Research in Engineering Design
https://doi.org/10.1007/s00163-023-00431-5

ORIGINAL PAPER

Optimize or satisfice in engineering design?

Lin Guo[1] · Janet K. Allen[2] · Farrokh Mistree[3]

Received: 26 June 2022 / Revised: 11 September 2023 / Accepted: 20 December 2023
© The Author(s) 2024

Abstract
In this paper, we address the issue of whether to optimize or satisfice in model-based engineering design. When dealing with operations research problems in the context of engineering design, one may encounter (i) nonlinear, nonconvex objectives and constraints, (ii) objectives with different units, and (iii) computational models that are abstractions of reality and fidelity, Seeking a single-point optimal solution that meets the necessary and sufficient Karush–Kuhn–Tucker (KKT) conditions makes it impossible to obtain a solution that satisfies all the targeted goals. Instead, a method to identify satisficing solutions that satisfies necessary KKT condition but not the sufficient condition is proposed. These solutions are relatively robust, easy to acquire, and often good enough. In this paper, we demonstrate the combined use of the compromise Decision Support Problems and the adaptive linear programming algorithm, as proposed by Mistree and co-authors. This method is appropriate in formulating design problems and obtaining solutions that satisfy only the necessary KKT condition. Further, the use of the proposed method circumvents complications associated with the use of gradient-based optimization algorithms typically used to solve optimization problems. We discuss the efficacy of our proposed method using four test problems to illustrate how the satisficing strategy outperforms the optimizing strategy in model-based engineering design.

Keywords Satisficing · Karush–Kuhn–Tucker conditions · Model-based design

1 Frame of reference

Engineering design of complex systems typically requires a team of designers from multiple disciplines or backgrounds to discover intertwined alternatives and select among them (Reich et al. 2010). When selecting the most appropriate solution among alternatives using well-defined or fuzzy criteria, most designers cannot avoid relying on the utility function, no matter how the utility function is abstracted into a specific mathematical equation or tactically formed. There are different ways to identify and refine a utility function that facilitates decision making: through game theory (Vincent 1983), analytic hierarchy process (AHP) (Saaty 1994), expert-team discussion tools with evaluation matrices such as the Pugh controlled convergence method (Frey et al. 2009), voting theory (Arrow 2012), Pareto comparisons (Sen 2018), etc. Hazelrigg has pointed out the limitations of these methods and made clear why these methods are inappropriate for use in design (Hazelrigg 2010). Hazelrigg advocates the use of utility theory and the use of voting and social choice, the effectiveness and accuracy of shortcuts in engineering, possible risks of poor decision making, and the role of the nature of preferences, etc. Among the utility theory-based methods, optimization is one method that enables designers to select among alternatives. For example, using the simplex algorithm to identify vertex solutions and selecting the optimal one.

The simplex algorithm, proposed by Dantzig (Dantzig et al. 1955; Dantzig 1990), is used to identify solutions at the vertices as possible solutions. The vertices being tested are the vertices of a convex polyhedral set, usually bounded by the constraints and bounds of variables, which is known as the feasible space of a linear programming problem. The simplex algorithm is used to search vertices in this way: given the current vertex, identify the adjacent vertex that on the trajectory that makes the greatest contribution in

✉ Janet K. Allen
janet.allen@ou.edu

1 The Systems Realization Laboratory, Department of Industrial Engineering, South Dakota School of Mines and Technology, Rapid City, SD 57701, USA

2 The Systems Realization Laboratory, University of Oklahoma, 202 West Boyd Street, Room 219, Norman, OK 73019, USA

3 The Systems Realization Laboratory, University of Oklahoma, 865 Asp Avenue, Room 306, Norman, OK 73019, USA

optimizing the objective function. The algorithm performs this search iteratively until none of the adjacent vertices can be used to further optimize the objective, then the current vertex is returned as the optimal solution.

Not all design problems can be solved using utility theory-based optimization, especially the gradient-based optimization methods. In this paper, we focus on the problems that can be managed using mathematical programming, that is, the problems that can be abstracted and solved by minimizing an objective through determining the value of decision variables while system constraints and bounds on the variables wherein the utility function serves as the objective. In modeling the objective using utility theory we assume that all decision makers are rational and that the objective is perfect. Frequently this is not the case in engineering design.

The mathematical models of engineering design problems often have: (i) nonlinear, nonconvex objectives and constraints, (ii) multiple objectives with different units and scales, and (iii) *computational models that are abstractions of reality and fidelity,*. In spite of this, these problems are frequently addressed using gradient-based optimization techniques, although the mathematical requirements for optimization, the Karush–Kuhn–Tucker (KKT) conditions are not met. Further, an engineering system comprises of several subsystems. Even if a solution converges for a subsystem, it is unlikely to be a true optimal solution for the system.

Recognizing that mathematical models are abstractions of reality, and the objective function may not be exact in design, in this paper, we offer an alternate approach that involves *satisficing*. Therefore, instead of using an *optimization strategy*, we propose using a s*atisficing strategy* that results in solutions that meet an acceptable threshold and are satisfactory, but not necessarily optimal (Simon 1956, 1996). With the satisficing strategy, we account for the inaccuracy of the utility function, and we do not rely on utility theory (Byron 1998). We recognize the mathematical difficulties with optimization and instead use the compromise decision support problem (cDSP) construct which is solved using the adaptive linear programming algorithm (ALP), as developed by Mistree et al. (1993). The ALP algorithm is incorporated in the DSIDES software (Hajihashemi 2023). To obtain a satisficing solution, we relax the requirement for satisfying both the necessary and sufficient KKT conditions and require only the necessary condition to be satisfied. This approach allows us to obtain fairly rapid solutions to problems that are characterized by the difficulties described above (Guo and Chen 2023). The speed with which answers can be obtained allows us to search large regions of design space. Further information is available in the work of Guo (Guo 2021). The satisficing strategy is also used to select a solution from alternatives, but the criteria are different from optimization so that the pool of solution finalists are larger, diverse, and easier to acquire, because a satisficing solution only meets the necessary KKT conditions. The foundation of the satisficing strategy is no longer maximizing the utility of a design but identifying "good enough" solutions that are feasible, enable the goals to achieve their targets to an acceptable extent, and are relatively insensitive to uncertainties due to the nonlinear, multi-objective nature of the design. In essence, our focus, in line with many engineering designers' interests, is in identifying a range of solutions that are useful, feasible, and relatively insensitive to uncertainty (this results in changes in model structure, boundary, feasibility, and dimension). If a designer wants to find a single-point optimal solution using models that are abstractions of reality, then the alternate approach proposed in this paper should not be used.

Optimizers and satisficers in engineering design are from different cultures. They view engineering design from two very different perspectives. They subscribe to different premises, and, in our opinion, the twain shall never meet. Hence, the title of this paper *Optimize or satisfice in engineering design?*

In Section 2, we describe the formulation of an optimization problem and a satisficing problem, the KKT conditions, and the assumptions underlying the KKT conditions. In Section 3, four test problems are presented to illustrate the efficacy of adopting the satisficing strategy in the model-based design of complex engineering systems. We end with some closing remarks in Section 5.

2 The formulation of optimization problems and satisficing problems

There are differences between the *optimizing* and *satisficing* strategies These include differences in problem formulation, approximation, exploration, and evaluation which are important activities in engineering design (Guo 2021). These differences are embodied in the formulation construct, (the compromise decision support problem or cDSP), the approximation algorithm (the adaptive linear programming algorithm or ALP), exploration mechanism (the XPLORE function), and the evaluation method and are implemented in the DSIDES software (decision support in the design of engineering systems).

2.1 The differences and similarities in the formulation of the two strategies

For a given function $f(x)$, the Euler–Lagrange equations, developed by Leonhard Euler and Joseph–Louis Lagrange (Euler 1755, 1766), form a system of second-order ordinary differential equations $\nabla^2_{xx} f(x)$. The solutions to $\nabla^2_{xx} f(x)$ are stationary points of $f(x)$. Using the Euler–Lagrange equations, one can solve optimization problems which consist of variables that are to be minimized or maximized within set $\mathcal{F}$, where $\mathcal{F}$ is the feasible set of solutions to the problem, bounded by constraints and bounds. An optimization problem $\mathbb{O}$ is represented as follows.

The mathematical form of an optimization problem $\mathbb{O}$:

Given

$$f : \mathbb{R}^n \to \mathbb{R}, \mathcal{F} \subseteq \mathbb{R}^n,$$

$$\mathcal{F} = \left\{ x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \ldots, m, h_j(x) = 0, j = 1, \ldots, \ell \right\},$$

Find

$$x^*: f(x^*) \succeq f(x), \forall x \in \mathcal{F}$$

Any point x that is a local extrema of the set mapped by multiplying active equations with a non-negative vector is a local optima of $\mathbb{O}$ (Courant and Hilbert 1953), denoted as x^*. The elements of such a non-negative vector are Lagrange multipliers, μ and λ. An optimization strategy is, "formulating an optimization problem and obtaining optimal solutions."

One variant of optimization is goal programming. The format of a goal programming problem $\mathbb{O}^{goal}$ is represented as follows. A target value T is predefined for the objective function $f(x)$ as the right-hand side value, so the objective becomes an equation, and we call it a goal (Charnes et al. 1955). d^- and d^+ are deviation variable measuring the underachievement and over-achievement of the goal toward its target. The problem is solved by minimizing the deviation variables, which is minimizing the difference between $f(x)$ and T. In other words, goal programming is aimed at finding T's closest projection on $\mathcal{F}$.

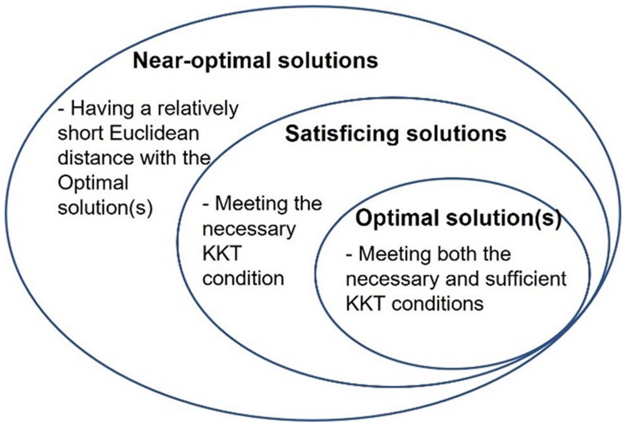
The mathematical form of a goal programming problem $\mathbb{O}^{goal}$:

Given

$$f : \mathbb{R}^n \to \mathbb{R}, \mathcal{F} \subseteq \mathbb{R}^n$$

$$\mathcal{F} = \left\{ x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \ldots, m, h_j(x) = 0, j = 1, \ldots, \ell d^- \bullet d^+ = 0, 0 \leq d^\mp \leq 1, Goal : f(x) + d^- - d^+ = T \right\},$$

Find



Near-optimal solutions
- Having a relatively short Euclidean distance with the Optimal solution(s)
Satisficing solutions
- Meeting the necessary KKT condition
Optimal solution(s)
- Meeting both the necessary and sufficient KKT conditions

Fig. 1 Relationship among optimal, satisficing, and near-optimal solutions

$$x^* : \mathbb{P}_{x \in \mathcal{F}}(Goal : f(x) = T).$$

Another strategy, a satisficing strategy, is to formulate a problem using the compromise decision support problem (cDSP) and obtain "good enough" but not necessarily optimal solutions (Mistree et al. 1993). For a nonlinear cDSP, we first linearize the nonlinear equations, including nonlinear constraints and nonlinear goal. Therefore, the nonlinear cDSP is approximated as a linear problem with a linear goal $Goal^{li}$, a linear feasible space $\mathcal{F}^{li}$ bounded by linear constraints $g(x)^{li} \geq 0$ and $h(x)^{li} = 0$. Thus, using the cDSP, we seek the closest projection from the linear goal set onto a linear feasible set. We define the solution as a satisficing solution, and we use x^s to denote it. A cDSP, $\mathbb{C}$, is represented as follows.

The mathematical form of a compromise decision support problem (cDSP) $\mathbb{C}$:

Given

$$f: \mathbb{R}^n \to \mathbb{R}, \mathcal{F} \subseteq \mathbb{R}^n,$$

$$\mathcal{F}^{li} = \left\{ x \in \mathbb{R}^n | g_i(x)^{li} \geq 0, i = 1, \ldots, m, h_j(x)^{li} = 0, j = 1, \ldots, \ell d_i^- \bullet d_i^+ = 0, 0 \leq d_i^\mp \leq 1, Goal^{li} : \frac{f(x)^{li}}{T} + d_i^- - d_i^+ = 1 \right\},$$

Find

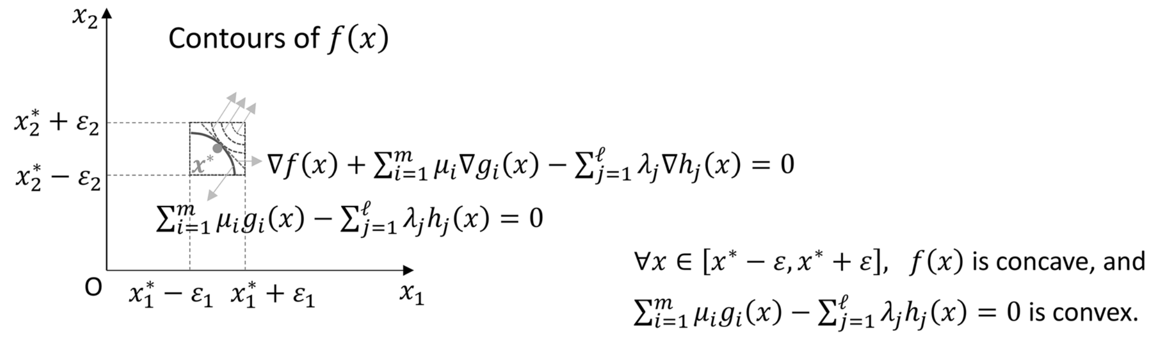

Contours of $f(x)$
x_2
$x_2^* + \varepsilon_2$
$x_2^* - \varepsilon_2$
x^*
$\nabla f(x) + \sum_{i=1}^m \mu_i \nabla g_i(x) - \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x) = 0$
$\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$
O
$x_1^* - \varepsilon_1$
$x_1^* + \varepsilon_1$
x_1
$\forall x \in [x^* - \varepsilon, x^* + \varepsilon]$, $f(x)$ is concave, and
$\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$ is convex.

Fig. 2 The first-order necessary Kuhn–Tucker conditions are satisfied at x^*

$$x^s : \mathbb{P}_{x \in \mathcal{F}^{li}} \left(Goal^{li} : f(x)^{li} = T \right).$$

A solution to a cDSP $\mathbb{C}$ is a satisficing solution denoted by x^s. To understand the difference between x^* and x^s, the KKT conditions are used. The relationship among optimal solutions, satisficing solutions, and near-optimal solutions is illustrated in Fig. 1.

