

Safe Learning and Optimization Techniques: Towards a Survey of the State of the Art

Youngmin Kim¹[0000–0002–7699–6532], Richard Allmendinger¹[0000–0003–1236–3143],
and Manuel López-Ibáñez^{1,2}[0000–0001–9974–1295]

¹ Alliance Manchester Business School, University of Manchester,
Manchester M15 6PB, UK

² School of Computer Science, University of Málaga, 29071 Málaga, Spain
 {youngmin.kim,richard.allmendinger,manuel.lopez-ibanez}@manchester.ac.uk

Abstract Safe learning and optimization deals with learning and optimization problems that avoid, as much as possible, the evaluation of non-safe input points, which are solutions, policies, or strategies that cause an irrecoverable loss (e.g., breakage of a machine or equipment, or life threat). Although a comprehensive survey of safe reinforcement learning algorithms was published in 2015, a number of new algorithms have been proposed thereafter, and related works in active learning and in optimization were not considered. This paper reviews those algorithms from a number of domains including reinforcement learning, Gaussian process regression and classification, evolutionary computing, and active learning. We provide the fundamental concepts on which the reviewed algorithms are based and a characterization of the individual algorithms. We conclude by explaining how the algorithms are connected and suggestions for future research.

Keywords: safe learning · safe optimization · Markov decision process · black-box optimization · expensive optimization

1 Introduction

Standard learning and optimization algorithms are generally concerned with trading off exploration and exploitation of an objective function such that approximation of the objective function is performed efficiently for learning problems, and such that an optimal solution, policy and/or strategy is discovered within as few evaluations as possible for optimization problems. This paper is concerned with learning/optimization problems where the evaluation of *unsafe* solutions imply some significant loss, such as damage of experimental equipment or personal injury.

Safe learning and optimization scenarios typically arise in black-box problems, particularly expensive ones. In *black-box* problems, no explicit mathematical model

Please cite as: Kim Y., Allmendinger R., López-Ibáñez M. (2021) Safe Learning and Optimization Techniques: Towards a Survey of the State of the Art. In: Heintz F., Milano M., O’Sullivan B. (eds) Trustworthy AI - Integrating Learning, Optimization and Reasoning. TAILOR 2020. Lecture Notes in Computer Science, vol 12641. Springer, Cham. The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-73959-1_12.

of the safety constraints is available and the value of the safety constraint function can only be known after a solution has been evaluated. If the number of total evaluations is very limited, from ten to a few thousands, due to limited time or available resources, then the problem is also called *expensive*. Such scenarios arise often when the evaluation of a solution requires a simulation [11] or a real-world experiment, as in closed-loop optimization [1, 19]. This paper provides an overview of the research carried out in the area of safe learning and optimization, primarily, for black-box and expensive problems.

Algorithms designed for solving expensive (black-box) problems exploit information obtained through a series of expensive evaluations to select the next input point (i.e., solution) for evaluation. In such problems, an evaluation is akin to executing a physical, chemical or biological experiment, and thus, involves the use of resources, such as raw materials, machines, operators, etc. Problems that require time-consuming computer simulations can be seen as another example of expensive problems [1, 4, 11, 23, 28]. As mentioned above, in expensive problems, the evaluation of an unsafe solution can lead to a waste of resources, such as damage/loss of equipment, of which we may have limited availability.

A large body of research has been carried out around algorithm design for expensive problems. Arguably, Bayesian optimization (also called surrogate-assisted optimization, meta-model and response surface methods) [12, 29] has become the default approach for tackling expensive optimization problems. However, research around non-standard problem features and their implications, such as safety, fairness, and dynamic problem aspects, remains dispersed across the different areas of machine learning, AI and optimization.

García and Fernández [13] provided a survey on safe reinforcement learning (safe RL), including *constrained criterion*-based optimization, which is relevant to this survey. In *constrained criterion*-based optimization, the safe domain of policies (i.e., of input points) is approximated by several constraints. There are several approaches in *constrained criterion*-based optimization. The most typical approach constraints a safety function above a certain threshold. Other approaches evaluate only input points that preserve *ergodicity* [13, 21] or ensure that evaluations are only allowed when the expected variance of their output does not exceed a certain threshold [13]. Classical *ergodicity* means that we can get from every state to every other state with positive probability. However, in the context of safe RL, the *ergodicity* assumption differentiates between safe and unsafe states, and the concepts of reachability (an input point is safe only if it can be reached without evaluating any unsafe input points) and returnability (an input point is safe if there is a sequence of transitions via safe input points that reaches any of a given safe state set) are limited to being able to move between safe states only [33–35]; hence, in the remainder of this paper, *ergodicity* refers to this notion from safe RL.

In this paper, we define a safe learning and optimization problem as one subject to one or more safety constraints and/or the preservation of ergodicity as defined above. An input point that violates a safety constraint or ergodicity is deemed *unsafe*. The goal when tackling such problems may be to identify an optimal input point, learn some unknown function, explore a search space or determine the bound-

aries of the safe input space, while constraining the evaluation of unsafe input points up to a maximum budget, which is often zero if no unsafe evaluations are allowed at all. Thus, we review algorithms that aim to satisfy all safety constraints and/or preserve ergodicity, the latter mostly applies to safe RL algorithms designed for learning tasks. Furthermore, we review several related algorithms in active learning and optimization communities such as Bayesian optimization and evolutionary computing, which were not covered in [13].

