_i(X) = 0, \quad i = 1,2,\dots,m$$

$$g_j(X) \le 0, \quad j = 1,2,\dots,p$$

Where m is the number of equality constraints and p is the number of inequality constraints. Each point X which satisfies all equality and inequality constraints is called a feasible point.

However, most of existing optimization techniques are merely devised for unconstrained problems. Thus, efficient constraint handling mechanisms should be incorporated to make them capable of dealing with constrained optimization problems. In this section, firstly, most common constraint handling strategies in classic optimization are explained, then constraint handling strategies existent in EA literature will be analysed and eventually, constraint handling strategies adopted in PSO will be explored in depth.

2.1 Constraint Handling Mechanisms in Classic Optimization Methods

2.1.1 Lagrangian Algorithm

In this method, constrained problem in (1), is converted to an unconstrained problem via creating lagrangian function as follows (Chong and Zak, 2008).

$$l(X,\lambda) = f(X) + \boldsymbol{\lambda}^T \boldsymbol{h}(X) + \boldsymbol{\mu}^T \boldsymbol{g}(X) \quad (2)$$

Where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_m]$ and $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_p]$ are its multipliers. $\boldsymbol{h}(X) = [h_1(X), h_2(X), \dots, h_m(X)]$,

and $\boldsymbol{g}(X) = [g_1(X), g_2(X), \dots, g_m(X)]$. By applying first order necessity conditions and second order sufficiency condition, the value of X^* (minimizer), $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are calculated holding following two constraints.

$$\begin{cases} \boldsymbol{\mu} \geq \mathbf{0} \\ \boldsymbol{\mu}^T \boldsymbol{g}(X^*) = 0 \end{cases} \quad (3)$$

2.1.2 Projection Technique

In this method, each infeasible point is transferred to its closest feasible point in search space. For example in feasible direction methods that use $X(t+1) = X(t) + \alpha_t \boldsymbol{d}_t$ to find optimal solution (α_t is step size and $\boldsymbol{d}_t$ is feasible direction), projection method uses $X(t+1) = \prod[X(t) + \alpha_t \boldsymbol{d}_t]$ instead, where $\prod(X)$ is the closest point in feasible region to X. Most common form of projection method is the gradient projection method, in which $X(t+1) = \prod\big[X(t) - \alpha_t \nabla f\big(X(t)\big)\big]$ (Chong and Zak, 2008).

2.1.3 Sequential Unconstrained Technique

This technique solves a non-linear optimization problem by transforming it into a sequence of optimization problems so that each sub-problem has a quadratic objective function with linear constraints (Chong and Zak, 2008).

2.1.4 Overall View on Constraint Handling with Classic Methods

Overally, classic optimization techniques are not appropriate enough for solving constrained problems. The main reasons are:

- They need pre-requisites such as differentiability and continuity of objective functions and constraints that are not met in most of problems especially real-world problems.
- They give out local optimum not global one.
- They are complex and inflexible.

3. Constraint-Handling Strategies in Heuristics

3.1 Penalty Functions

This technique transforms a constrained optimization problem into an unconstrained one by adding a certain value to the objective function based on the amount of constraint violation or the number of constraint violations in a certain individual. Mainly due to its simplicity and ease of implementation, it is the most commonly used technique for dealing with COP's. There are two types of penalty functions; exterior and interior. In exterior paradigm, the optimization is started with an infeasible individual; afterwards individuals are attracted to feasible regions of search space due to the effect caused by penalties. On the other hand, in interior models, a penalty function is defined whose values at points away from constraint boundaries are small and tend to infinity when the constraint boundaries are approached. So, if the search starts with a feasible individual, the subsequent generated individuals all lie within feasible region since the constraint boundaries act as barriers during optimization process (Rao, 1996). Since finding an initial feasible individual is itself a NP-hard problem, the exterior penalty approach is commonly used (Back, Fogel and Michalewics, 1997).