2.2 The necessary and sufficient Karush–Kuhn–Tucker (KKT) conditions

In 1951, Harold W. Kuhn and Albert W Tucker proposed the Kuhn–Tucker conditions that are foundational to optimization (Kuhn and Tucker 1951). Later it was found that William Karush had summarized the necessary conditions in 1939 (Karush 1939). So, the Kuhn–Tucker conditions are now called the Karush–Kuhn–Tucker (KKT) conditions. The KKT approach is a generalization of the method of Lagrange multipliers (Lange 2013). A nonlinear problem with constraints can be represented as a Lagrange equation. The optimal point of the Lagrange function is a saddle point, so the KKT approach is also known as the saddle-point theorem (Hartley 1960).

The KKT conditions include first-order necessary conditions and second-order sufficient conditions. For an optimization problem $\mathbb{O}$, suppose that x^* is an optimal solution and μ_i and λ_j are Lagrange multipliers, the first-order necessary KKT conditions are represented in Eqs. 1–5.

Stationarity

$$\nabla f(x^*) + \sum\nolimits_{i=1}^{m} \mu_i \nabla g_i(x^*) - \sum\nolimits_{j=1}^{\ell} \lambda_j \nabla h_j(x^*) = 0. \tag{1}$$

Primal feasibility

$$g_i(x^*) \geq 0, \forall i = 1, \ldots, m, \tag{2}$$

$$h_j(x^*) = 0, \forall j = 1, \ldots, \ell \tag{3}$$

Dual feasibility

$$\mu_i \geq 0, \forall i = 1, \ldots, m. \tag{4}$$

Complementary slackness

$$\mu_i g_i(x^*) = 0, \forall i = 1, \ldots, m. \tag{5}$$

These first-order conditions are necessary conditions for optimality; at an optimal solution x^*, Eqs. 1–5 are always met. However, meeting Eqs. 1–5 does not guarantee that a solution is optimal. To guarantee optimality, the second-order sufficient conditions must also be met (Eqs. 6–8):

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x), \tag{6}$$

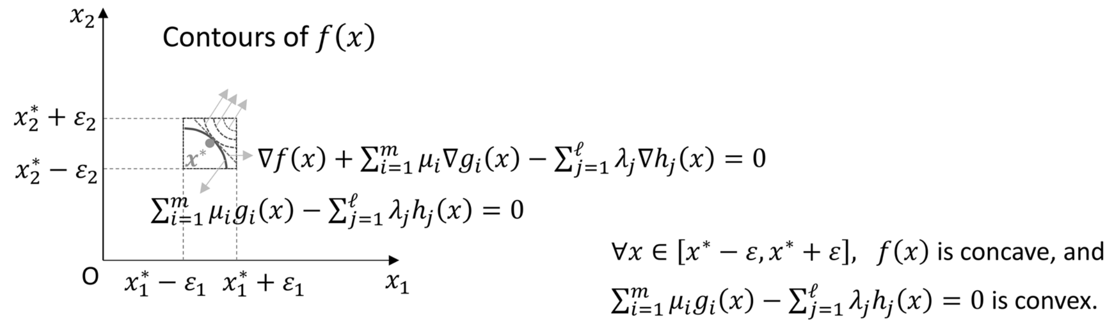
$$=> s^T \nabla_{xx}^2 \mathrm{L}(x^*, \lambda^*, \mu^*) \mathrm{s} \geq 0, \text{ where } s \neq 0, \tag{7}$$

$$\left[\nabla_x g_i(x^*), \nabla_x h_j(x^*) \right]^T s = 0. \tag{8}$$

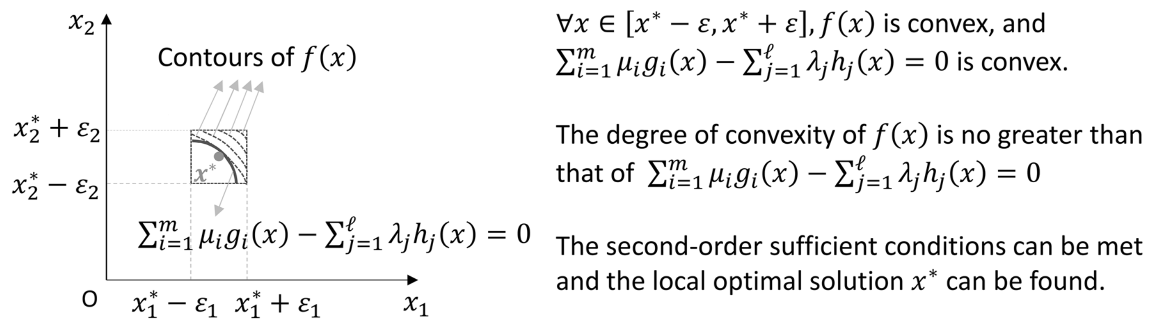
Equations 1–8 hold at an optimal solution x^*; whereas only Eqs. 1–5 hold for a satisficing solution x^s.

2.3 The analytical geometric meaning of the Kuhn–Tucker conditions

The analytical geometric meaning of the necessary conditions is that at the solution point x^*, where both the primal and the dual are feasible, the gradient vector of the objective $\nabla f(x^*)$ can be represented as the non-zero linear combination of the gradient matrix of all equality constraints


x_2
Contours of $f(x)$
$x_2^* + \varepsilon_2$
$x_2^* - \varepsilon_2$
x^*
$\nabla f(x) + \sum_{i=1}^{m} \mu_i \nabla g_i(x) - \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x) = 0$
$\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$
O
$x_1^* - \varepsilon_1$
$x_1^* + \varepsilon_1$
x_1
$\forall x \in [x^* - \varepsilon, x^* + \varepsilon]$, $f(x)$ is concave, and
$\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$ is convex.

(a)


x_2
Contours of $f(x)$
$x_2^* + \varepsilon_2$
$x_2^* - \varepsilon_2$
x^*
$\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$
O
$x_1^* - \varepsilon_1$
$x_1^* + \varepsilon_1$
x_1
$\forall x \in [x^* - \varepsilon, x^* + \varepsilon]$, $f(x)$ is convex, and
$\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$ is convex.
The degree of convexity of $f(x)$ is no greater than
that of $\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$
The second-order sufficient conditions can be met
and the local optimal solution x^* can be found.

(b)

Fig. 3 The convexity requirements for satisfying the second-order sufficient Kuhn–Tucker condition—when the degree of convexity of the objective does not exceed at least one non-negative linear combination of constraints. **a** Is when a linear combination of constraints is convex, and the contours of *f*(*x*) are concave. **b** Is when both of them are convex, but the degree of convexity of a linear combination of constraints is larger than the degree of convexity of the contours of *f*(*x*)

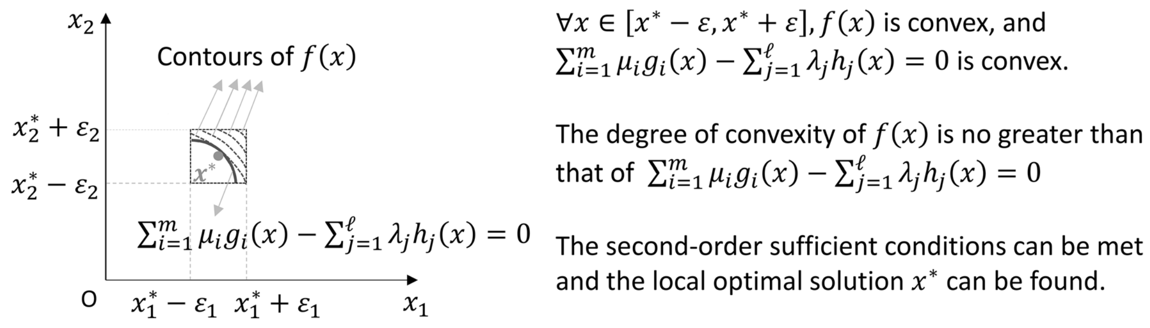

x_2
Contours of $f(x)$
$x_2^* + \varepsilon_2$
$x_2^* - \varepsilon_2$
x^*
$\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$
O
$x_1^* - \varepsilon_1$
$x_1^* + \varepsilon_1$
x_1
$\forall x \in [x^* - \varepsilon, x^* + \varepsilon]$, $f(x)$ is convex, and
$\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$ is convex.
The degree of convexity of $f(x)$ is no greater than
that of $\sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$
The second-order sufficient conditions can be met
and the local optimal solution x^* can be found.

Fig. 4 An optimal solution cannot be identified by Lagrange multipliers—when the convexity degree of the objective is higher than any non-zero linear combination of constraints

$\nabla h_j(x^*)$ and all the active inequality constraints[1] $\nabla g_i(x^*)$ for i iff $g_i(x^*) = 0$; Fig. 2.

The analytical geometric meaning of the sufficient condition is that at the solution point x^*, there exists a non-zero vector, s, that is orthogonal to the gradient matrix of all active inequality constraints and equality constraints, such that the second-order matrix of the Lagrange equation with respect to decision variables, x^*, is conditionally positive semidefinite: $s^T \nabla_{xx}^2 \mathrm{L}(x^*, \lambda^*, \mu^*) s \geq 0, \forall s \in \mathcal{S}$. In other words, the sufficient condition requires that the

[1] In this paper, we define an active constraint (or an active inequality constraint) as the constraint with zero slack or surplus at the solution point.

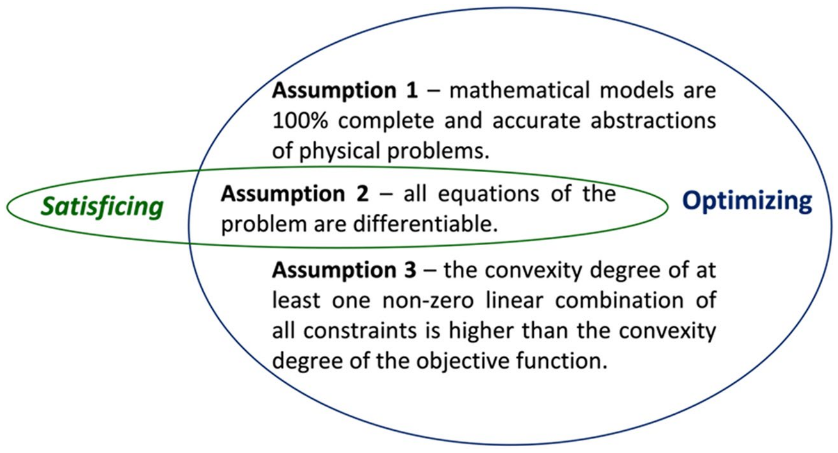

Assumption 1 – mathematical models are 100% complete and accurate abstractions of physical problems.
Satisficing
Assumption 2 – all equations of the problem are differentiable.
Optimizing
Assumption 3 – the convexity degree of at least one non-zero linear combination of all constraints is higher than the convexity degree of the objective function.

Fig. 5 The assumptions when using the optimizing strategy and the satisficing strategies

Lagrange equation is convex at x^*, that is, in a relatively small range containing x^*, the degree of convexity[2] of the objective should not exceed the degree of convexity of the constraints combined by Lagrange multipliers; see Fig. 3. However, the sufficient condition significantly reduces the possibility of achieving a solution for problems with an objective with a relatively higher degree of convexity, that is, problems in which the degree of convexity of the objective is higher than the degree of convexity of any non-zero linear combination of constraints, Fig. 4.

Design problems typically embody a highly convex objective, or a nonconvex and non-concave objective making it virtually impossible to find a solution using gradient-based optimization algorithms. To improve the design to: (i) closely achieve the target of the goals, and (ii) be robust to uncertainties that may cause variations to the boundary of the feasible set, $\mathcal{F}$, designers need to explore satisficing solutions. That is why we propose a satisficing strategy to deal with engineering design problems.

2.4 Assumptions when using the KKT conditions

There are assumptions when applying the necessary and sufficient KKT conditions to solve an optimization (nonlinear programming) problem. These assumptions also provide mathematical requirements for the problem in order to use the KKT conditions to obtain an optimal solution.

Assumption 1: The mathematical models of the constraints and goals used to model a design problem are complete and accurate representations of reality, namely the physical problem. This is to ensure that the optimal solution to a design problem is also optimal for the corresponding physical system. To use Lagrange functions to solve nonlinear problems, one must assume that the mathematical models accurately capture all the necessary details of the physical problem. Thus, an optimal solution to an engineering design problem is also optimal for its corresponding physical problem. As George Box said, "all models are wrong, but some are useful," (Box 1979), models are sometimes simplified, or, perhaps, not simplified correctly, and this may not be noticeable. When designing a system, there also may be information that cannot be captured and abstracted, such as hidden variables, implicit functional relationships among variables, factors defined as decision variables which cannot be fully controlled by designers, etc. Therefore, the optimal solution to an engineering design problem formulated as an optimization problem is unlikely to be optimal for the physical system.

Assumption 2: All equations of the problem are differentiable. Although according to the KKT conditions, it is only required that, within a small finite area around the optimal solution, the objective function, active constraints, and equality constraints should be differentiable, but, as we need to solve the Lagrange function globally, it is required that all equations should be differentiable.

Assumption 3: The degree of convexity of at least one non-zero linear combination of all constraints is higher than the degree of convexity of the objective. This assumption is critical to meet the KKT sufficiency condition. This condition guarantees finding an optimal solution to the design problem shown in Fig. 3 but cannot identify an optimal solution to the problem in Fig. 4.

When using an optimizing strategy, all three assumptions are applied, whereas when using the *satisficing* strategy, only the second assumption is applied; see Fig. 5.

2.5 Definition of robustness

There are several definitions of "robustness" in the engineering design literature. Carlson and Doyle define robustness as

[2] We define "convexity degree" as the average value of the diagonal terms of thee Hessian matrix of a function.

Table 1 Challenges in the model-based design of complex engineering systems and discussion of advantage

	Challenge surmounted	Discussion of advantage
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	Discussed in Section 3.5 and Table 12
Approximation	Using second-order sequential linearization	Discussed in Section 3.4 and Table 9
	Using accumulated linearization	Discussed in Section 3.4 and Table 9
Exploration	Combining interior-point search and vertex search	Discussed in Section 3.3 and Table 6
Evaluation	Allowing some violations of soft requirements, such as the bounds of deviation variables	Discussed in Section 3.6 and Table 15

a system's capacity to maintain some typical features under variations in the behavior of its components and environment (Carlson and Doyle 2002). After reviewing more than 170 papers, Beyer and Sendhoff summarize that, "the appeal of robust design is that its solutions and performance results remain relatively unchanged when exposed to uncertain conditions" (Beyer and Sendhoff 2007). Magni and co-authors indicate that, "robustness is the capacity of a controlled system working satisfactorily under the circumstances it may encounter in practice" (Magni et al. 1997). The performance of a solution algorithm is usually assessed by optimality, diversity of solutions, and the computational complexity of the algorithm (Soltani et al. 2002; Guo et al. 2022). Based on these ideas, in this paper, we consider robustness in model-based engineering design to have three components: (i) the capability of a method to identify feasible solutions given certain complexities, (ii) the capability of a model to maintain a solution under specific uncertainties, and (iii) a diversity of solutions for various representative design scenarios. Explanations of each robustness goal are as follows.

Robustness goal 1: Minimize the probability of having an empty satisficing set $\mathcal{S} \neq \emptyset$, given complexities ç; see Eq. 9. ç may include nonlinear and nonconvex equations, multiple goals (objectives) with different units or scales, and the targets of the multiple goals having different levels of achievability.