The remainder of the paper is organized as follows. In Section 2, we propose a general formal definition of safe optimization problems that generalizes many of the formulations found in the literature, and discuss the scope of this survey. Section 3 provides a brief summary of several fundamental concepts related to safe learning and optimization. Section 4 reviews existing safe learning and optimization algorithms that have been proposed after the survey by García and Fernández [13], originating from evolutionary computing, active learning, reinforcement learning and Bayesian optimization. Finally, Section 5 discusses links between these algorithms and provides ideas for future research.

2 Problem statement

This section provides a formal definition of a general safe optimization problem and discusses various special cases.

In a safe optimization problem, we are given an objective function $f: \mathbb{R}^m \rightarrow \mathbb{R}$ that is typically both black-box and expensive (e.g., costly, time-consuming, resource-intensive etc.). The goal is to discover a feasible and safe m -dimensional input point $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{R}^m$ that maximizes the objective function $f(\mathbf{x})$ while avoiding the evaluation of *unsafe* input points as much as possible. The objective function may represent reward, efficiency or cost, and input points may represent policies, strategies, states/actions (of an agent/system) or solutions. Formally,

$$\begin{aligned} & \text{Maximize} && f(\mathbf{x}) && \mathbf{x} \in \mathbb{R}^m \\ & \text{subject to} && g_i(\mathbf{x}) \leq 0 && i = 1, \dots, q \\ & && s_j(\mathbf{x}) \geq h_j && j = 1, \dots, p, \end{aligned} \tag{1}$$

where $g_i(\mathbf{x}) \leq 0$ are q *feasibility constraints*, $s_j(\mathbf{x})$ are p *safety functions* [14] and h_j are p *safety thresholds* (constants). We explicitly separate the safety thresholds from the safety functions since, depending on the application at hand, either the function is black-box [30] or the threshold is unknown *a priori* [26], meaning we only know whether a solution is safe during or after its evaluation. A common special case arises when the safety function is the same as the objective function, that is, there is only one safety constraint ($p = 1$) such that $s_1(\mathbf{x}) = f(\mathbf{x}) \geq h$ [2, 30, 33].

There is a key difference between (hard and soft) feasibility and safety constraints: Feasibility constraints model aspects that are relevant for an input point solution to be of practical use, such as bounds of instrument settings, physical limitations or design requirements. Depending on the application, an input point that violates a feasibility constraint can or cannot be evaluated (hard vs soft constraints) [20] but if a

feasibility constraint is violated by a solution, then this has no serious consequences. On the other hand, evaluating an input point that violates a safety constraint leads to an unsafe situation, e.g., an experimental kit breaks or a human dies. That is, prior to evaluating a solution, an optimizer needs to have as much certainty about its safety status as possible, based either on a continuous measure [18] or binary property of solutions. It is important to note that an input point can be feasible but unsafe, or vice versa.

An input point that violates any safety constraint is *unsafe*. The evaluation of an unsafe input point is counted as a *failure*. Let us assume that a given budget of failures is allowed, $n^{\text{failure}} \in \mathbb{N}_0$. After this budget is consumed, the algorithm has failed and cannot continue. We can distinguish two special cases in the literature. In one case, evaluating *unsafe* input points encountered during the search process is endurable only a limited number of times [2, 4, 18, 23–26]; thus, $n^{\text{failure}} > 0$. When a *failure* represents a relatively innocuous event, such as the crash of an expensive simulation [4, 23], a relatively large number of failures may be allowed, but each of them has a cost. However, in other cases, the assumption is that no *unsafe* input point should be evaluated ever ($n^{\text{failure}} = 0$) [6–8, 10, 30, 31, 33, 35].

In some safe optimization and learning problems, a safety constraint can be violated if the associated safety function $s_j(\mathbf{x})$ cannot be evaluated because the unsafe input point represents an incomplete experiment or expensive simulation [4, 23] or physical damage to the input point [2] that prevents measuring the value of the safety function. Thus, it is only known whether the constraint was violated, but not by how much. Within the above definition of safe optimization problem (Eq. 1), such cases are equivalent to the following safety constraint:

$$s'_j(\mathbf{x}) \geq 0 \quad \text{where} \quad s'_j(\mathbf{x}) = \begin{cases} s_j(\mathbf{x}) & \text{if } s_j(\mathbf{x}) \geq h_j \text{ (safe)} \\ -1 & \text{if } s_j(\mathbf{x}) < h_j \text{ (unsafe)} \end{cases} \quad (2)$$

where only the value $s'_j(\mathbf{x})$ is observable, whereas $s_j(\mathbf{x})$ is not directly observable. Moreover, in some contexts [4], the evaluation of an unsafe input point may also prevent the objective function (or feasibility constraints) to be evaluated, even if different from the safety function.

The above optimization model also covers optimal parameter control, if the problem may be modelled as online optimization or multi-arm bandit [7]. Furthermore, the above model can be adapted easily to safe learning and RL. In safe learning, the goal becomes to discover a feasible and safe input point that minimizes the largest amount of uncertainty. Thus, the objective function may represent uncertainty about a performance or safety function. Uncertainty may be measured in different ways, e.g., variance and width of confidence interval. In the particular case of safe RL, a Markov decision process (MDP) is typically used for modelling the problem, such that an agent (e.g., rover or quadrotor) needs to explore the state or state-action space in an uncertain environment [8, 33–35]. The sequence of input points (states or state-action pairs) is determined by a transition function, which is often unknown. In this context, a safe input point must also satisfy *ergodicity*, i.e., it needs to satisfy the properties of reachability and returnability to a safe state [33–35]. Here, returnability is allowed to be met in n steps, however, reachability is problem-specific, as

we will explain in more detail later. In the context of MDP, the goal may be to optimize a reward function [15, 35] or to find the largest set of safe input points (*safe exploration*) [8, 33, 34]. In either case, the objective function in Eq. (1) may be adapted to reflect those goals.