The general form of a penalised (or expanded) objective function is:

$$\varphi(X) = f(X) + \left[\sum_{i=1}^{i=m} K_i H_i + \sum_{j=1}^{j=p} C_j G_j\right] \quad (4)$$

Where $\varphi(X)$ is called expanded objective function, $G_j = \max\{0,\ g_j(X)\}^{\beta}$ and $H_i = |h_i(X)|^{\gamma}$. C_i's and K_i are called penalty factors. β and γ are commonly set as 1 or 2. The right hand term, which is in bracket, called penalty function.

The value of penalty factors is crucial and strongly affects the performance of penalty function approach. If they are chosen too small, too much search effort is put on infeasible regions of search space while feasible regions are not explored efficiently and even the algorithm is likely to not converge to a feasible solution. On the other hand, if they are too large, infeasible regions may not be explored enough and the valuable information existent in infeasible regions may not be extracted.

It should be noted that, normally equality constraints are transformed into inequality ones. That is $h_i(X) = 0$ is transformed to $|h_i(X)| \leq \varepsilon$ where ε is a tolerance allowed.

3.1.1 Static Penalty Functions

In this category, penalty functions do not change during the evolutionary process. In (Homaifar, Lai, and Qi, 1994) a static penalty function is introduced wherein several levels of violation are defined by user and a penalty factor is chosen for each level so that the penalty factor is increased when higher levels of violation are approached. The expanded objective function is:

$$\varphi(X) = f(X) + \sum_{j=1}^{j=p} C_{kj} G_j \quad (5)$$

Where $G_j = \max\{0, g_j(X)\}^{\beta}$ and k=1,2,…,l where l is the number of levels of violation. Here, all equality constraints have been transformed to inequality ones.

In (Morales and Quezada, 1998) another static penalty function with following expanded objective function has been introduced:

$$\varphi(X) = \begin{cases} f(X) & \text{if } X \text{ is feasible} \\ K - \sum_{i=1}^{i=s} \left(\frac{K}{p}\right) & \text{otherwise} \end{cases} \quad (6)$$

Where s represents the number of constraints satisfied, p is the number of constraints (note that equality constrains are transformed to inequality ones) and K is a large constant. Indeed, in this mechanism all infeasible individuals are penalised according to the number of constraints they have satisfied regardless of their distance to feasible region.

In (Hoffmeister and Sprave, 1996) the following expanded objective function has been utilised wherein like most other penalty functions, infeasible individuals are penalised according to their distance to feasible region.

$$\varphi(X) = f(X) + \sqrt{\sum_{j=1}^{p} H\left(-g_j(X)\right) g_j(X)^2} \quad (7)$$

Where $H(X) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$

3.1.1.1 Overall View on Static Penalty Functions

Overally speaking about the static penalty functions, although they can successfully solve some constrained problems, but their main disadvantage is that they commonly contain some parameters to be tuned by user and their performance is too sensitive to these parameters.

3.1.2 Dynamic Penalty Functions

Dynamic penalty functions are the functions which vary according to the iteration number. In (Jones and Houck, 1994) following dynamic expanded objective function is proposed in which the penalty increases with increasing iteration number.

$$\varphi(X) = f(X) + (C.t)^{\alpha}.SVC(\beta, X) \quad (8)$$

Where:

$$SVC(\beta, X) = \sum_{i=1}^{i=m} H_i(X) + \sum_{j=1}^{p} \left(G_j(X)\right)^{\beta} \quad (9)$$

$$H_i(X) = \begin{cases} 0 & \text{if } \varepsilon \le H_i(X) \le \varepsilon \\ |H_i(X)| & \text{otherwise} \end{cases} \quad i = 1,2,\dots,m \quad (10)$$

$$G_j(X) = \max\{0, g_j(X)\} \quad j = 1,2,\dots,p \quad (11)$$

3.1.2.1 Overall View on Dynamic Penalty Functions

Although some researchers claim that dynamic penalty functions outperform static ones, but the process of setting their parameters is as difficult as that of static penalty functions.