Robustness goal 2: Maximize the probability of maintaining a good enough (satisficing) solution x^s, under uncertainty P. The representation or result of uncertainty P is the variation of the feasible set $\widetilde{\mathcal{F}}$. Unlike some other methods that focus on eliminating uncertainties or minimizing model variations under uncertainties, that is to minimize $\Pr(P)$ or $\Pr\left(\widetilde{\mathcal{F}}|P\right)$, we acknowledge that model variations due to uncertainties may not be under a designers' control or awareness, and the optimal solution, x^*, may not be optimal with model variations, that is $x^* \notin \widetilde{\mathcal{F}}$ or $f(x^*) \ngeqq f(x)$, so we want to have a relatively high chance of maintaining a satisficing solution x^s; see Fig. 10.

Robustness goal 3: Maximize the diversity of solutions. Why should the diversity of solutions be associated with *robustness?* "Considering the goodness of a solution set regarding its distribution quality is key to multi-objective problems" (Farhang-Mehr and Azarm 2002). It is important to balance the convergence of the search algorithm and the diversity of solutions for multi-objective problems (Xu et al. 2022). Engineering designers often need different solutions under multiple design scenarios that represent their various design preferences among the multiple goals, meanwhile the diversity of solutions brought about by different solution processes, such as different search starting points, should be minimized. When design preferences evolve, solutions can be adapted. High solution diversity means that range and richness of solutions has been expanded therefore accommodating evolving design preferences. Therefore, we desire a large the sum of squares of solutions under all design scenarios SS_{DS} but a small sum of squares of solutions for all starting points SS_{SP}; see Eq. 11. This idea is based on ANOVA F-test (Wu and Hamada 2011).

$$\text{Robustness goal 1}: \text{ Minimize } \Pr\left(\mathcal{S} \neq \emptyset | \text{ç}\right), \tag{9}$$

$$\text{Robustness goal 2}: \text{ Maximize} \Pr(x^s|P), \tag{10}$$

$$\text{Robustness goal 3}: \text{ Maximize } SS_{DS}/SS_{SP}. \tag{11}$$

In Section 3, we use four test problems to demonstrate using the cDSP and the ALP (Mistree 1993) to

Table 2 The features of the test problems (TP)

Test problem (TP)	TP-I	TP-II	TP-III	TP-IV
Feature				
Multiple objectives	*	*	*	*
Nonlinear	*	*	*	*
Nonconvex		*	*	*
Objectives with various units (scales)			*	*
Goal targets with various degrees of achievability				*

Table 3 Methods for comparison of the optimizing and satisficing strategies

Strategy	Optimizing	Satisficing
Item		
Model formulation construct	Mathematical programming Goal programming	Compromise Decision Support Problem
Solution algorithms	Constrained Optimization by Linear Approximation (COBYLA) algorithm Trust-region constrained (trust-constr) algorithm Sequential Least Squares Programming (SLSQP) algorithm Nondominated Sorting Generation Algorithm II/III (NSGA II)	Adaptive Linear Programing (ALP) algorithm
Solver	Python SciPy.optimize	DSIDES

realize the satisficing strategy by removing Assumptions 1 and 3 (Fig. 5) and better achieving the robustness goals (Eqs. 9–11).

3 Advantages of using the satisficing strategy in engineering design

We use four test problems TP-I, TP-II, TP-III, and TP-IV (Table 2 and Sections 3.3–3.6), to demonstrate differences between the satisficing strategy and optimizing strategy. The satisficing strategy, in this paper, is accomplished by using the cDSP (the formulation construct) (Mistree et al. 1993), ALP (the solution algorithm) (Mistree et al. 1981, 1993), and DSIDES (the computational platform) (Ming et al. 2018). In this section, we introduce the challenge surmounted by using the cDSP and ALP in four stages of engineering design listed in Table 1, using test problems. We discuss the advantage of surmounting each challenge in the subsection and table association with the challenge.

3.1 Some typical features of engineering design problems

Key challenges in the model-based design of complex engineering systems? Janet K. Allen and Farrokh Mistree's design experience includes the co-design of materials (Sinha et al. 2013; Nellippallil et al. 2020; Messer et al. 2010), products (McDowell et al. 2010; Simpson et al. 2001; Pedersen et al. 2013; Choi et al. 2005), and associated manufacturing processes (Nellippallil et al. 2017, 2019, 2020), the design of mechanical, thermal, and structural systems (Nellippallil et al. 2019; Samadiani et al. 2010; Panchal et al. 2005); ships (Mistree et al. 1990) and aircraft (Chen et al. 2000; Simpson et al. 2001) and engineered networks (Milisavljevic-Syed et al. 2020) Based on this experience, in Table 1, we list 5 challenges for consideration. The test problems are very simple, in other words, they are just toy problems (Table 2). *Why do we use toy problems instead of more complicated problems?* Our rationale for using simple toy problems is: i) to easily visualize the goals and constraints on a two-dimension plane, ii) to demonstrate that some hyperparameters of NSGA II may impact the results (but we do not focus on accurately quantifying how each hyperparameter affects the results), and iii) these toy problems, although quite simple, do have typical complexities that engineering design problems may have[3].

The methods used in optimizing and satisficing strategies are listed in Table 3. For optimizing methods, we used the "SciPy.optimize" package[4]. There are ten algorithms in the package, but only three are used to solve the test problems. The algorithms that are not used are: the Nelder–Mead, Powell (Powell's conjugate direction method), (Powell 1964), CG (conjugate gradient methods), (Straeter 1971), and the BFGS, (Broyden–Fletcher–Coldfarb–Shannon algorithm) (Fletcher 1987) cannot easily manage problems with constraints. Newton-CG (Newton conjugate gradient method) (Khosla and Rubin 1981; Nash 1984), L-BFGS-B (an extension BFGS for large-scale, bounded problems) (Zhu et al. 1997), and TNC (truncated Newton method or Hessian-free optimization) (Zhu et al. 1997). These methods either cannot deal with problems without Jacobians[5] or return infeasible solutions without recognizing that they are infeasible.

[3] There are practical, complicated engineering design problems dealt with by the cDSP and ALP, and they cannot be solved using optimization, for example, hot rod rolling design ***(Nellippallil et al. 2017), reconfigurable manufacturing systems, architecting fail-safe supply networks (Rezapour et al. 2018). Those problems incorporate one or more of the typical features as listed in Table 1. We choose not to use those complicated problems as test problems in this paper because describing the problems clearly would require a lot of space and it is difficult to visualize the objectives, feasible space, and solutions for those high dimension problems.

[4] https://docs.scipy.org/doc/scipy/reference/optimize.html

[5] When using Newton-CG, even setting the Jacobian as false, the algorithm may not work without a partially provided Jacobian because the default temporary memory of Jacobian cannot be cleared; see: https://stackoverflow.com/questions/33926357/jacobian-is-required-for-newton-cg-method-when-doing-a-approximation-to-a-jaco

Therefore, in SciPy we use COBYLA, (the constrained optimization by linear approximation algorithm) (Powell 1989, 2007), Trust-constr[6], and SLSQP, (sequential least squares programming)[7] to solve the test problems.

3.2 Verification method: NSGA II

As NSGA II (non-dominated sorting genetic algorithm II) (Deb et al. 2002) can be used to solve problems with all five features listed in Table 2, therefore, we use NSGA II in MATLAB, as NSGA II can converge near-optimal solutions to engineering design problems with the five typical features we identified (in Table 2). Why is it necessary to study a satisficing algorithm to solve engineering design problems? The reason is that we observe that NSGA II has the following drawbacks that may prevent engineering designers from acquiring insight to improve the design formulation and exploring the solutions space:

NSGA II cannot give designers insight into possible ways to improve the model. NSGA II uses metaheuristics to search for solutions that generationally improve the optimality and diversity of the solutions, but information such as the bottlenecks in the model, the sensitivity of each segment of the model, the rationality of the dimensions of the model, or anything else that may indicate model improvement, are not provided with an NSGA II search. The performance of NSGA II, including convergence speed, optimality of solutions, and diversity of solutions, is sensitive to hyperparameter settings. Typical hyperparameters, such as the population size and generation number, must be predefined. However, usually designers assume that a larger population size or a larger number of generations returns better solutions, but they may not know how large is adequate, see TP-II and TP-III.
NSGA II requires more computational power than the satisficing algorithms.

Why do we use NSGA II as a verification algorithm if its performance depends on hyperparameter settings? By using a sufficient number of scenarios for hyperparameter settings, we may find the best result, and using that, we can verify the solutions obtained from the ALP are also near optimal. However, although NSGA II can be used to verify a test problem, when managing engineering designs, designers may be not able or willing to invest the effort in running a number of scenarios. The sensitivity to hyperparameter setting is a weakness of NSGA II since not every designer is aware of how sensitive it can be and how many scenarios are sufficient as the size, complexity, and nature of each problem may have different demands for determining the sufficient number of scenarios. Therefore, given that not all designers have the ability or patience to try a number of scenarios, we suggest that there is some uncertainty in NSGA II as to whether it can deliver an optimum solution. NSGA II has been used to verify the solutions to the test problems and we have invested the necessary effort to ensure that the solution is indeed an optimum.

Table 4 The optimization and satisficing models of TP-I

Strategy	Optimizing	Satisficing
TP		
TP-I	Objective Functions $f_1(x) = (x_1-1)^2 + (x_2-1)^2 + 3 \bullet x_1 \bullet x_2$ $f_2(x) = \frac{1}{2} \bullet (x_1-2)^2 + (x_2-2)^2 + x_1 \bullet x_2^2$ Constraints and Bounds $s.t. \begin{cases} x_1 \bullet x_2 \le 1 \\ 0 \le x_1 \le 2 \\ 0 \le x_2 \le 2 \end{cases}$ Combination of Objective Functions $Max \sum_{i=1}^{2} w_i \bullet f_i(x)$	Given $x_1, x_2, d_1^{\pm}, d_2^{\pm}$ $f_1(x) = (x_1-1)^2 + (x_2-1)^2 + 3 \bullet x_1 \bullet x_2$ $f_2(x) = \frac{1}{2} \bullet (x_1-2)^2 + (x_2-2)^2 + x_1 \bullet x_2^2$ Find $x_1, x_2, d_1^{\mp}, d_2^{\mp}$ Satisfy Goals: $\frac{f_1(x)}{14} + d_1^- - d_1^+ = 1$ $\frac{f_2(x)}{8} + d_2^- - d_2^+ = 1$ Constraints: $x_1 \bullet x_2 \le 1$ $d_i^- \bullet d_i^+ = 0, i = 1, 2$ Bounds: $0 \le x_1, x_2 \le 2$ $0 \le d_i^{\pm} \le 1$ Minimize $Z = \sum_{i=1}^{2} w_i \bullet (d_i^- + d_i^+)$

3.3 Advantage of problem exploration: combining interior-point searching and vertex searching

We use a multi-objective, nonlinear test problem (TP-I), to demonstrate the advantages of problem exploration using the cDSP and ALP, which makes designers less reliant on Assumption 1 and are better achieve Robustness Goal 3.

Formulation: The optimization model and the corresponding cDSP of TP-I are shown in Table 4. For the cDSP, a target value is assigned to each objective. Solving the problem requires minimizing the deviation between the achieved value and the goal target while satisfying all constraints and bounds.

[6] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-trustconstr.html

[7] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html

Table 5 Solutions to TP-I (dominated solutions for each weight are in italics)

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II	
		Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$
(1, 0)	(0.5, 1)	(2, 0.5)	4.25	(2,0.5)	4.25	(2,0.5)	4.25	(2,0.5)	4.25	(2,0.5)	4.25
	(0, 0)	*(0, 0)*	*2*	*(0, 0)*	*2*	*(0, 0)*	*2*				
	(2, 0.5)	(2, 0.5)	4.25	(2,0.5)	4.25	(2,0.5)	4.25				
(0, 1)	(0.5, 1)	*(0.5, 2)*	*3.13*	(0, 0)	6	(0, 0)	6	(0, 0)	6	(0, 0)	6
	(0, 0)	(0, 0)	6								
	(2, 0.5)					*(2, 0)*	*4*				
(0.5,0.5)	(0.5, 1)	*(0.5, 2)*	*3.69*	*(0.5 2)*	*3.69*	*(0.5,2)*	*3.69*	*(0.5,2)*	*3.69*	(0, 0)	4
	(0, 0)	(0, 0)	4	(0, 0)	4	(0, 0)	4				
	(2, 0.5)	*(2, 0.5)*	*3.5*	*(2,0.5)*	*3.5*	*(2,0.5)*	*3.5*				
(0.7,0.3)	(0.5, 1)	(0.5, 2)	3.91	(0.5,2)	3.91	(0.5,2)	3.91	(0.5,2)	3.91	(0.5,2)	3.91
	(0, 0)	*(0, 0)*	*3.2*	*(0, 0)*	*3.2*	*(0, 0)*	*3.2*				
	(2, 0.5)	*(2, 0.5)*	*3.8*	*(2,0.5)*	*3.8*	*(2,0.5)*	*3.8*				
(0.3,0.7)	(0.5, 1)	*(2, 0)*	*3.4*	(0, 0)	4.8	(0, 0)	4.8	(0, 0)	4.8	(0, 0)	4.8
	(0, 0)	(0, 0)	4.8								
	(2, 0.5)	*(2, 0.5)*	*3.2*	*(2,0.5)*	*3.2*	*(2,0.5)*	*3.2*				

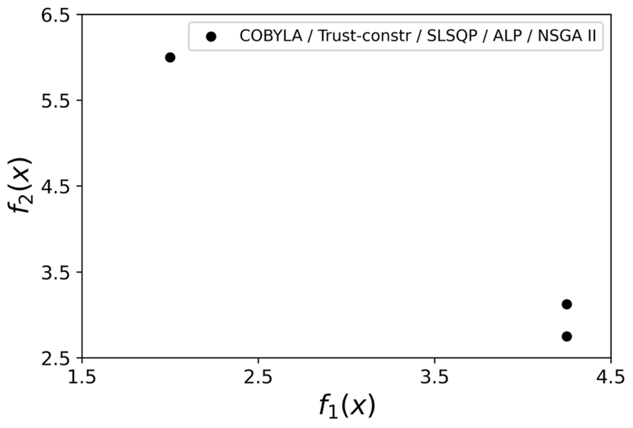


Fig. 6 The solution points to TP-I on the objective space using five algorithms: all the five solution algorithms return the same three solution points

Weights (weight vectors or weight scenarios): the problem has two goals (objectives) but the preferences for the goals are unknown. So, we use an Archimedean strategy (scalarization) (Jahn 1985; Lin 1976; Mistree et al. 1993) to combine the goals and use different weights to explore various preferences and understand the tradeoffs among the goals. We apply five weight scenarios, as listed in the "Weight" column in Table 5. *Why do we use multiple weight scenarios?* Suppose that we do not know how to set the weight—this happens quite often in multi-goal engineering designs—so we need to explore different weight scenarios and associate them with design preferences. We apply the weight generation method in (Messac and Mattson 2002) and use the version customized to engineering design problems (Guo et al. 2021) to obtain weight scenarios. For example, for a two-goal problem, due to the non-orthogonality between the two goals and the variation in sensitivity of each goal along each dimension, the weight scenario (0.5, 0.5) may neither allow the two goals to be equally prioritized, nor allow the two goals to be achieved halfway through to targets. A weight scenario (0.7, 0.3) does not mean that Goal 1's target can be achieved 70% whereas Goal 2's target can be achieved 30%. In short, the weight for each goal is not linear with the rate of achieving their targets. Therefore, we try multiple weight scenarios to realize different design preferences. *Why do we use five scenarios?* First, we start with two weight scenarios, (0, 1) and (1, 0) as "parents." If the solution to the two weight scenarios are different, we choose another weight scenario as a "child" weight scenario, laying between the parents, for example (0.5, 0.5). This process is repeated until the new child weight scenario has the same solution as one of its parents. For Test Problem 1 (TP-I), using this method, we stop introducing more weight scenarios when five scenarios have been used.