3 Fundamental concepts in safe learning and optimization

Safe learning and optimization is a domain that is spread across several research communities, each approaching the problem from a different angle, though there are overlaps in methodologies. This section provides an overview of the main concepts underpinning the different methodologies including Markov decision process (MDP), Gaussian process regression (GPR), Evolutionary algorithm (EA), *safe set operator* (which examines safety of input points based on L -Lipschitz continuity), and *optimistic expander operator* and *optimistic safe set operator* (which is operated based on a safe set established by *safe set operator* followed by naive confidence interval and intersectional confidence interval applied to the operators). In addition, **classifiers**, such as Gaussian process classifier (GPC) [4, 26] and least-squares support vector machine (LS-SVM) [23], have been used for inferring the safety of input points, however, we do not introduce them here for brevity.

We distinguish between function exploration in safe learning and safe **MDP** exploration in RL. In function exploration, the goal is to learn some unknown function, while avoiding unsafe input points. By contrast, in safe MDP exploration, an agent explores the state space of the MDP. Hence, the problem definition includes a reachability constraint, which ensures that safe input points are reachable in one step of the MDP (one-step reachability [33, 35]) or in several steps (n -step reachability [34]), and a returnability constraint, which ensures that the agent can return to any of a given safe state set, usually, in several steps (n -step returnability [8, 33–35]). Details about RL and MDPs can be found in [32].

GPR is a regression method that learns a function using Gaussian processes. GPR can be used for function exploration, exploitation and exploration-exploitation [3, 27]. Following most recent research in safe learning and optimization, we focus here on learning (function exploration) and optimization (function exploration-exploitation) subject to safety constraints, which we denote as safe learning and safe optimization, respectively. Arguably, the majority of safe learning and optimization approaches make use of GPR in one way or the other, primarily to infer (i) an objective function, as generally used in expensive learning/optimization, and/or (ii) safety function(s), which provides information about the likelihood of an input point being safe. For general information about GPR, the reader is referred to [9, 22, 27].

The literature also reports some applications of **Evolutionary Algorithms (EAs)** [5] to safe optimization problems. EAs are heuristic optimization methods inspired by biological evolution. Loosely defined, EAs evolve a population of individuals (solutions) through the application of variation operators (mutation and/or crossover) and selection of the fittest. Naive variation operators involve a great deal of (guided) randomness to generate innovative solutions and thus cover a larger part of the search space. Moreover, naive EAs do not account for the expected mean and

uncertainty of a solution before evaluating it, unlike GPR. The combination of being too innovative and unconscious about the safety of a solution prior to evaluation may explain their limited application to problems with safety constraints.

In the literature on safe learning and optimization, two kinds of methods are mostly used for determining the level of safety of input points: L -Lipschitz continuity and classifiers. The concept of **L -Lipschitz continuity** is often used in combination with problems where a known constant safety threshold h is available to the learning/optimization algorithm. An evaluation that yields an output value above the threshold would deem the input point as safe. More formally, given a Lipschitz constant L , the L -Lipschitz continuity assumption is met if

$$d(f(\mathbf{x}), f(\mathbf{x}')) \leq L \cdot d(\mathbf{x}, \mathbf{x}'), \quad (3)$$

where both \mathbf{x} and $\mathbf{x}' \in \mathbb{R}^m$, and $d(\cdot, \cdot)$ denotes the distance between two input points, typically, the Euclidean distance [8]. Then, given a set of safe input points $S \subset \mathbb{R}^m$, an input point $\mathbf{x} \in \mathbb{R}^m$ is deemed safe if $\exists \mathbf{x}' \in S$ such that

$$l(\mathbf{x}') - L \cdot d(\mathbf{x}', \mathbf{x}) \geq h, \quad (4)$$

where $l(\mathbf{x}')$ is the lower bound of the confidence interval for \mathbf{x}' , and L is the Lipschitz constant [30]. Using the lower bound makes the above inequality more strict when compared to using the mean or upper bound, thus preventing unsafe evaluations resulting from noisy measurements (i.e., an input point \mathbf{x} that satisfies the above inequality is highly likely to be safe even if a noisy measurement affects its output value). We refer to Eq. (4) as **safe set operator**.

Having derived a set of safe solutions using the safe set operator, next, what is called an **optimistic expander operator** can be used to select input points from that set. The selected points could potentially expand the safe set by helping to classify additional input points (at least one) as safe in the next iteration. That is, an input point \mathbf{x} , not known to be safe, may be classified as safe if the following condition is satisfied:

$$u(\mathbf{x}^{\text{safe}}) - L \cdot d(\mathbf{x}^{\text{safe}}, \mathbf{x}) - \epsilon \geq h, \quad (5)$$

where $\epsilon = 0$ and the evaluation of the safe input point \mathbf{x}^{safe} gives a value equal to $u(\mathbf{x}^{\text{safe}})$, the upper bound of the confidence interval for the safety function of \mathbf{x}^{safe} [30] (optimistic attitude). The **optimistic expander operator** would select input points that are expected to be safe and also increase the size of the safe set if evaluated. Finally, an **optimistic safe set operator** uses the condition (Eq. 5), but \mathbf{x} can be any input point in \mathbb{R}^m and ϵ is the parameter representing the noise of a safety function, constructs an optimistic safe set consisting of \mathbf{x} .