3.1.3 Adaptive Penalty Functions

These penalty functions penalise infeasible individuals according to the feedback taken from search process. For example in (Alouane and Bean, 1997).

$$\varphi(X) = f(X) + \lambda(t)\left[\sum_{i=1}^{i=m} |H_i(X)| + \sum_{j=1}^{p} \left(g_j(X)\right)^2\right] \quad (12)$$

Where $\lambda(t)$ is updated via:

$$\lambda(t+1) = \begin{cases} \frac{\lambda(t)}{\beta_1} & \text{If case I occurs} \\ \beta_2.\lambda(t) & \text{If case II occurs} \\ \lambda(t) & \text{Otherwise} \end{cases} \quad (13)$$

Where cases I and II represent situations in which the best individual in last K iterations was always feasible (in case I) or was never feasible (in case II). $\beta_1 > \beta_2 > 1$. Indeed, (13) implies that if the algorithm is searching mainly in feasible regions, the penalty is decreased to increase exploring infeasible regions and conversely, if the search is being conducted insufficiently in feasible regions, the penalty is reduced to attract individuals more toward feasible regions. The main drawback of this adaptive method is the difficulty of setting K, β_1 and β_2 .

3.1.4 Annealing-Based Penalty Functions

In (Scalak, et. al) a multiplicative penalty function inspired from simulated annealing is introduced as:

$$\varphi(X) = A.f(X) \quad (14)$$

In this equation A is given by:

$$A = e^{-M/T} \quad (15)$$

Where M represents the amount of constraint violation and T is temperature that decreases during the run according to:

$$T = \frac{1}{\sqrt{t}} \qquad (16)$$

That is, with increasing iteration number, the temperature decreases that leads to increase in penalty, so the search is propelled toward feasible regions of search space.

3.1.5 Co-Evolutionary-Based Penalty Functions

In co-evolutionary method, the population is partitioned into two subpopulation with sizes M_1 and M_2, where the function of first population is to evolve solutions and second population evolves penalty factors W_1 and W_1. Here also the main concern is the setting of M_1 and M_2. The expanded objective function is as follows (Coello, 2000).

$$\varphi(X) = f(X) + W_1 . \sum_{j=1}^{j=p} C_{kj}\, G_j + W_2 . N \qquad (17)$$

Where N represents the number of constraint violations and

$$G_j = \max\{0, g_j(X)\}^{\beta}$$

That is, both the information about amount of constraint violation and number of violations are considered in computation of penalty function.

3.1.6 Death Penalty

This method can be considered as the static penalty function approach whose penalty factors are all infinity, so all the infeasible solutions are rejected and just feasible regions are explored. The search is started with feasible solutions and also continues with feasible solutions.

The main advantage of this method is the lack of the necessity to set penalty factors and its main drawbacks are the followings:

- Generating initial feasible solutions may be difficult or so time-consuming.
- Since infeasible regions of search space are not explored, the valuable information present in those regions cannot be exploited, even in cases where the algorithm cannot reach global optimum, especially in problems which search space contains many scattered disjoined regions.

3.2 Repair Approaches

In this approach, at each iteration the feasible individuals are converted to feasible ones in a repair process (Michalewics and Nazhiyath, 1995), (Xiao, Michalewics and Trojanowski, 1997). The repaired individual can either be used just for evaluation or may replace the original individual. This approach is suitable for problems wherein repairing infeasible individuals can be conducted easily. Furthermore, each repair process merely behaves well in certain environments.

3.3 Separatist Approaches

These approaches do not combine objective function and constraints, but handle them separately.