Starting points: To test the sensitivity of each solution algorithm to starting points, we intuitively select three starting points that are vertices of the feasible space bounded by the constraint and bounds.

Results are listed in Table 5. Using five weight scenarios and trying three starting points, for every solution algorithm, three solutions are obtained (only the best solutions among all starting points are shown in Fig. 6), as visualized on objective space in Fig. 6 and on x-$f(x)$ Space in Fig. 7. As TP-I is to maximize $f_1(x)$ and $f_2(x)$, in Fig. 6, ideal solutions should be close to the upper right corner of the objective space, and in Fig. 7, the value of the objective function should be as large as possible. For TP-I, the solutions using different algorithms are identical.

Observation: For a test problem with nonlinear constraints and multiple objectives, the same solutions are obtained using the optimizing and satisficing strategies. The results of the optimizing solution algorithms

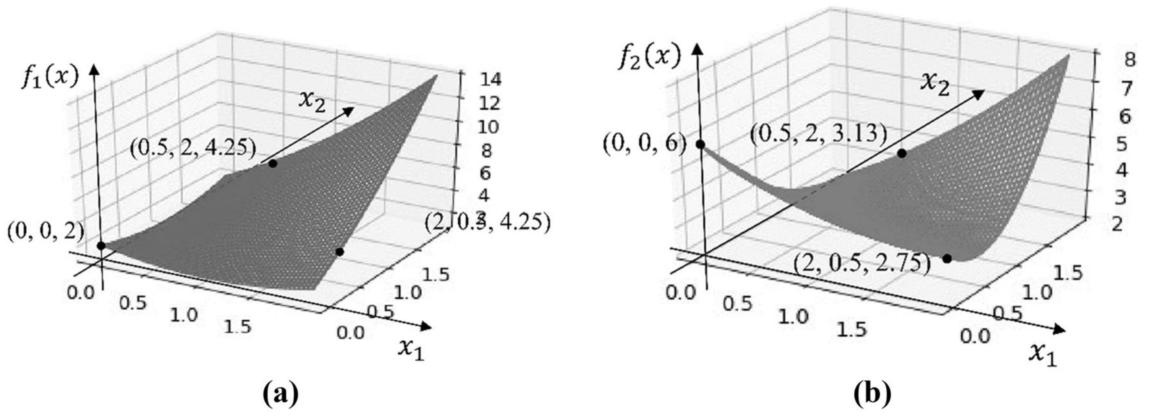
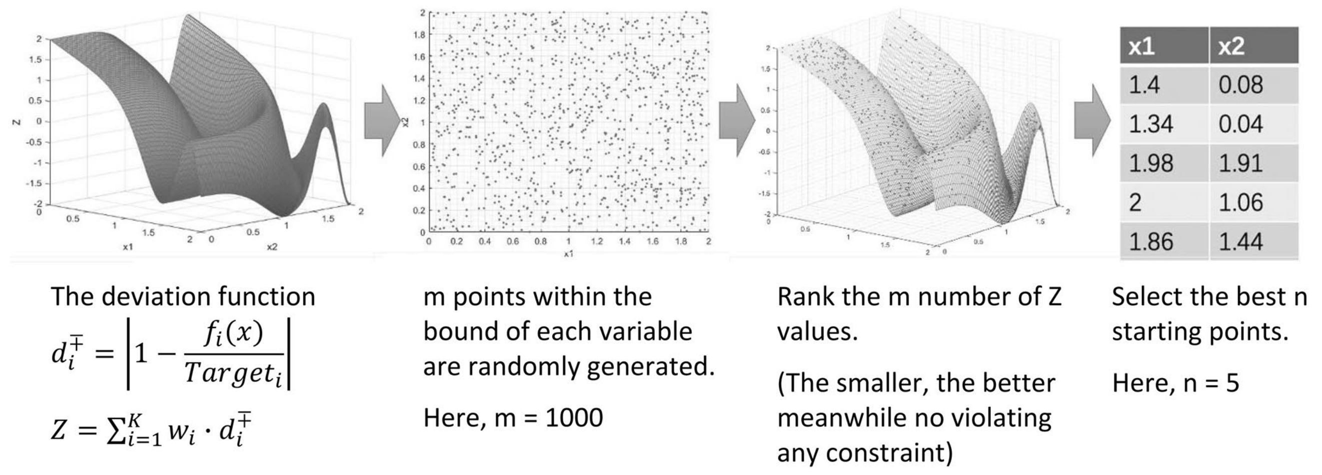


Fig. 7 The solution points to TP-I on the *x-f(x)* space. **a** Is the 3D illustration of $f_1(x)$. **b** Is the 3D illustration of $f_2(x)$



x1	x2
1.4	0.08
1.34	0.04
1.98	1.91
2	1.06
1.86	1.44

The deviation function

$d_i^{\mp} = \left|1 - \frac{f_i(x)}{Target_i}\right|$

$Z = \sum_{i=1}^{K} w_i \cdot d_i^{\mp}$

m points within the bound of each variable are randomly generated.

Here, m = 1000

Rank the m number of Z values.

(The smaller, the better meanwhile no violating any constraint)

Select the best n starting points.

Here, n = 5

Fig. 8 Using the “XPLORE” module in DSIDES to identify the best starting points

(COBYLA, Trust-constr, and SLSQP) are sensitive to the starting point. In other words, using the satisficing strategy, Robustness Goal 3 is better achieved versus using optimizing strategy; Eq. 12. If the model is incomplete or of low fidelity, avoiding being trapped in a local optimum allows an engineering designer to have a better chance of obtaining a solution for the physical system, which means that Assumption 1 is less important.

$$\left[\mathrm{SS}_{\mathrm{DS}}/\mathrm{SS}_{\mathrm{SP}}\right]_{\mathrm{Satisficing}} \geq \left[\mathrm{SS}_{\mathrm{DS}}/\mathrm{SS}_{\mathrm{SP}}\right]_{\mathrm{Optimizing}}. \tag{12}$$

Why are the solutions insensitive to the starting point when using the cDSP and ALP? This is the result of the combination of interior-point searches and vertex searches in the “XPLORE” module in the DSIDES software. In Fig. 8, we illustrate the “XPLORE” procedure to identify appropriate starting points for approximation and solution search. “*m*” points within the bounds of each variable are chosen randomly. Then, the value of the goal and the feasibility at “*m*” points are computed. The feasible points are ranked and “*n*” of them with the lowest deviation from the goal are selected as starting points and used sequentially. The nonlinear equations are approximated as linear equations around a starting point using second-order sequential linearization, and then the linear problem is solved using the revised dual simplex algorithm (Ignizio 1985). Usually “m” is chosen to be a number in [800, 1200], and “*n*” is chosen to be a number in [5, 20]. Therefore, using “XPLORE,” the feasible points with relatively good goal-achievement values are selected as starting points. To some extent this reduces the chances of returning a local optimum.

We summarize the advantages in exploration and fill in the corresponding blank in Table 1 which becomes Table 6.

3.4 Advantage in approximation: using second-order sequential linearization and constraint accumulation

The test problem II (TP-II), a multi-objective problem with a nonconvex equation, is used to demonstrate how Assumption 3 is removed and robust goals 1 and 3 are achieved better. The formulations of TP-II using the different strategies are in Table 7. The solutions are listed in Table 8. The best solutions (among different starting points) of each design scenario are visualized on objective space and x-$f(x)$ Space in Figs. 9 and 10, respectively. When using NSGA II to verify the results, we use two values of the population size hyperparameter. With a population of 50, solutions are closer to the upper right corner, so only the results of using NSGA II with a population of 50 are presented here.

Observation: Using the cDSP and ALP, we obtain more diverse solutions closer to the upper right corner, that is, more satisficing solutions are obtained when changing design scenarios. Using the satisficing strategy, Robustness Goal 1 is better achieved for nonconvex problems; see Eq. 13. Therefore, designers can manage nonconvex problems, especially when the degree of convexity of a goal exceeds the degree of convexity of the linear combinations of constraints in a local area, which makes designers rely less on Assumption 3.

$$\begin{aligned}&\left[Pr(S \neq \emptyset|\text{Nonconvexity})\right]_{\text{Satisficing}}\\ &\geq \left[Pr((S \neq \emptyset|\text{Nonconvexity})\right]_{\text{Optimizing}}.\end{aligned} \tag{13}$$

NSGA II solutions are sensitive to hyperparameter setting but can return non-dominated solutions if the hyperparameter is set appropriately; however, the solutions are not diverse. DSIDES can return satisficing and relatively

Table 6 Summary of advantages in exploration

Stage	Feature	Advantage	Assumption removed	Test problem (TP)	Robustness goal	Introduction
Exploration	Combining interior-point search and vertex search	Designers can avoid getting trapped in local optima to some extent and identify satisficing solutions which are relatively insensitive when the starting points changing	Assumption 1	TP-I	Robustness Goal 3	Section 3.3

Table 7 The optimization model and compromise DSP of the TP-II

Strategy	Optimizing	Satisficing
TP II	Objective Functions $f_1(x) = cos(x_1{}^2 + x_2{}^3)$ $f_2(x) = \frac{1}{2} \bullet (x_1 - 2)^3 + (x_2 - 2)^3 + x_1 \bullet x_2{}^2$ Constraints and Bounds $s.t. \begin{cases} x_1 \bullet x_2 \le 1 \\ 0 \le x_1 \le 2 \\ 0 \le x_2 \le 2 \end{cases}$ Combination of Objective Functions $Max \sum_{i=1}^{2} w_i \bullet f_i(x)$	Given $x_1, x_2, d_1{}^{\pm}, d_2{}^{\pm}$ $f_1(x) = \cos(x_1{}^2 + x_2{}^3)$ $f_2(x) = \frac{1}{2} \bullet (x_1 - 2)^3 + (x_2 - 2)^3 + x_1 \bullet x_2$ Find $x_1, x_2, d_1{}^{\mp}, d_2{}^{\mp}$ Satisfy Goals: $\frac{f_1(x)}{1.2} + d_1{}^- - d_1{}^+ = 1$ $\frac{f_2(x)}{8} + d_2{}^- - d_2{}^+ = 1$ Constraints: $x_1 \bullet x_2 \le 1$ $d_i{}^- \bullet d_i{}^+ = 0, i = 1, 2$ Bounds: $0 \le x_1, x_2 \le 2$ $0 \le d_1{}^{\pm}, d_2{}^{\pm} \le 1$ Minimize $Z = \sum_{i=1}^{2} w_i \bullet (d_i{}^- + d_i{}^+)$

diverse solutions and robustness goal 3 is better achieved using satisficing; see Eq. 14.

$$[SS_{DS}/SS_{SP}]_{Satisficing} \ge [SS_{DS}/SS_{SP}]_{NSGAII}. \tag{14}$$

Why does the cDSP and ALP manage nonconvex problems and return solutions close to the non-dominated solutions (the solutions returned by NSGA II)? Two mechanisms in the ALP make it possible to linearize the nonconvex function relatively accurately and converge with diverse solutions—using second-order sequential linearization and the accumulation of the linear constraints.

The ALP algorithm incorporates a local approximation algorithm (Mistree et al. 1981, 1993; Chen et al. 1996), in which a secant plane of the paraboloid (with the second-order derivatives at the starting point as the coefficients) replaces the original nonlinear function, as in Fig. 11, in which two dimensions of a problem being approximated in two iterations (synthesis cycle) is shown. The weighted-sum of the goals is $\sum_{k \in K} W_k \bullet G_k$. The starting point X_0^0 may not be in the feasible region. A random search or a Hook-Jeeves pattern search can be invoked to identify a point X_1^0 in the feasible area. In the first iteration, the problem is linearized at X_1^0.

In iteration i, Fig. 11a, a nonlinear constraint NF_j is linearized at X_i^0, so an approximated linear constraint $LF_{i,j}$ is obtained. Doing this for all nonlinear functions and framing a linear model, the revised simplex dual algorithm is used to obtain Solution X_i^*. Using the reduced move coefficient

Table 8 Solutions to TP-II—dominated solutions of each weight are in *italics*; for each method, the best solution of each design scenario is marked using a capital letter (A, B, C, D, E, F, G, H, and J)

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II	
		Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$
(1, 0)	(0.5, 1)	A (0, 0)	1	A (0, 0)	1	A (0, 0)	1	*F (0.5,1.84)*	*0.97*	A (0, 0)	1
	(0, 0)										
	(2, 0.5)			*(2, 0.5)*	*− 0.55*	*(2, 0.5)*	*− 0.55*				
(0, 1)	(0.5, 1)	B (0.5, 2)	0.31	B (0.5, 2)	0.31	B (0.5, 2)	0.31	G (0.51, 1.96)	0.31	B (0.5, 2)	0.31
	(0, 0)										
	(2, 0.5)	*(2.0, 0.5)*	*-2.88*								
(0.5, 0.5)	(0.5, 1)	*(0.9, 1.12)*	*− 0.42*	*D (0.9, 1.12)*	*− 0.42*	*(0.9, 1.12)*	*− 0.42*	*H (0.52, 1.87)*	*0.54*	C (0.55, 1.82)	0.64
	(0, 0)					C (0.55, 1.82)	0.64				
	(2, 0.5)	C (0.55, 1.82)	0.64			*(2, 0.5)*	*− 1.71*				
(0.7, 0.3)	(0.5, 1)	C (0.55, 1.82)	0.79	*E (0.73, 0.88)*	*− 0.31*	*(0.73, 0.88)*	*-0.31*	*F (0.5, 1.84)*	*0.68*	J (0.56, 1.8)	0.79
	(0, 0)					C (0.55, 1.82)	0.79				
	(2, 0.5)	*(0.73, 0.88)*	*− 0.31*	*(2, 0.5)*	*− 1.25*	*(2, 0.5)*	*− 1.25*				
(0.3, 0.7)	(0.5, 1)	C (0.55, 1.82)	0.5	C (0.55, 1.82)	0.5	C (0.55, 1.82)	0.5	C (0.55, 1.82)	0.5	C (0.55, 1.82)	0.5
	(0, 0)										
	(2, 0.5)	*(2, 0.5)*	*− 2.18*								

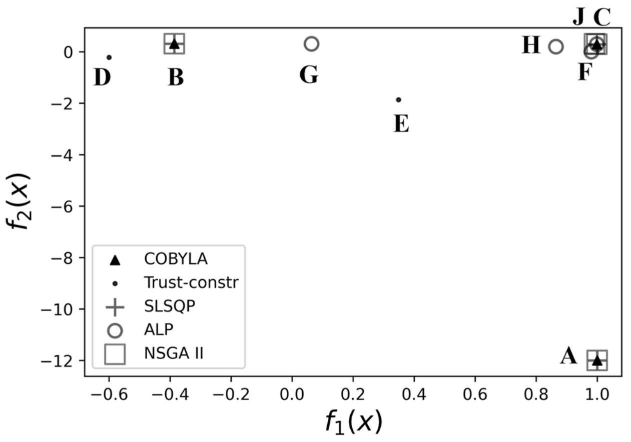


Fig. 9 The solution points to TP-II on the objective space—solutions returned by ALP are relatively close to the upper right corner and diverse

(RMC) to linearly combine X_i^0 and X_i^*, we find the starting point of the next iteration X_{i+1}^0. In Iteration *i+1*, Fig. 11b, the approximated linear constraints of all previous iterations (in Fig. 11, we only show $LF_{i,j}$ and $LF_{i+1,j}$) are accumulated, and a solution X_{i+1}^*, is returned and the starting point of iteration *i+2* , X_{i+2}^0, is obtained using the RMC to linearly combine X_{i+1}^0 and X_{i+1}^*. This procedure is iterated for n times.