Some algorithms calculate **naive confidence interval** and **intersectional confidence interval** [34], respectively defined as:

$$l_t = \mu_{t-1}(\mathbf{x}) - \beta_t \sigma_{t-1}(\mathbf{x}), \quad u_t = \mu_{t-1}(\mathbf{x}) + \beta_t \sigma_{t-1}(\mathbf{x}) \quad (6)$$

$$l_t = \max(l_{t-1}, \mu_{t-1}(\mathbf{x}) - \beta_t \sigma_{t-1}(\mathbf{x})), \quad u_t = \min(u_{t-1}, \mu_{t-1}(\mathbf{x}) + \beta_t \sigma_{t-1}(\mathbf{x})), \quad (7)$$

where β_t is a parameter that determines the width of the confidence interval and σ_{t-1} is the predicted standard deviation at point \mathbf{x} [27] at iteration t of the algorithm.

4 Algorithms

This section provides an overview of existing safe learning and optimization algorithms, and relationships between them (Tables 1 and 2). The tables can be seen as a first attempt of a classification of these algorithms.

Table 1 classifies the safe learning and optimization algorithms according to their aim, either optimization, which is subdivided by the type of method (EA and/or GPR), or learning, which is subdivided into RL or active learning. For each algorithm, the column *Method* shows key features of the algorithm (PSO denotes particle swarm optimization). We notice that most published works use either L -Lipschitz continuity or a classifier to infer the safety of input points. VA [18] and GoOSE [34] are methods that augment other algorithms to handle the safety of input points.

Table 2 divides the algorithms by the environment they assume: MDP or non-MDP. For each algorithm, the column *Initial safe seed* shows whether it requires at least one starting input point known to be safe. The column *Safety likelihood* presents the form in which safety of unobserved input points is predicted, either using labels derived from a classifier (safe vs unsafe) or safety constraint(s) (safe set vs the others) (*Label*), probability of safety estimated from a classifier (*Probability*), or none (safety is not inferred: *NI*). As shown in the table, classifiers are used to both predict safety labels of input points and calculate probability of safety. The column *Number of objectives* gives the number of objective functions considered by the algorithm. For example, the number of objectives that GoOSE [34] can handle is problem-specific as the algorithm is augmented onto algorithms that do not have a built-in approach to cope with safety constraints (similar to VA [18]) and thus is driven by the number of objectives that the underlying algorithm is dealing with. The column *Safety constraints* show how many safety constraint(s) can be handled by the algorithm, noticing that some algorithms are limited to a single *safety constraint*. MDP problems additionally include various forms of *ergodicity* as a safety requirement.

4.1 Evolutionary algorithms (EAs)

Arguably, the evolutionary computation community was one of the first to investigate safety issues as defined in this paper. The work of Kaji et al. [18] in 2009 introduced the violation avoidance (VA) method, a classification tool that is augmented onto an EA, while the work of Allmendinger and Knowles [2] in 2011 investigated the impact of safety issues on different stochastic optimizers (TGA, RBS, PHC). VA is able to deal with either binary or continuous input variables, while the optimizers considered in [2] assumed binary input variables.

The purpose of the Violation Avoidance (VA) method [18] is to avoid *risky* evaluations by replacing the ordinary offspring generation process of EAs. It applies the nearest neighbors method (NNs) using a weighted distance to decide over the safety of an input point prior to evaluating it. VA assumes the safety label of an input point to be same as the label of the nearest previously evaluated input point.³

³ The notion of risk is considered more formally in SAL [26], discussed in Section 4.3.

Table 1. Characterization of existing safe learning and optimization algorithms (Part 1).

Discipline	Paper	Year	Algorithm	Method	Comment
Optimization EA	Kaji et al. [18]	2009	VA	NNs	VA is a classifier augmented onto other EAs replacing their offspring generation process.
	Allmendinger and Knowles [2]	2011	TGA, RBS and PHC		Various EAs were proposed.
GPR	Sui et al. [30]	2015	SAFE _{OPT}	L -Lipschitz continuity	Inspired several safe algorithms [6, 7, 10, 31, 33–35].
	Berkenkamp et al. [7]	2016	Modified SAFE _{OPT}		Lipschitz constant-free model.
	Berkenkamp et al. [6]	2016	SAFE _{OPT} -MC	L -Lipschitz continuity	Multiple safety constraints are dealt with.
	Duivenvoorden et al. [10]	2017	Swarm-based SAFE _{OPT}	PSO	Lipschitz constant-free model.
	Schillinger et al. [24]	2017	SBO		Application of SAL [26] to optimization problem.
	Sui et al. [31]	2018	STAGE _{OPT}	L -Lipschitz continuity	Multiple safety constraints are dealt with. Optimization is performed in two independent processes: learning and optimization stages.
	Bachoc et al. [4]	2020	<i>EFI GPC sign</i>	GPC	A problem-specific GPC was proposed.
GPR with EA	Sacher et al. [23]	2018	EGO-LS-SVM	LS-SVM	GPR is used combined with EA.
Learning					
Reinforcement Learning	Turchetta et al. [33]	2016	SAFE _{MDP}	MDP, GPR, L -Lipschitz continuity	Application of SAFE _{OPT} to exploration task.
	Wachi et al. [35]	2018	SAFE _{EXP} _{OPT} -MDP	MDPs, GPR, L -Lipschitz continuity	Extension of SAFE _{MDP} made to maximize cumulative reward while safely exploring the state space.
	Turchetta et al. [34]	2019	GoOSE	GPR, L -Lipschitz continuity	The algorithm is augmented onto other unsafe exploration algorithms.
	Biçik et al. [8]	2019	SEOFUR	MDP, L -Lipschitz continuity	Transition functions are unknown.
Active Learning	Schreiter et al. [26]	2015	SAL	GPR, GPC	A problem-specific GPC was introduced.
	Schillinger et al. [25]	2018	SAL	GPR	Application of SAL to high pressure fuel supply system (HPFS).