3.3.1 MO-Based Separatist Approaches

In this approach, the minimization of the single-objective constrained problem in (1) is translated to the minimization of the following unconstrained multi-objective function (Surry, Radcliffe and Boyd, 1995).

$$F(X) = \left(f(X), f_1(X), f_2(X), \dots, f_{m+p}(X)\right) \qquad (18)$$

Where $f_i(X)$ represents the violation amount related to ith constraint.

Then, this MO problem can be solved by various MO techniques present in literature.

3.3.2 Co-Evolutionary-Based Separatist Approaches

In this approach, the population is partitioned into two sub-populations. The first sub-population contains constraints while the second one contains potential solutions of problem. Using an analogy with predator-prey model, the selection pressure on members of one sub-population relies on the members of the other sub-population (Paredis, 1994). The outcomes of this approach are promising and efficient mainly because there is no need to check all constraints at each iteration.

3.3.3 Separatists Based on Deb's Comparison Rules

This is the most commonly-used constraint-handling mechanism in EA's. In this approach, the following rules are applied for pairwise comparison of individuals (Deb, 2000):

1. A feasible individual always wins an infeasible one.
2. Between two feasible individuals, the one with better fitness value wins.
3. Between two infeasible individuals, the one with less amount of constraint violation wins. They are evaluated via:

$$\varphi(X) = f_{worst} + \sum_{i=1}^{i=p} \mathrm{Max}\left(0,\ g_i(X)\right) \qquad (19)$$

Where all constraints have been transformed to inequality constraints and f_{worst} represents the fitness of the worst feasible individual. In computing sum of

the constraints, each constraint is normalised with respect to the worst amount of its violation.

Indeed, Deb's approach represents a lexicographic ordering wherein first, the sum of constraint violation in two individuals is compared. If the violation of one of them is lower than the other one, it wins. Otherwise, if their violations are equal, their objective function is compared to determine the winner individual.

Due to the following reasons, Deb's approach is considered as the best and most-commonly used approach for constraint handling.

- It is so simple.
- It is parameter-free, unlike most other constraint handling approaches wherein the main concern is to set their parameters.
- Unlike approaches like death penalty approach, Deb's approach explores infeasible regions of search space. Therefore valuable information extracted in infeasible regions can be extracted.

However, in this approach, there is an overpressure for selecting feasible individuals, so the exploration of infeasible regions may not be implemented sufficiently. As a result, premature convergence issue is more acute here. So, in this approach, the incorporation of efficient diversity-enhancement strategies is of higher importance with respect to other constraint-handling mechanisms.

3.3.4 Stochastic Ranking

In this approach which does not require any penalty factor, the balance between objective and penalty function is attained through a stochastic ranking procedure based on the bubble-sort algorithm. For any pair of two adjacent individuals, if both individuals are feasible, the probability of comparing them according to the objective function is unity, otherwise the probability is set to P_f . λ individuals are ranked by comparing adjacent individuals in at least λ sweeps. The procedure is halted when no change in the rank happens within a complete sweep. The experiments reveal that $0.4 < P_f < 0.5$ works well is most problems (Runarsson and Yao, 2000). The massive advantage of this approach is the lack of necessity to set penalty factors.

3.4 Hybrid Approaches

In some cases, two different EA's or an EA with another optimization technique are hybridised to solve COP's more efficiently (Le, 1995), (Adeli and Cheng, 1994). For example in (Le, 1995) EA is hybridised with fuzzy logic to handle constraints. This technique puts the constraints in the form of $g_i(X) \le b_i$ and then fuzzifies them via:

$$\mu C_i(X) = \mu_\sigma(b_i, \epsilon_i).\, g_i(X) \quad i = 1,2,\dots p \qquad (20)$$

Where ϵ_i represents the allowable tolerance of violation in constraint i and:

$$\mu_\sigma(a,s)(X) = \begin{cases} 1 & for\ X \le a \\ \dfrac{exp\left(-p\left(\dfrac{X-a}{s}\right)^2\right) - exp(-p)}{1 - exp(-p)} & a < X \le a+s \\ 0 & for\ X \le a \end{cases} \qquad (21)$$

The new objective function is defined by:

$$\varphi(X) = f(X).\min\left(\mu C_1(X), \mu C_2(X), \dots, \mu C_p(X)\right) \qquad (22)$$

Like most other constraint handling mechanisms, the main drawback in this method is the necessity to set parameters b_i and ϵ_i.