We use Fig. 12a to illustrate the second-order sequential linearization used in the ALP. In Iteration *i*, first, NF_j (Paraboloid *ABC*) is approximated to $NF_{i,j}$" (Paraboloid AB^*C^*) with the diagonal terms of its Hessian matrix at X_i^0 as coefficients. Then, NF_j" is approximated to a secant Plane $LF_{i,j}$ (Plane AB^*C^*). NF_j" and $LF_{i,j}$ are computed as follows.

$NF_{i,j}$" is obtained using the second-order full derivatives at X_i^0, Eq. 15, because the second-order partial derivatives have limited impact on the gradient (Mistree et al. 1981).

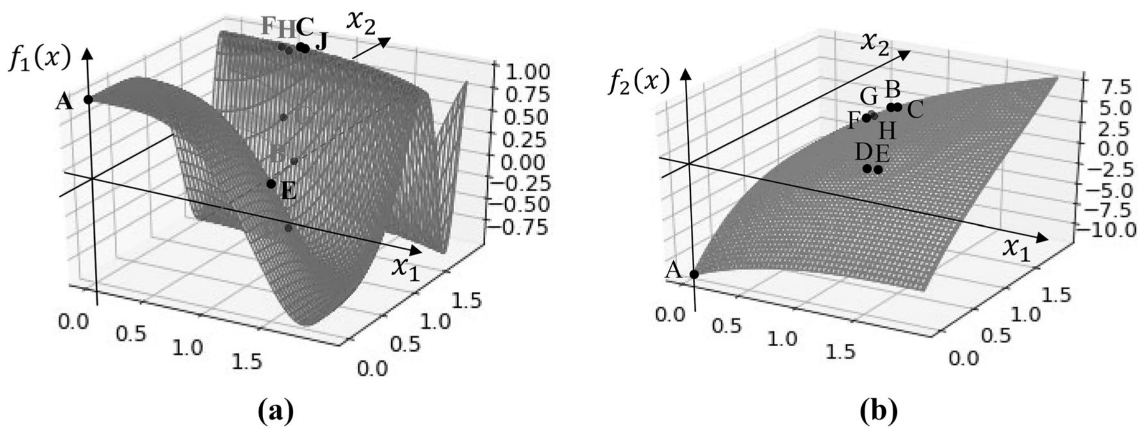


Fig. 10 The solution points to TP-II on the *x*-*f*(*x*) space—using trust-constraint and SLSQP fall into local optima. **a** Is the 3D illustration of $f_1(x)$. **b** Is the 3D illustration of $f_2(x)$

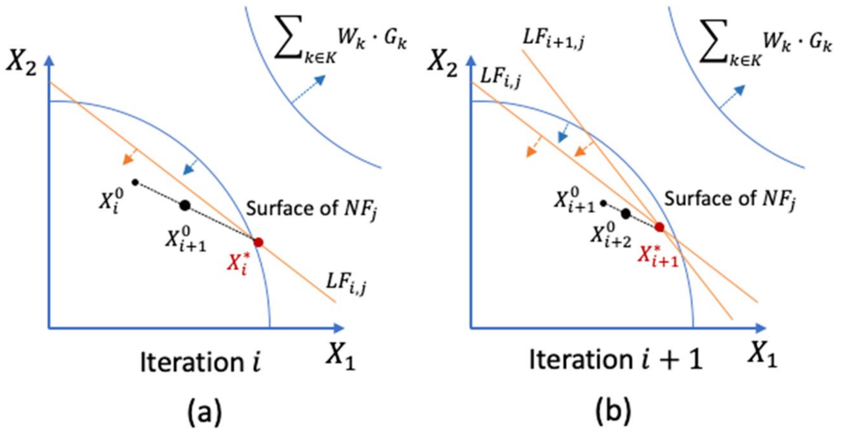


Fig. 11. The approximation and solution using ALP in two iterations. **a** Is Iteration *i*. **b** Is Iteration *i*+1

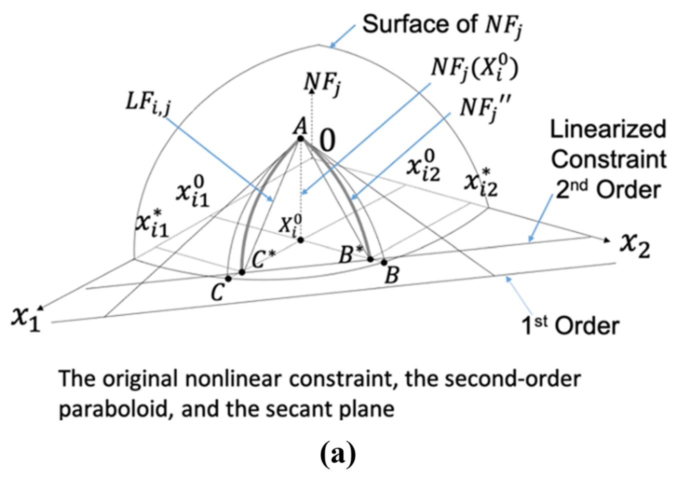


Fig. 12 The two-step linear approximation methods using the ALP. **a** Illustrates the situation when the second-order paraboloid of the original nonlinear constraint has intersection with plane $x_1 - x_2$. **b** Illustrates the situation when the second-order paraboloid of the original nonlinear constraint has no intersection with plane $x_1 - x_2$

$$
\begin{aligned}
NF_{i,j}'' =\, & NF_j\left(X_i^0\right) \\
& + \sum_{p=1}^{n}\left(x_{ip} - x_{ip}^0\right)\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0 \\
& + \frac{1}{2}\sum_{p=1}^{n}\left(x_{ip} - x_{ip}^0\right)^2\left(\frac{\partial^2 NF_j}{\partial x_{ip}^2}\right)_0 .
\end{aligned} \tag{15}
$$

From Fig. 15, for the p^{th} dimension, the quadratic to be solved to obtain $\left(x_{ip} - x_{ip}^0\right)$ is:

$$
NF_j\left(X_i^0\right) + \left(x_{ip} - x_{ip}^0\right)\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0 + \frac{1}{2}\left(x_{ip} - x_{ip}^0\right)^2\left(\frac{\partial^2 NF_j}{\partial x_{ip}^2}\right)_0 = 0. \tag{16}
$$

If Fig. 16 has real roots, as is the situation in Fig. 12a, by solving Eq. 16 and selecting the root between Eqs. 17 and 18 with the smaller absolute value for each dimension, we obtain an intersection that is closer to the paraboloid in each dimension, that is $B*$ and $C*$.

$$
\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^* = \frac{-NF_j\left(X_i^0\right)\left(\frac{\partial^2 NF_j}{\partial x_{ip}^2}\right)_0}{-\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0 - \sqrt{\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^2 - 2NF_j\left(X_i^0\right)\left(\frac{\partial^2 NF_j}{\partial x_{ip}^2}\right)_0}}, \tag{17}
$$

$$
\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^* = \frac{-NF_j\left(X_i^0\right)\left(\frac{\partial^2 NF_j}{\partial x_{ip}^2}\right)_0}{-\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0 + \sqrt{\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^2 - 2NF_j\left(X_i^0\right)\left(\frac{\partial^2 NF_j}{\partial x_{ip}^2}\right)_0}}. \tag{18}
$$

If Eq. 16 has no real roots, as is the case in Fig. 12b, NF_i does not intersect with Plane x, and the first-order derivative at X_i^0 is used, as in Eq. 19.

$$
\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^* = \left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0. \tag{19}
$$

Based on the intersections in each dimension, for example, $B*$ and $C*$ we get LF_i.

$$
LF_{i,j} = \sum_{p=1}^{n} x_{ip}\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^* - \left(\sum_{p=1}^{n} x_{ip}^0\left(\frac{\partial NF_j}{\partial x_{ip}}\right)_0^* - NF_j\left(X_i^0\right)\right). \tag{20}
$$

If the degree of convexity of NF_j is positive or slightly negative (greater than – 0.015) at the starting point of the i^{th} iteration, and if the constraint is active in the (i-1)th iteration, that is, X_{i-1}^* is on the surface of NF_j,, then the accumulated constraints replace NF_j, Eq. 21; otherwise, the single linear constraint in the ith iteration replaces NF_j, Fig. 22.

Constraint Accumulation Algorithm

In the ith iteration

for every j in J

if $\quad \text{degree}_{\text{convexity}} = \frac{1}{n}\sum_{p=1}^{n}\frac{\partial^2 NF_j}{\partial x_{ip}^2} \le -0.015$

and $NF_j\left(X_{i-1}^*\right) = 0$

$$
LF_{i,j,} = LF_{i-1,j} \cup LF_{i,j} \tag{21}
$$

else

$$
LF_{i,j,} = LF_{i,j} \tag{22}
$$

Then, the revised simplex dual algorithm is invoked to solve the linear problem P_i^L, so a solution X_i^* is obtained. A point X_{i+1}^0, which is a point between the starting point X_i^0 and the solution X_i^*, becomes the starting point of the next iteration.

Constraint accumulation is only used when the local convexity of an equation is greater than – 0.015, which is a value determined by experiments and heuristics. When the degree of convexity of an equation around the starting point is greater than – 0.015, the equation at the linearization point is convex or slightly concave, as the example shown in

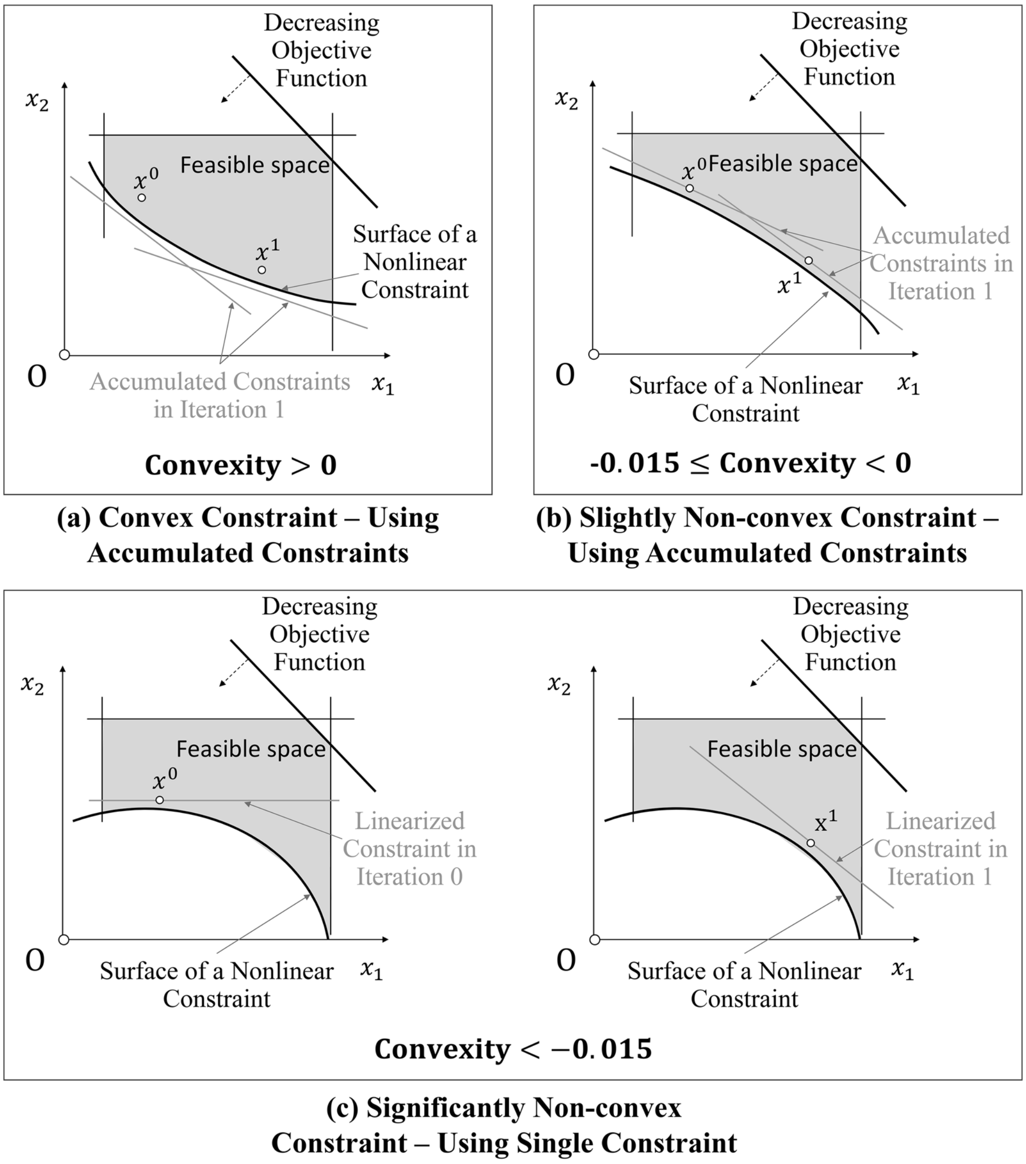

Decreasing Objective Function
x_2
Feasible space
x^0
x^1
Surface of a Nonlinear Constraint
O
Accumulated Constraints in Iteration 1
x_1
Convexity > 0
(a) Convex Constraint – Using Accumulated Constraints
Decreasing Objective Function
x_2
x^0 Feasible space
Accumulated Constraints in Iteration 1
x^1
O
Surface of a Nonlinear Constraint
x_1
-0.015 ≤ Convexity < 0
(b) Slightly Non-convex Constraint – Using Accumulated Constraints
Decreasing Objective Function
x_2
Feasible space
x^0
Linearized Constraint in Iteration 0
O
Surface of a Nonlinear Constraint
x_1
Decreasing Objective Function
x_2
Feasible space
x¹
Linearized Constraint in Iteration 1
O
Surface of a Nonlinear Constraint
x_1
Convexity < −0.015
(c) Significantly Non-convex Constraint – Using Single Constraint