Table 2. Characterization of existing safe learning and optimization algorithms (Part 2).

Environment	Algorithm	Initial safe seed	Safety likelihood	Number of objectives	Safety constraints
Non-MDP	VA[18]	Not required	Label	Single/ Multiple	One or multiple
	TGA, RBS, and PHC [2]	Not required	NI	Single	One
	SAFEOPT [30]	Required	Label	Single	One
	Modified SAFEOPT [7]	Required	Label	Single	One
	SAFEOPT-MC [6]	Required	Label	Single	One or multiple
	Swarm-based SAFEOPT [10]	Required	Label	Single	One or multiple
	STAGEOPT [31]	Required	Label	Single	One or multiple
	SAL [26]	Required	Label	Single	One
	SBO [24]	Required	Label	Single	One
	SAL [25]	Required	Label	Single	One
	EGO-LS-SVM [23]	Not required	Probability/ Label	Single	One or multiple
	<i>EFI GPC sign</i> [4]	Not required	Probability	Single	One
MDP	SAFEMDP [33]	Required	Label	Single	One <i>safety constraint</i> <i>Ergodicity</i> : One-step reachability and returnability
	SAFEEXP- <small>OPT</small> -MDP [35]	Required	Label	Single	One <i>safety constraint</i> <i>Ergodicity</i> : One-step reachability and returnability
	SEOFUR [8]	Required	Label	Single	Multiple <i>safety constraints</i> <i>Ergodicity</i> : Returnability
	GoOSE [34]	Required	Label	Single/Multiple	One <i>safety constraint</i> <i>Ergodicity</i> : <i>n</i> -step reachability and returnability

Allmendinger and Knowles [2] studied reconfigurable, destructible and unreplaceable experimental platforms in closed-loop optimization using EAs. Three types of EAs were investigated varying in the way offspring are generated and the level of collaboration of the individuals in a population. The EAs optimized a single objective function, while avoiding trials that have an output value less than a pre-defined (but unknown to the optimizer) lethal threshold, i.e. the safety threshold (h_j) in Eq. (1). An optimization run was terminated if a predefined number of unsafe input points was evaluated or a maximum number of function evaluations reached. No mechanism was put in place to determine safety of an input point prior to evaluating it, but any evaluated unsafe input point was banned from entering the population as a way to guide the population away from the unsafe region in the search space.

4.2 Algorithms related to SAFEOPT

Several safe learning and optimization algorithms are based on GPR, L -Lipschitz continuity and set theory. SAFEOPT [30] was the first algorithm of this type, and has inspired others to follow, such as modified SAFEOPT [7], SAFEOPT-MC [6], Swarm-based SAFEOPT [10], STAGEOPT [31], SAFEMDP [33], SAFEEXPOPT-MDP [35], Safe Exploration Optimized For Uncertainty Reduction (SEOFUR [8]) and GoOSE [34].

SAFEOPT [30] was proposed in 2015 for safe optimization aiming to avoid evaluating any *unsafe* input points altogether during the search, i.e., the number of allowed failures is zero. Roughly speaking, the algorithm uses a GP to model the objective function, which together with a known safety threshold is taken as the safety constraint, and the algorithm selects an input point that has the maximum width of confidence interval among those belonging to a *maximizers set* or *expanders set*. The algorithm constructs a safe set using the *safe set operator* (Eq. 4), and to construct the *expanders set* it uses the *optimistic expander operator* (Eq. 5). That is, given an input point deemed to be safe and whose evaluation could potentially help to classify additional input points as safe, then (i) it belongs to the *expanders set* and (ii) the *maximizers set* will comprise input points belonging to the safe set whose upper confidence bound is greater than the greatest lower confidence bound among those calculated for all input points in the safe set. Intuitively speaking, generating *expanders set* and *maximizers set* corresponds to classifying input points in the safe set into (i) *safe set expansion*, which is expected to expand the safe set at the next iteration, and (ii) *safe set exploitation*, which is likely to yield high output value at the current iteration, by evaluating an input point in the set at the current iteration. Generally, whether to expand or exploit is decided at each step of the algorithm. SAFEOPT [30] uses intersectional confidence intervals (Eq. 7), and the same is done by SAFEMDP [33], SAFEEXPOPT-MDP [35], STAGEOPT [31] and GoOSE [34].

In general, algorithms inspired by SAFEOPT [30] share the fundamental structure of SAFEOPT [30]: Constructing the safe set first, and then using it to construct the *expanders set* and *maximizers set*. However, there are some distinctive features between the algorithms. In 2016, three algorithms were proposed: Modified SAFEOPT [7], SAFEOPT-MC [6] and SAFEMDP [33]. Unlike the original SAFEOPT approach [30], the modified SAFEOPT [7] estimates the safe, *maximizers* and *expanders* sets without the specification of a Lipschitz constant. The safe set consists