4. Conclusions and Future Research Directions

In this paper, different strategies in heuristics for handling constrained problems are analysed deeply. Based on conducted analysis, the followings are proposed as some directions for future research on this area. Moreover, despite all the research effort has been put on and all the existent achievements, there is still room for improvement especially in the following terms.

- Devising diversity-enhancement mechanisms specific to constrained environments.
- Devising constraint-handling mechanisms for dynamic and/or multi-objective problems.
- Devising mechanisms for handling dynamic constraints (that is when the constraints vary over time)
- Dividing the population into some subpopulations and using a different constraint handling mechanism for each subpopulation has been scarcely used while it may be so promising.
- Using different constraint handling mechanisms in different stages of optimisation process (it has not been implemented yet).

Corresponding Author:
Nouradin Hashemi
Department of Electrical Engineering
University Putra Malaysia
UPM Serdang, Selangor 43400, Malaysia
E-mail: nourutm@yahoo.com

References

1. Chong EKP, Zak SH. An introduction to optimization. John Wiley Sons, New Jersey, 2008.
2. Rao SS. Engineering optimization. John Wiley Sons, New Jersey, 1996
3. Back T, Fogel DB, Michalewics Z. Handbook of evolutionary computation. Oxford University Press, 1997.
4. Homaifar A. Lai SHY, Qi X. Constrained optimization via genetic algorithms. Simulation 1994;62(4):242-54.
5. Morales AK, Quezada CV. A universal electric genetic algorithm for constrained optimization. Proc. IEEE Int. Congr. on Intelligent Techniques and Soft Computing, 1998: 518-22.
6. Hoffmeister F, Sprave J. Problem-dependent handling of constraints by use of metric penalty functions. Proc. IEEE Int. Congr. on Evolutionary Programming, 1996: 289-94.
7. Jones J, Houck C. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. Proc. IEEE Int. Conf. on Evolutionary Computation, 1994: 579-84.
8. Alouane ABH, Bean JC. A genetic algorithm for the multiple choice integer program. Operations Research 1997;45:92-101.
9. Scalak SC, Shonkwiler R, Babar S, Aral M. Annealing a genetic algorithm over constraints. Availbale online at: http://vlead.mech.virginia.edu./publications/shenk paper.html.
10. Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 2000;41(2):113-27.
11. Michalewics Z, Nazhiyath J. GENOCOP III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints. Proc. IEEE Int. Conf. on Evolutionary Computation, 1995: 647-51.
12. Xiao J, Michalewics Z, Trojanowski K. Adaptive evolutionary planner/navigator for mobile robots. IEEE Trans. Evol. Computation 1997;1(1):18-28.
13. Surry PD, Radcliffe NJ, Boyd ID. A multi-objective approach to constrained optimization of gas supply networks: the COMOGA method. Lecture Notes in Computer Science 1995;1(1):166-80.
14. Paredis J. Co-evolutionary constraint satisfaction. Proc. IEEE Int. Congr. on Parallel Problem Solving from Nature, 1994: 46-55.
15. Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 2000;186:311-38.
16. Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Computation 2000;4(3):284-94.
17. Le TV. A fuzzy evolutionary approach to constraint optimization problems. Proc. IEEE Int. Conf. on Evolutionary Computation, 1995: 274-8.
18. Adeli H, Cheng NT. Augmented lagrangian genetic algorithm for structural optimization. Journal of Aerospace Engineering 1994;7(1):104-18.

6/24/2012