Fig. 13 Using accumulated constraints from multiple linearization iterations for a convex equation (**a**) or slightly nonconvex equation (**b**) and using a single linearized constraint for a significantly nonconvex constraint (**c**)

Table 9 Summary of advantages in approximation

Stage	Feature	Advantage	Assumption removed	Test problem (TP)	Robustness goal	Introduction
Approximation	Using second-order sequential linearization	Designers can balance between linearization accuracy and computational complexity	Assumption 3	TP-II	Robustness Goal 1 and 3	Section 3.4
	Using accumulated linearization	Designers can manage nonconvex problems and deal with highly convex, nonlinear problems relatively more accurately				

Table 10 The optimization model and compromise DSP of TP-III

Strategy	Optimizing	*Satisficing*
TP III	<u>Objective Functions</u> $f_1(x) = cos\left(x_1{}^2 + x_2{}^3\right)$ $f_2(x) = 25 \bullet (x_1 - 2)^3 + 50 \bullet (x_2 - 2)^3 + 50 \bullet x_1 \bullet x_2{}^2$ <u>Constraints and Bounds</u> $s.t. \begin{cases} x_1 \bullet x_2 \leq 1 \\ f_1(x) \geq 0 \\ f_2(x) \geq 0 \\ 0 \leq x_1 \leq 2 \\ 0 \leq x_2 \leq 2 \end{cases}$ <u>Combination of Objective Functions</u> $Max \sum_{i=1}^{2} w_i \bullet f_i(x)$	<u>Given</u> $x_1, x_2, d_1{}^{\pm}, d_2{}^{\pm}$ $f_1(x) = \cos\left(x_1{}^2 + x_2{}^3\right)$ $f_2(x) = 25 \bullet (x_1 - 2)^3 + 50 \bullet (x_2 - 2)^3 + 50 \bullet x_1 \bullet x_2{}^2$ <u>Find</u> $x_1, x_2, d_1{}^{\mp}, d_2{}^{\mp}$ <u>Satisfy</u> <u>Goals:</u> $\frac{f_1(x)}{1.2} + d_1{}^- - d_1{}^+ = 1$ $\frac{f_2(x)}{400} + d_2{}^- - d_2{}^+ = 1$ <u>Constraints:</u> $x_1 \bullet x_2 \leq 1$ $f_1(x) \geq 0$ $f_2(x) \geq 0$ $d_i{}^- \bullet d_i{}^+ = 0, i = 1, 2$ <u>Bounds:</u> $0 \leq x_1, x_2 \leq 2$ $0 \leq d_1{}^{\pm}, d_2{}^{\pm} \leq 1$ <u>Minimize</u> $Z = \sum_{i=1}^{2} w_i \bullet (d_i{}^- + d_i{}^+)$

Fig. 13a and b, or as the contours of $f(x)$ in Fig. 3a and b, so the approximated linear equations from multiple iterations are accumulated to replace the nonlinear equation. However, if the local convexity of an equation is below -0.015, as shown in Fig. 13c, or as the contours of $f(x)$ in Fig. 3, the equation is nonconvex, we only use a single linearized constraint in each iteration to avoid cutting off too much of the feasible space. Using accumulated constraints, designers can manage nonconvex problems accurately.

The advantages in the approximation are summarized and filled into the corresponding blank in Table 1 to become Table 9.

3.5 Advantage in formulation: minimizing deviation functions instead of objectives

We adjust the scale of $f_2(x)$ in TP-II and add the lower bounds to the two objectives to avoid negative values, so this becomes TP-III; see Table 8. We use TP-III to show how one can manage problems with large differences in the scale of multiple objectives using the cDSP and ALP, this removes Assumptions 1 and 3 and better achieves Robust Goals 1 and 2. The solutions are given in Table 10 and visualized on objective space and x-$f(x)$ Space in Figs. 14 and 15. The three optimization algorithms cannot return any feasible solutions. Due to the difference in the scale of the two objectives, one cannot use optimizing algorithms to solve the problem by linearly combining them, because: i) the objective with the largest scale dominates the other objective(s), and ii) a linearized function of the weighted-sum objective at a local area can be singular (Table 11).

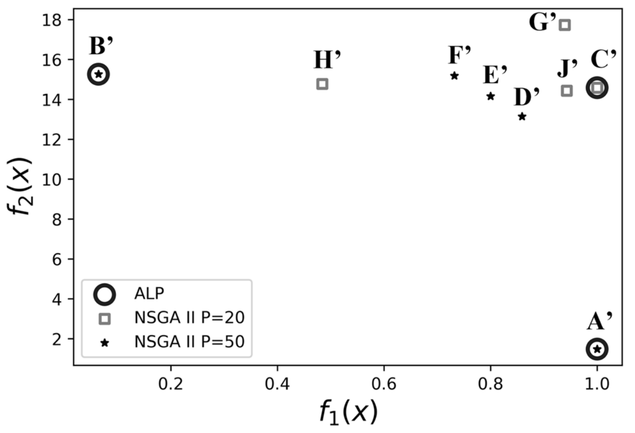


Fig. 14 The solution points to TP-III on objective space using two algorithms – solutions returned by NSGA II are more diverse but sensitive to parameter settings and increasing the population and iterations do not always produce better results

When using NSGA II to solve the problem, we use two values to the population size, 20 and 50 - one of the hyperparameters, because when we use these values, the solution returned is also changed. There are multiple hyperparameters that need to be tuned when using NSGA II. However, since we are not aiming to identify all the hyperparameters that may affect solutions and demonstrate how and how much each hyperparameter affects solutions, we only use "population" as an example to illustrate that different values of a hyperparameter have an impact on the solutions.

Observation: For a multi-objective (multi-goal) problem with nonconvex functions, when the scale of the objectives varies largely, optimizing algorithms cannot return feasible solutions, whereas a satisficing strategy may allow designers to identify adequate solutions. This is explained using the KKT.

When using optimizing algorithms to solve optimization problems, the first-order derivative of Lagrange equation with respect to decision variable x, which is a function of the parameters $\mathcal{P}$ of the model (the coefficients in objectives and constraints), decision variables x (if any objective or constraint is nonlinear), Lagrange multipliers μ and λ, and weights $\surd$ that combining the multiple goals is shown in Eq. 25.

$$\nabla_x L(x, \mu, \lambda) = y\left(\mathcal{P}, x, \mu, \lambda, \cancel{p}\right). \tag{23}$$

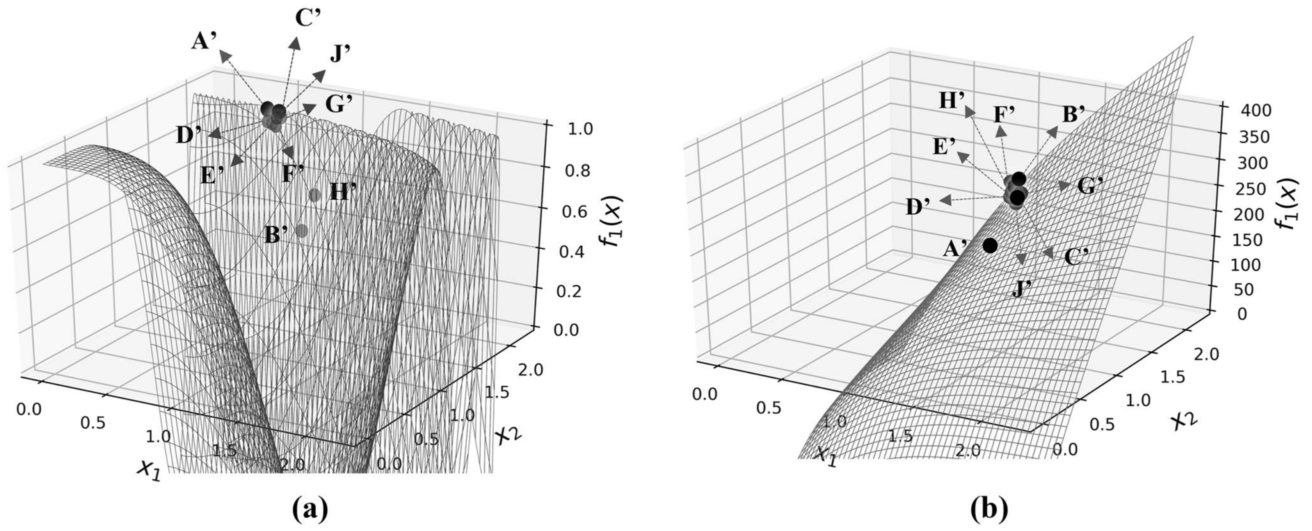


Fig. 15 The solution points to TP-III on the x-$f(x)$ space. **a** Is the 3D illustration of $f_1(x)$. **b** Is the 3D illustration of $f_2(x)$. All solutions' $f_1(x)$ values are close to 1 except B' and H'; all solutions' $f_2(x)$ values are between close to 18 except A'

For a satisficing strategy, the first-order derivative of the Lagrange equation contains only the coefficients of the deviation variables in the objective, since only deviation variables $d^{\mp}$ constitute the objective (not the decision variables, x). For a κ-goal cDSP with m inequality constraints $g(x)$ and ℓ equality constraints $h(x)$, if we use weights to combine the κ goals $G(x, d)$, i.e., using the Archimedean strategy to manage a multi-goal cDSP, then the coefficients in the first-order Lagrange equation would be only the weights and $\tau - \tau$ is the Lagrange multiplier of the goal functions; see Eq. 26.

maintaining the first-order KKT conditions for the two strategies under uncertainty varies. If any uncertainty with probability P takes place to an item $\mathfrak{I}$ in the first-order equation that destroys its equilibrium, we denote it as $Pr\left(\widetilde{\mathfrak{I}}|P\right)$. For a N-dimension, Q-parameter[8], and κ-goal problem, using optimizing strategy, the source of $\widetilde{\mathfrak{I}}$ can be decision variables $\widetilde{x_n}$, Lagrange multipliers $\widetilde{\mu_i}$ and $\widetilde{\lambda_j}$, and weights $\widetilde{\surd_k}$; for the *satisficing* strategy, the source of $\widetilde{\mathfrak{I}}$ can only be the weights $\widetilde{\surd_k}$ and the Lagrange multipliers for

$$\nabla_d L(x^s, d, \mu, \lambda, \tau) = \nabla_d z(d) + \sum_{i=1}^{m} \mu_i \nabla_d g_i(x^s) - \sum_{j=1}^{\ell} \lambda_j \nabla_d h_j(x^s) - \sum_{k=1}^{\kappa} \tau_k \nabla_d G(x^s, d) = \mathscr{g}(p, \tau). \tag{24}$$

When using optimization, the second-order Lagrange equation may still have parameters and decision variables due to nonlinearity; Eq. 27. For satisficing, the second-order Lagrange equation with respect to deviation variables degenerates to zero because the objective of a cDSP is a linear combination of deviation variables; see Eq. 28. That is why satisficing solutions do not need to meet the second-order KKT conditions.

$$\nabla^2_{xx} L(x, \mu, \lambda) = \nabla_x \mathscr{g}\left(\mathcal{P}, x, \mu, \lambda, \mathscr{p}\right) = \mathscr{g}'\left(\mathcal{P}', x\right) \tag{25}$$

$$\nabla^2_{dd} L(x^s, d, \mu, \lambda, \tau) = \nabla_d\left(\mathscr{g}(\mathscr{p}, \tau)\right) \equiv 0 \tag{26}$$

Although both optimal solutions x^* and *satisficing* solutions x^s meet the first-order KKT conditions, the chance of

the goals $\widetilde{\tau_k}$. If and only if none of the items under the uncertainty breaks the equilibrium of first-order equation, then the optimal/*satisficing* solution is still optimal/*satisficing* under this uncertainty. For a N-dimension, Q-parameter, and κ-goal problem, the probability of maintaining an optimal solution and a *satisficing* solution under Uncertainty P are given in Eqs. 29 and 30, respectively.

[8] Here, we define a coefficient or an intercept of a constraint or an objective as a parameter. A parameter has a given value (either a constant value or a stochastic value) and the value does not depend on any decision variables.

Table 11 Solutions to TP-III—dominated solutions of each weight are in *italics*; for each method, the best solution of each design scenario is marked using a capital letter (A', B', C', D', E', F', G', H', and J')

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II/III–Population (P)=20/50	
		Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$
(1, 0)	(0.5, 1) (0, 0) (2, 0.5)	Cannot manage nonconvex equations with bounds		All solutions violate one or more constraints		All solutions violate one or more constraints		A' (0.51,1.82)	1	*P=20:* *G' (0.55, 1.85)* P=50: A' (0.51, 1.82)	*P=20:* *0.99* P=50: 1
(0, 1)	(0.5, 1) (0, 0) (2, 0.5)							B' (0.51, 1.96)	15.27	*P=20:* *H' (0.52, 1.92)* P=50: B' (0.51, 1.96)	*P=20:* *14.86* P=50: 15.36
(0.5, 0.5)	(0.5, 1) (0, 0) (2, 0.5)							*C' (0.55, 1.82)*	*7.5*	*P=20:* *H' (0.52, 1.92)* P=50: D' (0.53, 1.87)	*P=20:* *7.69* P=50: 7.78
(0.7, 0.3)	(0.5, 1) (0, 0) (2, 0.5)							*C' (0.55, 1.82)*	*4.9*	*P=20:* *C' (0.55, 1.82)* P=50: E' (0.53, 1.88)	*P=20:* *4.9* P=50: 5.01
(0.3, 0.7)	(0.5, 1) (0, 0) (2, 0.5)							*C' (0.55, 1.82)*	*10.01*	*P=20:* *J' (0.54, 1.85)* P=50: F' (0.53, 1.89)	*P=20:* *10.49* P=50: 10.6

Table 12 Summary of advantages in formulation

Stage	Feature	Advantage	Assumption removed	Test problem (TP)	Robustness Goal	Introduction
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary Kuhn–Tucker conditions are met, whereas the sufficient Kuhn–Tucker conditions do not have to be met Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and/or unparameterizable uncertainties	Assumption 1	TP-III	Robustness Goal 2	Section 3.5

Table 13 The compromise DSP of the TP-IV

TP	The Compromise DSP
IV	<u>Given</u> $x_1, x_2, d_1^{\mp}, d_2^{\mp}$ $f_1(x) = \cos\left(x_1^2 + x_2^3\right)$ $f_2(x) = 25 \bullet (x_1 - 2)^3 + 50 \bullet (x_2 - 2)^3 + 50 \bullet x_1 \bullet x_2^2$ <u>Find</u> $x_1, x_2, d_1^{\mp}, d_2^{\mp}$ <u>Satisfy</u> <u>Goals:</u> $\frac{f_1(x)}{1.2} + d_1^- - d_1^+ = 1$ $\frac{f_2(x)}{400} + d_2^- - d_2^+ = 5$ <u>Constraints:</u> $x_1 \bullet x_2 \le 1$ $d_i^- \bullet d_i^+ = 0, i = 1, 2$ <u>Bounds:</u> $0 \le x_1, x_2 \le 2$ $0 \le d_1^{\mp}, d_2^{\mp} \le 1$ <u>Minimize</u> $Z = \sum_{i=1}^{2} w_i \bullet (d_i^- + d_i^+)$

$$Pr(x^*|P) \approx \prod_{q=1}^{Q}\left[1 - Pr\left(\widetilde{\mathcal{P}}_q|P\right)\right] \prod_{n=1}^{N}\left[1 - Pr\left(\widetilde{x}_n|P\right)\right] \prod_{i=1}^{m}\left[1 - \Pr\left(\widetilde{\mu}_i|P\right)\right]\prod_{j=1}^{\ell}\left[1 - \Pr\left(\widetilde{\lambda}_j|P\right)\right] \prod_{k=1}^{\kappa}\left[1 - \Pr\left(\widetilde{\surd}_k|P\right)\right], \tag{27}$$

$$Pr(x^s|P) \approx \prod_{k=1}^{\kappa}\left[1 - \Pr\left(\widetilde{\surd}_k|P\right)\right]\prod_{k=1}^{\kappa}\left[1 - \Pr\left(\widetilde{\tau}_k|P\right)\right]. \tag{28}$$

As the value of any probability is in the range of [0, 1], the more items on the right-hand side we multiply (the more items the probability depends on), the lower the probability becomes. The items in Eq. 30 are fewer than those of Eq. 29. Hence, the chance of maintaining an optimal solution under Uncertainty P is often smaller than the chance of maintaining a satisficing solution with the same uncertainty; Eq. 31.

$$\Pr(x^*|P) \le \Pr(x^s|P) \tag{29}$$

In summary, using the satisficing strategy, robustness goals 1 and 2 are better achieved for nonconvex problems with multiple objectives that have various scales; see Eqs. 32 and 33. Using satisficing, designers can deal with nonconvex, multi-objective problems that may be incomplete or inaccurate and with uncertainties, which helps remove Assumptions 1 and 3.