of input points whose lower confidence bound is greater than a safety threshold. To estimate the *expanders* set, the algorithm constructs a GP based on both previously evaluated input points and an artificial data point (with a noiseless measurement of the upper confidence bound) selected from the safe set (as opposed to using previously evaluated input points only as done in the original SAFEOPT approach). SAFEOPT-MC can deal with multiple constraints, and SAFEMDP applies SAFEOPT to RL problems. Swarm-based SAFEOPT [10], proposed in 2017, applies a variant of PSO to SAFEOPT, where multiple independent swarms (sets of input points) are used to construct the *maximizers set* and *expanders set*. The objective function differs across swarms to reflect the different goals when constructing the *maximizers* and *expanders set*. Initialized with initial safe seeds, it updates the safe set only when the input points (also referred to as particles in PSO), found at each run of the PSO, are sufficiently far away from the safe input points in the safe set. Here, input points whose lower confidence bound is greater than a safety threshold are assumed to be safe. The input points in the safe set are used to decide the initial positions of the particles at each iteration. As in typical PSO, the particles move toward new positions by considering their current positions and velocities. In 2018, Sui et al. [31] and Wachi et al. [35] proposed the algorithms STAGEOPT and SAFEEXPOPT-MDP, respectively. STAGEOPT can deal with multiple safety constraints and divides the process in two independent stages: Safe region expansion stage and optimization stage corresponding to *safe set exploration* and *safe set exploitation* [31], respectively. SAFEEXPOPT-MDP is an extension of SAFEMDP (e.g., unsafe set, uncertain set, etc), and its goal is to maximize the cumulative reward rather than *safe set expansion* (i.e., the safety function is not the objective function). Lastly, in 2019, the RL community proposed the algorithm GoOSE [34], which is augmented onto other unsafe algorithms as a way to control the selection of safe input points.

Algorithms able to handle problems with multiple safety constraints, e.g., SAFEOPT-MC [6] and STAGEOPT [31], model the individual safety constraints, as well as the objective function, using independent GPs. Then, the safe set is the intersection of safe sets estimated from the safety functions, where the *safe set operator* (Eq. 4) is applied to each safety function separately. Now let us remind that the *maximizers set* is used for *safe set exploitation*, meaning this set is constructed based on the objective function values of input points in the current safe set (ignoring information from the safety functions). Since there is one objective function only, the algorithms SAFEOPT and SAFEOPT-MC construct the *maximizers set* using the same approach. However, it is not required for STAGEOPT, as it has its own independent stage devoted for *safe set exploitation*, and optimization is dealt with by input point selection criterion in that stage. Lastly, *optimistic expander operator* (Eq. 5) is applied to each safety constraint, and all of them should be met when including an input point from the safe set into the *expanders set*. However, SAFEOPT-MC and STAGEOPT apply the structure in slightly different ways, e.g., they define Lipschitz constant(s) and safety threshold(s) differently. When SAFEOPT is applied to RL problems [33, 35], the set of states allowed to visit is restricted by *ergodicity*. This means that the suggested states should be reachable in one-step and returnable to states in the safe set, established at the previous iteration, in several

steps. However, GoOSE [34] applies n -step expansion until convergence of safe set and *expanders set* at each algorithm iteration by applying n -step reachability and returnability. GoOSE is augmented onto an unsafe RL or Bayesian optimization algorithm (i.e., one that is not designed to deal with safety constraints). The unsafe algorithm suggests an input point \mathbf{x}^* that belongs to an *optimistic safe set*. Then, GoOSE evaluates input points from the *expanders set*, until \mathbf{x}^* is inferred to be safe. Otherwise, GoOSE asks for another input point from the unsafe algorithm. In GoOSE, however, additional conditions are used with *optimistic expander operator* for constructing *expanders set*.

SEOFUR [8] is an algorithm for safe exploration of deterministic MDPs with unknown transition functions. Starting from a known safe state set and a list of actions that connect the safe states, and assuming that the unknown transition function is L -Lipschitz continuous over both states and actions, the algorithm tries to efficiently and safely explore the search space. The learned (known) transitions are represented in form of a (deterministic) transition function, while an uncertain transition function is defined to handle unknown states. The uncertain transition function maps each state-action pair to all of its possible outcomes, and if there is a state-action pair whose possible outcomes constitute a subset of the known safe state set, then the state is deemed safe. This problem-specific *safe set expansion* method is repeated at each algorithm iteration until it converges (n -step returnability). The algorithm removes uncertainty as much as possible at each iteration by greedily optimizing an expected uncertainty reduction measure. The paper [8] applies SEOFUR to a simulation problem with safety constraints.

4.3 Safe learning and optimization with a classifier

In addition to the VA method [18] (Section 4.1), there are three more algorithms that use a classification method for safety inference of input points: Safe Active Learning (SAL) [26], EGO-LS-SVM [23] and *EFI GPC sign* [4]. In particular, EGO-LS-SVM [23] and *EFI GPC sign* [4] were proposed to avoid simulations from crashing.

SAL [26] is an algorithm for learning a regression model when unknown regions of the input space can be unsafe. SAL builds two GPs to approximate an objective function and a discriminative function, mapped to the unit interval to describe the class (i.e., safe or unsafe) likelihood, for a problem-specific Gaussian Process classifier (GPC). SAL assumes that each evaluated input point \mathbf{x} provides two additional output values: A negative (unsafe) or positive (safe) label $c(\mathbf{x}) \in \{-1, +1\}$ and the value of a black-box, possibly noisy function $h: \mathbb{X} \rightarrow (-1, 1)$, where $\mathbb{X} \subset \mathbb{R}^m$ for m -dimensional input points. The function $h(\mathbf{x})$ provides a noisy *risk* value for evaluated safe points $\mathbf{x} \in \mathbb{X}$ (i.e., $c(\mathbf{x}) = +1$) close to the unknown boundary of the safe input region, while it provides no useful information for unsafe points, i.e., $c(\mathbf{x}) = -1$. SAL also assumes an upper bound for the expected number of failures. At each iteration, the algorithm selects an input point with the highest conditional variance given previous observations among those expected to be safe according to the GPC. Safe Bayesian Optimization (SBO) [24] is fundamentally the same as SAL with the core difference being that, at each iteration of the algorithm, it selects an

input point whose lower confidence bound is the minimum, assuming minimization of the loss function. Another difference between SAL and SBO is that in SBO a standard GP regression, which is trained with discriminative function values only, is used for the discriminative function, while the use of class labels (safe vs unsafe) for training GPC is also an option for SAL. Training a standard GP with discriminative function values only was also the approach adopted in [25], where SAL was applied to safe learning of a high pressure fuel supply system.

EGO-LS-SVM [23] has been designed for safe optimization of complex systems, where simulations are subject to abrupt terminations due to unphysical configurations, ill-posed problems or lack of numerical robustness. That is, it assumes that there is a non-computable sub-domain in the search domain that cannot be expressed by inequality constraints, and thus, applies a binary classifier (least-squares support vector machine or LS-SVM) for this sub-domain. However, the classifier may be used with some inequality constraints that define other non-computable sub-domains. EGO-LS-SVM constructs an independent GP to model an objective function, and assigns observations into the *safe* or *unsafe class*, which represent computable and non-computable input points, respectively. Based on the safety labels attached to the previous observations, LS-SVM predicts the probability that an input point belongs to the safe/unsafe class. The paper [23] also proposed four different selection criteria for the next input point that combine this probability with the augmented expected improvement (AEI) acquisition function [17], which is an input point selection function for Bayesian optimization. Another interesting aspect of the algorithm is that it uses the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [16] to estimate the hyper-parameters of the kernel function for GP regression and for LS-SVM.

Lastly, *EFI GPC sign* [4] was also designed to prevent simulation crashes; thus, it aims to progress an optimization process by efficiently avoiding input points that are likely to fail. In this context, the safety function is binary (safe/unsafe) and a failure implies that the objective function cannot be evaluated. The approach differs from EGO-LS-SVM in that it cannot construct multiple models for safety constraints. *EFI GPC sign* constructs a GP for learning binary inputs, representing safe/unsafe evaluations, that is different from the classical GPC and more appropriate for deterministic safety functions. In addition, GPR is used to model the objective function by keeping only the safe input points. Finally, *EFI GPC sign* selects an input point by considering the probability of non-failure multiplied by the standard expected improvement acquisition function.

5 Discussion and future research

In this paper, we reviewed and contextualized 18 algorithms designed for safe learning and optimization.

Two studies in the area of evolutionary computing [2, 18] proposed algorithms (VA [18] and modified versions of EAs [2]) designed for safe optimization. In particular, VA [18] is a flexible approach that can be augmented onto other EAs.

Except for SEOFUR [8] and the aforementioned EAs, the other safe learning and optimization algorithms reviewed in this survey are based on GP regression. However, we are able to observe a trend and can divide these algorithms into categories. The first division we observe is that algorithms adopt the L -Lipschitz continuity assumption [6, 8, 30, 31, 33–35], use the lower confidence bound [7, 10], or apply a classifier to measure the safety of input points [4, 23–26]. Interestingly, it was found that modified SAFEOP [7] and Swarm-based SAFEOP [10] share a similar structure with algorithms using the L -Lipschitz continuity assumption, but are free from deciding the Lipschitz constant. Furthermore, SEOFUR [8] is based on a method that applies the L -Lipschitz continuity assumption for the problems where transition functions are unknown. Also, while some algorithms [4, 7, 24–26, 30, 33–35] deal with one *safety constraint*, others [6, 10, 23, 31] can handle multiple ones. Lastly, while *ergodicity* and *safety constraint(s)* were classified as different concepts applied to distinctive approaches in a previous survey in 2015 [13], we observed that they are used together in SAFEMDP [33], SAFEEXPOT-MDP [35] and GoOSE [34].

Given the above observations, we envision several open questions for future research. First, how to estimate the Lipschitz constant and safety threshold when these are *a priori* unknown. Second, are there real-world applications that would benefit/require alternative formulations of the problem, such as safety thresholds being a function of the input variables and/or change over time, i.e., use $h_j(\mathbf{x})$, $h_j(t)$ or $h_j(\mathbf{x}, t)$ instead of a fixed constant h_j . Finally, it is not clear how to apply multiple safety constraints to MDP problems.

Acknowledgements. M. López-Ibáñez is a “Beatriz Galindo” Senior Distinguished Researcher (BEAGAL 18/00053) funded by the Spanish Ministry of Science and Innovation.

References

1. Allmendinger, R.: Tuning evolutionary search for closed-loop optimization. Ph.D. thesis, The University of Manchester, UK (2012)
2. Allmendinger, R., Knowles, J.D.: Evolutionary search in lethal environments. In: International Conference on Evolutionary Computation Theory and Applications, pp. 63–72, SciTePress (2011)
3. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* **3**, 397–422 (Nov 2002)
4. Bachoc, F., Helbert, C., Picheny, V.: Gaussian process optimization with failures: Classification and convergence proof. *Journal of Global Optimization* (2020)
5. Bäck, T.: Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press (1996)
6. Berkenkamp, F., Krause, A., Schoellig, A.P.: Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics. Arxiv preprint [arXiv:1602.04450](https://arxiv.org/abs/1602.04450) (2016), URL <http://arxiv.org/abs/1602.04450>
7. Berkenkamp, F., Schoellig, A.P., Krause, A.: Safe controller optimization for quadrotors with Gaussian processes. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 491–496, IEEE (2016)