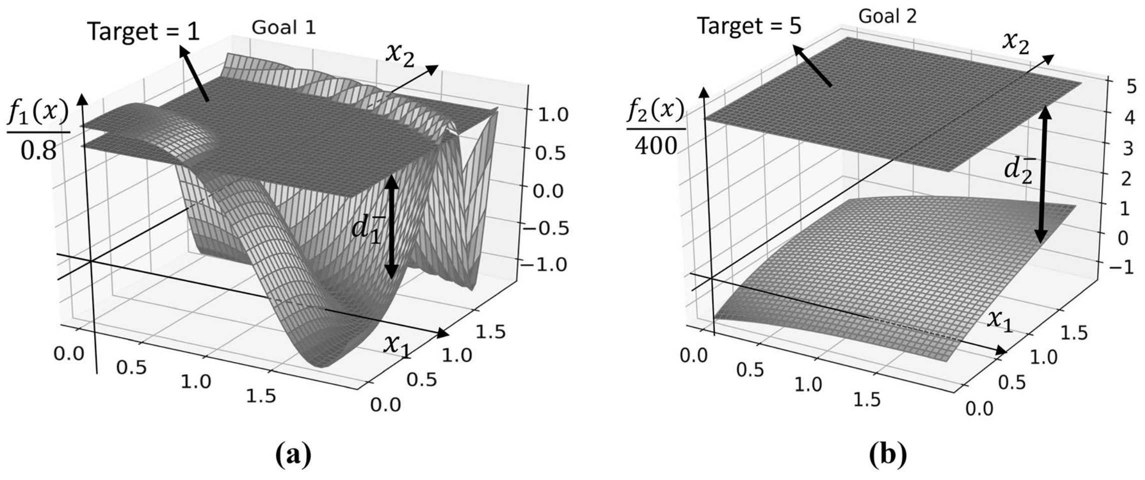


Fig. 16 The left-hand side (objective function) and the right-hand side (target) of the two goals of TP-IV on the *x*-*f*(*x*) space. **a** Is the 3D illustration of $f_1(x)$. **b** Is the 3D illustration of $f_2(x)$

$$\left[Pr\left(\mathcal{S}\neq\emptyset\middle|\left\{\begin{matrix}Nonconvexity\\ objectiveswithvariousscales\end{matrix}\right\}\right)\right]_{Satisficing}\geq$$
$$\left[Pr\left(\mathcal{S}\neq\emptyset\middle|\left\{\begin{matrix}Nonconvexity\\ objectiveswithvariousscales\end{matrix}\right\}\right)\right]_{Optimizing}, \tag{30}$$

$$\left[Pr(x^s|P)\right]_{Satisficing}\geq\left[\Pr(x^*|P)\right]_{Optimizing}. \tag{31}$$

The advantages in formulation are summarized and filled into the corresponding blank in Table 1 so it becomes Table 12.

3.6 Advantage in evaluation: allowing violations of soft requirements while avoiding violations of rigid requirements

The advantages in a formulation using a satisficing strategy are shown in Section 3.5. One may wonder if optimizing algorithms are used to solve a cDSP, will the results be the same as the ALP? The answer is no, if the problem is nonconvex and the difficulty of achieving the multiple goals varies greatly. To illustrate the advantage in evaluation using a satisficing strategy we use TP-IV—in which the difficulty of achieving each goal's target varies enormously. This allows designers to rely less on Assumption 1 and better achieve robust goal 1. We formulate TP-IV into a cDSP and use both the optimizing and satisficing strategies to solve it and compare results. The cDSP, visualization of the two goals, solutions, and visualization of solutions on objective space and *x*-*f*(*x*) space of TP-IV are given in Table 13, Fig. 16, Table 14, Figs. 17 and 18, respectively.

Observation: cDSP and ALP are designed to formulate and explore engineering design problems with various levels of achievability of the goals, in other words, when $\boldsymbol{f_i(x)/T_i} \gg \boldsymbol{f_i(x)/T_i}$ for Goal *i* and Goal *j*, which often happens in engineering design, feasible solutions can be identified.

Using the ALP to solve the cDSP, satisficing solutions (compared with the solutions from NSGA II) can be obtained, whereas using optimizing algorithms, no feasible solutions are returned. Why does using the ALP allow designers to obtain satisficing solutions to a cDSP with various degrees of achievability of the goals, but optimizing algorithms do not?

When $\boldsymbol{f_i(x)/T_i} \gg \boldsymbol{f_i(x)/T_i}$, especially when d_j^- cannot meet its upper bound of $d_j^- \leq 1$, its upper bound must be violated. However, optimization algorithms do not have a mechanism to violate the deviation bounds but not to violate other constraints or bounds. Because some optimization algorithms, such as Trust- constr and SLSQP, treat all constraint and bound priorities equally, if at least one of the constraints and bounds is violated, the solution point is considered to be infeasible.

Unlike those optimizing algorithms, using the ALP, the constraints, and the bounds of the decision variables *x* are the highest priority, and the bounds of the deviation variables $d_i^{\pm}$ are assigned second priority. If the violation of any deviation bound allows a point x^s in the feasible area bounded by the constraints and the bounds of system variables to be found, then X^S is returned as a solution. So, for an *n*-dimension, *m*-constraint (with *p* inequality constraints and *m-p* equality constraints), *k*-goal cDSP: x^s is a satisficing solution,

Table 14 Solutions to TP-IV (close-to-non-dominated solutions or good enough solutions)

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II – Population (P)=50	
		Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$	Solution	$\sum_{i=1}^{2} w_i \bullet f_i(x)$
(1, 0)	(0.5, 1)	Cannot manage nonconvex equations with bounds		All solutions violate one or more constraints		All solutions violate one or more constraints		*A'' (0.53,1.75)*	*0.8*	D'' (0.51, 1.88)	0.81
	(0, 0)										
	(2, 0.5)										
(0, 1)	(0.5, 1)							B'' (0.51, 1.97)	16.26	*E'' (0.51, 1.96)*	*15.26*
	(0, 0)										
	(2, 0.5)										
(0.5, 0.5)	(0.5, 1)							C'' (0.57, 1.74)	7.29	*F'' (0.53, 1.88)*	*7.21*
	(0, 0)										
	(2, 0.5)										
(0.7, 0.3)	(0.5, 1)							C'' (0.57, 1.74)	4.69	*G'' (0.57, 1.75)*	*4.6*
	(0, 0)										
	(2, 0.5)										
(0.3, 0.7)	(0.5, 1)							*C'' (0.57, 1.74)*	*9.88*	F'' (0.53, 1.88)	10.54
	(0, 0)										
	(2, 0.5)										

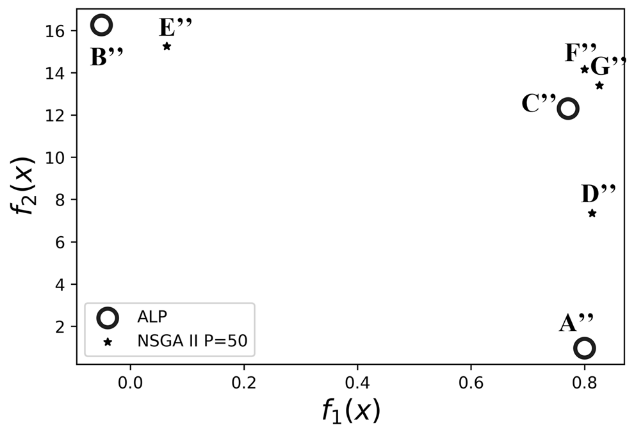


Fig. 17 The solutions to TP-IV using the ALP and NSGA II—NSGA II finds more non-dominated solutions, whereas ALP finds solutions with better Z values (the weighted combination of deviations)

if and only if

$\frac{f_i(x^s)}{T_i} + d_i^- - d_i^+ = 1, \forall i = 1, \ldots k$ //Goal functions hold at x^s, ***and***

$g_i(x^s) \geq 0, \forall i = 1, \ldots p$//Inequality constraints are satisfied at x^s, ***and***

$h_i(x^s) = 0, \forall i = p + 1, \ldots m$//Equality constraints are satisfied at x^s, ***and***

$lowerbound \leq x^s \leq upperbound$, ***and***

Minimize the violation of deviation bounds

In summary, for multi-objective, engineering design problems with nonlinear, nonconvex functions, goals with various scales, and the target of the goals with various degrees of achievability, using the cDSP and ALP allows designers to obtain satisficing solutions that are close to the non-dominated solutions obtained by using NSGA II; see Eq. 34. Multi-objective problems can be designed, especially when the scale of the goals varies largely, which helps remove Assumptions 1 and 3.

$$\left[Pr\left(\mathcal{S} \neq \emptyset \middle| \left\{ \begin{array}{c} Nonconvexity \\ Goalswithvariousscales \\ Goals' targetswithvariousachievabilities \end{array} \right\} \right) \right]_{Satisficing} \geq \left[Pr\left(\mathcal{S} \neq \emptyset \middle| \left\{ \begin{array}{c} Nonconvexity \\ Goalswithvariousscales \\ Goals' targetswithvariousachievabilities \end{array} \right\} \right) \right]_{Optimizing} . \tag{32}$$

The advantages in the formulation are summarized and filled into the corresponding blank in Table 1 so it becomes Table 15.

We demonstrated the advantages of the cDSP and ALP in the four stages using four test problems. We summarize them by adding information to Table 1, so it becomes Table 16.

4 Computational efficiency

How does the computational efficiency of the ALP compare to the simplex algorithm and interior-point searching algorithms such as NSGA II?

The ALP algorithm is used for linearizing the compromise Decision Support Problems (cDSP) through second-order sequential linearization and solving them with the revised dual simplex algorithm. Second-order sequential linearization requires multiple iterations. In each iteration, the nonlinear equation is linearized. Nonlinear constraints

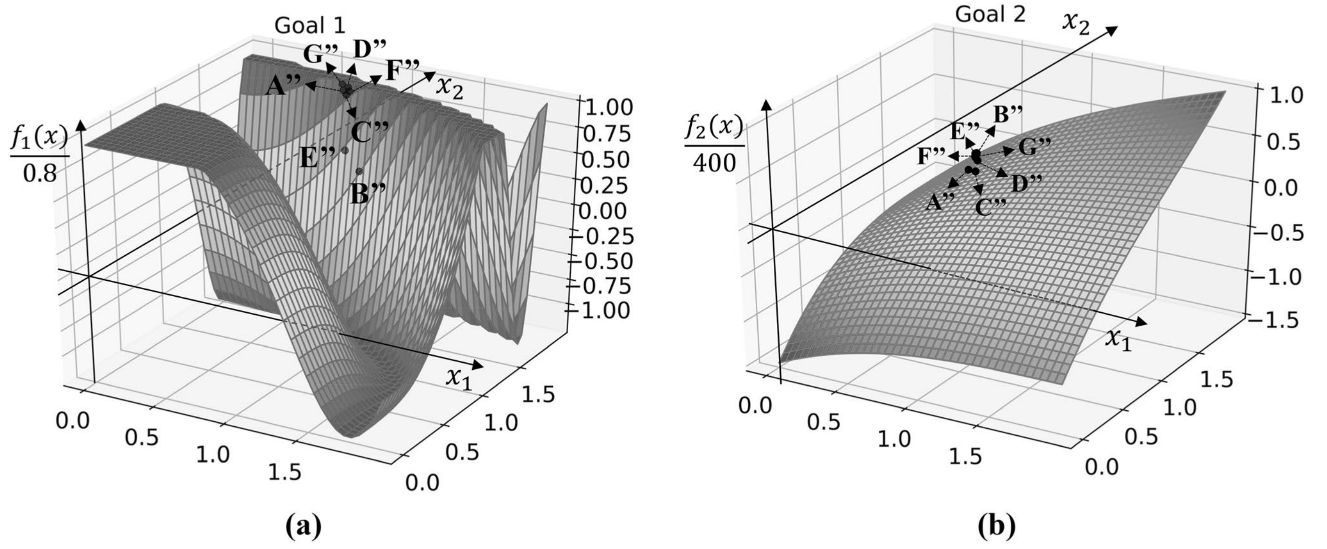


Fig. 18 The solutions to TP-IV on the x-$f(x)$ space—there is little difference between ALP solutions and NSGA II solutions (the arrows are used to identify the solutions and their directions have no meaning). **a** Is the 3D illustration of $f_1(x)$. **b** Is the 3D illustration of $f_2(x)$

Table 15 Summary of advantages in evaluation

Stage	Feature	Advantage	Assumption removed	Test problem (TP)	Robustness goal	Introduction
Evaluation	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility As a result, goals and constraints with different scale can be managed	Assumption 1 and 3	TP-IV	Robustness Goal 1	Section 3.6

and nonlinear goals are included in a second-order equation by using the diagonal terms of the Hessian matrix of the equation at the linearization point as coefficients. This is used to solve the second-order equation and the solution with the lower absolute value is selected. Using this value and the linearization point as two points to determine a line, this line then becomes the linear equation. Then the problem becomes linear and is solved with the revised dual simplex algorithm. Then, the solution point becomes the linearization point for the next iteration. The linear equations in each iteration are accumulated as linear equations in the linear problem in the latest iteration. The algorithm converges when the difference among the solutions of several iterations is smaller than a predefined threshold, or when the number of iterations reaches the predefined maximum number of iterations.