8. Bıynk, E., Margoliash, J., Alimo, S.R., Sadigh, D.: Efficient and safe exploration in deterministic Markov decision processes with unknown transition models. In: 2019 American Control Conference (ACC), pp. 1792–1799, IEEE (2019)
9. Brochu, E., Cora, V., de Freitas, N.: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Arxiv preprint [arXiv:1012.2599](https://arxiv.org/abs/1012.2599) (Dec 2010), URL <https://arxiv.org/abs/1012.2599>
10. Duivenvoorden, R.R.P.R., Berkenkamp, F., Carion, N., Krause, A., Schoellig, A.P.: Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning. *IFAC-PapersOnLine* **50**(1), 11800–11807 (2017)
11. Ferrer, J., López-Ibáñez, M., Alba, E.: Reliable simulation-optimization of traffic lights in a real-world city. *Applied Soft Computing* **78**, 697–711 (2019)
12. Forrester, A.I.J., Keane, A.J.: Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences* **45**(1-3), 50–79 (2009)
13. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* **16**(1), 1437–1480 (2015)
14. Geibel, P.: Reinforcement learning for MDPs with constraints. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, vol. 4212, pp. 646–653 (2006)
15. Gosavi, A.: Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing* **21**(2), 178–192 (2009)
16. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* **9**(2), 159–195 (2001)
17. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization* **34**(3), 441–466 (2006)
18. Kaji, H., Ikeda, K., Kita, H.: Avoidance of constraint violation for experiment-based evolutionary multi-objective optimization. In: *Proceedings of the 2009 Congress on Evolutionary Computation (CEC 2009)*, pp. 2756–2763, IEEE Press, Piscataway, NJ (2009)
19. Knowles, J.D.: Closed-loop evolutionary multiobjective optimization. *IEEE Computational Intelligence Magazine* **4**, 77–91 (2009)
20. Likar, B., Kocijan, J.: Predictive control of a gas–liquid separation plant based on a Gaussian process model. *Computers & Chemical engineering* **31**(3), 142–152 (2007)
21. Moldovan, T.M., Abbeel, P.: Safe exploration in Markov decision processes. In: Langford, J., Pineau, J. (eds.) *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, pp. 1451–1458, Omnipress (2012)
22. Rasmussen, C.E., Williams, C.K.I.: *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA (2006)
23. Sacher, M., Duvigneau, R., Le Maitre, O., Durand, M., Berrini, E., Hauville, F., Astolfi, J.A.: A classification approach to efficient global optimization in presence of non-computable domains. *Structural and Multidisciplinary Optimization* **58**(4), 1537–1557 (2018)

24. Schillinger, M., Hartmann, B., Skalecki, P., Meister, M., Nguyen-Tuong, D., Nelles, O.: Safe active learning and safe Bayesian optimization for tuning a PI-controller. *IFAC-PapersOnLine* **50**(1), 5967–5972 (2017)
25. Schillinger, M., Ortelt, B., Hartmann, B., Schreiter, J., Meister, M., Nguyen-Tuong, D., Nelles, O.: Safe active learning of a high pressure fuel supply system. In: *Proceedings of the 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016 and the 57th SIMS Conference on Simulation and Modelling SIMS 2016*, pp. 286–292, Linköping University Electronic Press (2018)
26. Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H., Toussaint, M.: Safe exploration for active learning with Gaussian processes. In: *Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2015, Lecture Notes in Computer Science*, vol. 9286, pp. 133–149, Springer (2015)
27. Schulz, E., Speekenbrink, M., Krause, A.: A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology* **85**, 1–16 (Aug 2018)
28. Small, B.G., McColl, B.W., Allmendinger, R., Pahle, J., López-Castejón, G., Rothwell, N.J., Knowles, J.D., Mendes, P., Brough, D., Kell, D.B.: Efficient discovery of anti-inflammatory small-molecule combinations using evolutionary computing. *Nature Chemical Biology* **7**(12), 902–908 (2011)
29. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems (NIPS 25)*, pp. 2960–2968, Curran Associates, Red Hook, NY (2012)
30. Sui, Y., Gotovos, A., Burdick, J.W., Krause, A.: Safe exploration for optimization with Gaussian processes. In: Bach, F., Blei, D. (eds.) *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, vol. 37, pp. 997–1005 (2015)
31. Sui, Y., Zhuang, V., Burdick, J.W., Yue, Y.: Stagewise safe Bayesian optimization with Gaussian processes. In: Dy, J.G., Krause, A. (eds.) *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Proceedings of Machine Learning Research*, vol. 80, pp. 4788–4796, PMLR (2018)
32. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press-Cambridge, MA, 2nd edn. (2018)
33. Turchetta, M., Berkenkamp, F., Krause, A.: Safe exploration in finite Markov decision processes with Gaussian processes. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems (NIPS 29)*, pp. 4312–4320 (2016)
34. Turchetta, M., Berkenkamp, F., Krause, A.: Safe exploration for interactive machine learning. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) *Advances in Neural Information Processing Systems (NIPS 32)*, pp. 2887–2897 (2019)
35. Wachi, A., Sui, Y., Yue, Y., Ono, M.: Safe exploration and optimization of constrained MDPs using Gaussian processes. In: McIlraith, S.A., Weinberger, K.Q. (eds.) *AAAI Conference on Artificial Intelligence*, pp. 6548–6556, AAAI Press (Feb 2018)