Suppose there are m equations in an n-dimensional design problem, and $m\prime$ of the equations are nonlinear. In Iteration i, the computational complexity of linearizing the $m\prime$ equations into second-order equations is $O(m\prime \bullet n)$. Suppose that there are m_i'' linear equations – including the $(m - m\prime)$ equations that are linear in the original design problem and $(m_i'' - m + m\prime)$ equations that are linearized and accumulated from Iteration one to the current iteration, so the computational complexity of solving the linearized problem in Iteration i is polynomial on average (Kelner and Spielman 2006) although the worst-case is $O(2^n)$. Therefore, the computational complexity of the ALP is polynomial.

As the computational complexity of the simplex algorithm is polynomial for the average-case (Spielman and Teng 2004), the ALP has the same computational complexity as the Simplex algorithm. The computational complexity of NSGA II is $O(k \bullet N^2)$, where k is the number of objective functions and N is the population size. The computational complexity of ALP and NSGA II depend on different parameters—the former depends on the dimension and the latter depends on the population size. Thus the computational cost of the ALP does not exceed that of NSGA II. As users usually set a population much larger than the dimension of a problem when using NSGA II, more often than not, NSGA II requires greater computational power than the ALP does. (Spielman and Teng 2004).

5 Closing remarks

In this paper, in the context of the assumptions that are foundational to the KKT conditions we describe the differences between adopting an optimizing strategy and a satisficing strategy when design complex engineering systems; see Section 2.4 To use the optimizing strategy, to obtain optimal solutions, three assumptions are foundational to meet

Table 16 The advantages of the cDSP and ALP in the four stages of engineering design

Stage	Feature	Advantage	Assumption removed	Test problem (TP)	Robustness goal	Introduction
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary Kuhn–Tucker conditions are met, whereas the sufficient Kuhn–Tucker conditions do not have to be met Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and/or unparameterizable uncertainties	Assumption 1	TP-III	Robustness Goal 2	Section 3.5
Approximation	Using second-order sequential linearization	Designers can have a balance between linearization accuracy and computational complexity	Assumption 3	TP-II	Robustness Goal 1 and 3	Section 3.4
	Using accumulated linearization	Designers can manage nonconvex problems and deal with highly convex, nonlinear problems relatively more accurately				
Exploration	Combining interior-point search and vertex search	Designers can avoid getting trapped in local optima to some extent and identify satisficing solutions which are relatively insensitive when the starting points changing	Assumption 1	TP-I	Robustness Goal 3	Section 3.3
Evaluation	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility As a result, goals and constraints with different scale can be managed	Assumption 1 and 3	TP-IV	Robustness Goal 1	Section 3.6

both the necessary and sufficient KKT conditions. To use the satisficing strategy, to obtain satisficing solutions, only the necessary KKT condition is satisfied. We use four test problems to illustrate the different outcomes between the two strategies regarding three robustness goals (Section 2.5), namely, the chance of identifying satisficing solutions given certain complexities, the probability of maintaining a satisficing solution under uncertainties that result in feasible space boundary change, and the diversity of solutions under multiple design scenarios. Given the characteristics of engineering design problems—often having nonlinear, nonconvex goals and constraints, goals which may have different units, and the targets of goals may have different degrees of achievability, we find a satisficing strategy is more robust with respect to our three robustness goals. We verify the satisficing solutions by comparing them with NSGA II solutions. Thence, we conclude that adopting a satisficing strategy more robust and practical for use in model-based engineering design. To reinforce our conclusion, we offer the following references for consideration: Gautham et al. 2017; Nellippallil et al 2018; Pedersen et al. 2013; Samadiani et al. 2010; Simpson et al. 2001.

Acknowledgements Lin Guo and Janet K. Allen gratefully acknowledge the financial support from the John and Mary Moore Chair at the University of Oklahoma. Lin Guo and Farrokh Mistree gratefully acknowledge financial support from the L.A. Comp Chair at the University of Oklahoma. Lin Guo acknowledges the financial support from the Pietz Professorship and Start-Up Fund at the South Dakota School of Mines and Technology

Author contributions This paper is based on Lin Guo's Ph.D. Dissertation. Farrokh Mistree and Janet K. Allen mentored her, supervised her, and funded her. All three authors have contributed to what has been submitted

Data availability The data supporting the findings of this study are available within the article.

Declarations

Conflict of interest The authors declare no competing interests.

References

Arrow KJ (2012) Social choice and individual values, 3rd edn. Yale University Press, New Haven

Beyer H-G, Sendhoff B (2007) Robust optimization–a comprehensive survey. Comput Methods Appl Mech Eng 196(33–34):3190–3218

Box GE (1979) All models are wrong, but some are useful. Robust Stat 202(1979):549

Byron M (1998) Satisficing and optimality. Ethics 109(1):67–93

Carlson JM, Doyle J (2002) Complexity and robustness. Proc Natl Acad Sci 99(suppl 1):2538–2545

Charnes A, Cooper WW, Ferguson RO (1955) Optimal estimation of executive compensation by linear programming. Manag Sci 1(2):138–151

Chen W, Allen JK, Tsui K-L, Mistree F (1996) A procedure for robust design: minimizing variations caused by noise factors and control factors. ASME J Mech Des 118(4):478–485

Chen W, Allen JK, Mistree F (2000) Design knowledge development techniques and applications in productivity enhancement in concurrent systems design. In: Leondes CT (ed) Knowledge based systems techniques and applications. Academic Press, San Diego, pp 1037–1060

Choi H-J, Austin R, Allen JK, McDowell DL, Mistree F, Benson DJ (2005) An approach for robust design of reactive powder metal mixtures based on non-deterministic micro-scale shock simulation. J Comput Aided Mater Des 12:57–85

Courant R, Hilbert D (1953) Methods of mathematical physics. Interscience, New York

Dantzig GB (1990) Origins of the simplex method. Hist Sci Comput 5:141–151

Dantzig GB, Orden A, Wolfe P (1955) The generalized simplex method for minimizing a linear form under linear inequality restraints. Pac J Math 5(2):183–195

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Euler L (1755) Letter to Lagrange. Opera S. Iva 5:375–478

Euler L (1766) Eclaircissemens sur le Mouvement des Cordes Vibrantes. Melanges de Philosophie et de la Mathematique de la Societe Royale de Turin, pp 1–26

Farhang-Mehr A, Azarm S (2002) Diversity assessment of pareto optimal solution sets: an entropy approach. Proc Congr Evolut Comput 1:723–728

Fletcher R (1987) Practical methods of optimization, 2nd edn. John Wiley and Sons, Hoboken

Frey DD, Herder PM, Wijnia Y, Subrahmanian E, Katsikopoulos K, Clausing DP (2009) The Pugh controlled convergence method: model-based evaluation and implications for design theory. Res Eng Design 20(1):41–58 https://doi.org/10.1007/s00163-008-0056-z

Gautham BP, Kulkarni NH, Panchal JH, Allen JK, Mistree F (2017) A method for the preliminary design of gears using a reduced number of American gear manufacturers association (AGMA) correction factors. Eng Optim 49(4):565–582

Guo L (2021) Model evolution for the realization of complex systems. The University of Oklahoma

Guo L, Chen S, Allen JK, Mistree F (2021) A framework for designing the customer-order decoupling point to facilitate mass customization. ASME J Mech Des 143(2):022002

Guo L, Milisavljevic-Syed J, Wang R, Huang Y, Allen JK, Mistree F (2022) Managing multi-goal design problems using adaptive leveling-weighting-clustering algorithm. Res Eng Design 34:39–60

Guo L, Chen S (2023) Satisficing strategy in engineering design. In: International Design Engineering Technical Conferences and

Computers and Information in Engineering Conference, Boston, MA, August 20-23, 2023, Paper Number. DETC2023-109302

Hajihasemi S (2023) A user friendly wrapper for DSIDES (Decision Support in the Design of Engineering Systems), MS Thesis, University of Oklahoma

Hartley H (1960) Studies in linear and non-linear programming. JSTOR 55(292):758–760

Hazelrigg GA (2010) The Pugh controlled convergence method: model-based evaluation and implications for design theory. Res Eng Des 21(3):143–144. https://doi.org/10.1007/s00163-010-0087-0

Ignizio JP (1985) Multiobjective mathematical programming via the multiplex model and algorithm. Eur J Oper Res 22(3):338–346

Jahn J (1985) Scalarization in multi objective optimization. Mathematics of multi objective optimization international centre for mechanical sciences. Springer, Vienna, pp 45–88

Karush W (1939) Minima of functions of several variables with inequalities as side constraints. University of Chicago, Master's Thesis.

Kelner JA, Spielman DA (2006) A randomized polynomial-time simplex algorithm for linear programming. In: Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing, Seattle, pp 51–60

Khosla P, Rubin S (1981) A conjugate gradient iterative method. Comput Fluids 9(2):109–121

Kuhn HW, Tucker A (1951) Nonlinear programming. Proceedings of the second Berkeley symposium on mathematical statistics and probability. University of California Press

Lange K (2013) Optimization. Springer, New York

Lin JG (1976) Maximal vectors and multi-objective optimization. J Optim Theory Appl 18(1):41–64

Magni J-F, Bennani S, Terlouw J (1997) Robust flight control: a design challenge. Springer, Berlin

McDowell DL, Panchal JH, Choi H-J, Seepersad CC, Allen JK, Mistree F (2010) Integrated design of multiscale materials and products. Elsevier, New York

Messac A, Mattson CA (2002) Generating well-distributed sets of pareto points for engineering design using physical programming. Optim Eng 3(4):431–450

Messer M, Panchal JH, Krishnamurthy C, Klein B, Yoder PD, Allen JK, Mistree F (2010) Model selection under limited information using a value-of-information-based indicator. ASME J Mech Des 132(12):1210008

Milisavljevic-Syed J, Allen JK, Commuri S, Mistree F (2020) Architecting networked engineered systems. Springer, Cham

Ming Z, Nellippallil AB, Yan Y, Wang G, Goh CH, Allen JK, Mistree F (2018) PDSIDES—a knowledge-based platform for decision support in the design of engineering systems. J Comput Inf Sci Eng 18(4):041001. https://doi.org/10.1115/1.4040461

Mistree F, Hughes O, Phuoc H (1981) An optimization method for the design of large, highly constrained complex systems. Eng Optim 5(3):179–197

Mistree F, Smith WF, Bras B, Allen JK, Muster D (1990) Decision-based design: a contemporary paradigm for ship design. Trans Soc Naval Archit Mar Eng 98:565–597

Mistree F, Hughes OF, Bras B (1993) Compromise decision support problem and the adaptive linear programming algorithm. Progr Astron Aeronaut 150:251

Nash SG (1984) Newton-type minimization via the lanczos method. SIAM J Numer Anal 21(4):770–788

Nellippallil AB, Song KN, Goh C-H, Zagade P, Gautham B, Allen JK, Mistree F (2017) A goal-oriented, sequential, inverse design method for the horizontal integration of a multistage hot rod rolling system. ASME J Mech Des 139(3):031403

Nellippallil AB, Rangaraj V, Gautham B, Singh AK, Allen JK, Mistree F (2018) An inverse, decision-based design method for integrated design exploration of materials, products, and manufacturing processes. ASME J Mech Des 140(11):111403

Nellippallil AB, Ming Z, Allen JK, Mistree F (2019) Cloud-based materials and product realization: fostering ICME via industry 4.0. Integr Mater Manuf Innov 8(2):107–121

Nellippallil AB, Mohan P, Allen JK, Mistree F (2020) An inverse, robust design method for robust concept exploration. ASME J Mech Des 142(8):081703

Panchal JH, Fernández MG, Allen JK, Paredis CJJ, Mistree F (2005) Facilitating meta-design via separation of problem product and process information. ASME International Mechanical Engineering Congress and Exposition, Orlando

Pedersen K, Messer M, Allen JK, Mistree F (2013) Hierarchical product platform design: a domain-independent approach. Ships off-Shore Struct 8(2):367–382

Powell MJ (1964) An efficient method for finding the minimum of a function of several variables without calculating derivatives. Comput J 7(2):155–162

Powell M (1989) A tolerant algorithm for linearly constrained optimization calculations. Math Progr 45(1):547–566

Powell MJ (2007) A view of algorithms for optimization without derivatives. Math Today Bull Inst Math Appl 43(5):170–174

Reich D, Green RE, Kircher M, Krause J, Patterson N, Durand EY, Viola B, Briggs AW, Stenzel U, Johnson PLF, Maricic T, Good JM, Marques-Bonet T, Alkan C, Fu Q, Mallick S, Li H, Meyer M, Eichler EE, Stoneking M, Richards M, Talamo S, Shunkov MV, Derevianko AP, Hublin J-J, Kelso J, Slatkin M, Pääbo S (2010) Genetic history of an archaic hominin group from Denisova Cave in Siberia. Nature 468(7327):1053–1060. https://doi.org/10.1038/nature09710

Rezapour S, Khosrojerdi A, Rasoulifar G, Allen JK, Panchal JH, Srinivasan RS, Tew JD, Mistree F (2018) Architecting fail-safe supply networks. CRC Press, Boca Raton

Saaty TL (1994) Highlights and critical points in the theory and application of the Analytic Hierarchy Process. Eur J Oper Res 74(3):426–447. https://doi.org/10.1016/0377-2217(94)90222-4

Samadiani E, Joshi Y, Allen JK, Mistree F (2010) Adaptable robust design of multi-scale convective systems applied to energy efficient data centers. Numer Heat Transfer Part A 57(2):69–100

Sen A (2018) Collective choice and social welfare. Harvard University Press

Simon HA (1956) Rational choice and the structure of the environment. Psychol Rev 63(2):129

Simon HA (1996) The sciences of the artificial. MIT Press, Cambridge

Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. Res Eng Design 13:2–22

Sinha A, Bera N, Allen JK, Panchal JH, Mistree F (2013) Uncertainty management in the design of multiscale systems. ASME J Mech Des 135(1):0011008

Soltani AR, Tawfik H, Goulermas JY, Fernando T (2002) Path planning in construction sites: performance evaluation of the Dijkstra, A∗, and GA search algorithms. Adv Eng Inform 16(4):291–303

Spielman DA, Teng S-H (2004) Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. JACM 51(3):385–463

Straeter TA (1971) On the extension of the davidon-broyden class of rank one, quasi-newton minimization methods to an infinite dimensional hilbert space with applications to optimal control problems, PhD Dissertation in Mathematics, North Carolin State University.

Vincent TL (1983) Game theory as a design tool. J Mech Transm Autom Des 105(2):165–170. https://doi.org/10.1115/1.3258503

Wu CJ, Hamada MS (2011) Experiments: planning, analysis, and optimization. John Wiley and Sons, Hoboken

Xu Y, Zhang H, Zeng X, Nojima Y (2022) An adaptive convergence enhanced evolutionary algorithm for many-objective optimization problems. Swarm Evol Comput 75:101180

Zhu C, Byrd RH, Lu P, Nocedal J (1997) Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans Math Softw (TOMS) 23(4):550–560