Journal Pre-proof

Methods for constrained optimization of expensive mixed-integer multi-objective problems, with application to an internal combustion engine design problem

João A. Duro, Umud Esat Ozturk, Daniel C. Oara, Shaul Salomon, Robert J. Lygoe, Richard Burke, Robin C. Purshouse

PII: S0377-2217(22)00677-4
DOI: https://doi.org/10.1016/j.ejor.2022.08.032
Reference: EOR 18067

To appear in: *European Journal of Operational Research*

Received date: 10 December 2020
Accepted date: 24 August 2022

Please cite this article as: João A. Duro, Umud Esat Ozturk, Daniel C. Oara, Shaul Salomon, Robert J. Lygoe, Richard Burke, Robin C. Purshouse, Methods for constrained optimization of expensive mixed-integer multi-objective problems, with application to an internal combustion engine design problem, *European Journal of Operational Research* (2022), doi: https://doi.org/10.1016/j.ejor.2022.08.032

* Engine design problems are compute-expensive, mixed-integer and constrained
* A multi-objective Bayesian optimizer is extended to solve these type of problems
* Probability of feasibility constraint handling is better than using penalty functions
* The new Bayesian optimizer converges faster than a popular dominance-based algorithm
* The use of feasible and infeasible solutions in the training set improves performance

Methods for constrained optimization of expensive mixed-integer multi-objective problems, with application to an internal combustion engine design problem

João A. Duro[a], Umud Esat Ozturk[b], Daniel C. Oara[a], Shaul Salomon[a,c], Robert J. Lygoe[d], Richard Burke[b], Robin C. Purshouse[a]

[a] *Department of Automatic Control and Systems Engineering, The University of Sheffield, UK*
[b] *The Institute for Advanced Automotive Propulsion Systems, University of Bath, Bath, BA2 7AY, UK*
[c] *ORT Braude College of Engineering, Israel*
[d] *Product Development Europe, Dunton Technical Centre UK, Ford Motor Co. Ltd*

Abstract

Engineering design optimization problems increasingly require computationally expensive high-fidelity simulation models to evaluate candidate designs. The evaluation budget may be small, limiting the effectiveness of conventional multi-objective evolutionary algorithms. Bayesian optimization algorithms (BOAs) are an alternative approach for expensive problems but are underdeveloped in terms of support for constraints and non-continuous design variables—both of which are prevalent features of real-world design problems. This study investigates two constraint handling strategies for BOAs and introduces the first BOA for mixed-integer problems, intended for use on a real-world engine design problem. The new BOAs are empirically compared to their closest competitor for this problem—the multi-objective evolutionary algorithm NSGA-II, itself equipped with constraint handling and mixed-integer components. Performance is also analysed on two benchmark problems which have similar features to the engine design problem, but are computationally cheaper to evaluate. The BOAs offer statistically significant convergence improvements of between 5.9% and 31.9% over NSGA-II across the problems on a budget of 500 design evaluations. Of the two constraint handling methods, constrained expected improvement offers better convergence than the penalty function approach. For the engine problem, the BOAs identify improved feasible designs offering 36.4% reductions in nitrogen oxide emissions and 2.0% reductions in fuel consumption when compared to a notional baseline design. The use of constrained mixed-integer BOAs is recommended for expensive engineering design optimization problems.

Keywords: Multiple objective programming; Bayesian optimization; Mixed continuous and discrete variables; Constraint handling; Gasoline engine

1. Introduction

Complex manufacturing industries, such as automakers and their supply chains, are increasingly adopting digital technologies to transform their processes and products (IDE (2021)). Virtual engineering—the use of modelling and simulation to evaluate the performance of product designs—is used extensively within these industries, with increasing reliance on high-fidelity physics simulations, sometimes integrated together in complex multiphysics co-simulations (Keyes et al. (2013)), that are computationally costly to execute (Forrester & Keane (2009)). In this context, the number of alternative candidate designs that can be

Email addresses: j.a.duro@sheffield.ac.uk (João A. Duro), ueo24@bath.ac.uk (Umud Esat Ozturk), dcoara1@sheffield.ac.uk (Daniel C. Oara), shaulsal@braude.ac.il (Shaul Salomon), blygoe@ford.com (Robert J. Lygoe), r.d.burke@bath.ac.uk (Richard Burke), r.purshouse@sheffield.ac.uk (Robin C. Purshouse)

evaluated within a design optimization process may be substantially constrained by the available resources. Within the academic literature, the maximum budget for such expensive problems is typically assumed to be in the region of 500 evaluations or fewer (Chugh et al. (2019); Keane (2006); Knowles & Hughes (2005)).

Within the automotive industry, the increased use of computationally expensive simulations has been at least partially driven by commitments from governments around the world to achieve reductions in greenhouse gases and improvements in air quality. These commitments have led to tighter vehicle fuel consumption and emissions standards, with testing procedures more representative of real-life driving conditions (Ramos et al. (2018)). The most recent vehicle test procedures such as the *worldwide harmonised light vehicles test procedure* (WLTP) and *real driving emissions* (RDE) test procedure require evaluation of fuel consumption and exhaust emissions during 30-minute and 90-120-minute long tests. These tests need to be repeated at multiple temperatures, multiple altitudes and for multiple driving styles. Accurate evaluation of these test cycles requires multiple high-fidelity simulations that represent the transient response of the engine over the full test duration. Evaluation of a single engine design may take several hours to execute on a high performance computing platform, with resources available to conduct at most a few hundred such evaluations. Engine design engineers aim to find a Pareto optimal set of designs that capture the inherent trade-off between fuel efficiency (carbon dioxide–CO2–emissions) and nitric oxide–NOx–emissions. With a design space consisting of a dozen or more architectural variables and control parameters, together with a demanding set of constraints to satisfy, identifying a feasible Pareto optimal set can be challenging on a small evaluation budget. As a result, many existing approaches have limited their ambition to identifying just a single point on the Pareto front, by solving single-objective problems (Alonso et al. (2007); Karra & Kong (2010); Millo et al. (2018); Tadros et al. (2019); Togun & Baysec (2010)).

Over the past two decades, multi-objective evolutionary algorithms (MOEAs) have become very popular methods for identifying the Pareto optimal set of designs for a problem—recent examples include (Alcaraz et al. (2020); Avilés et al. (2020); Bird et al. (2020); Chen et al. (2021); Drake et al. (2020); Kuk et al. (2021)). There have also been attempts to use MOEAs to identify Pareto optimal sets of engine designs (Corre et al. (2019); DErrico et al. (2011); Lotfan et al. (2016)). However, a recent survey of MOEA applications for expensive-to-evaluate problems identified criticism of the unrealistically large number of evaluations that the algorithms required in order to achieve convergence (Chugh et al. (2019)). The response to this criticism has been to incorporate low-fidelity models into the optimization process, that aim to approximate the outputs of a high-fidelity simulation. These data-driven models are variously known as *metamodels*, *surrogates* or *emulators*, and are often estimated using data generated during the optimization process itself. Architectures for surrogate-assisted MOEAs vary in terms of the role of evolutionary computation in the search process. Here, we focus on a popular class of architectures, known as Bayesian Optimization Algorithms (BOAs), that typically use evolutionary algorithms to search over a so-called *acquisition function* or *infill criterion* that is generated from progressively updated beliefs about the implicit functional form of the black-box simulation model. This implicit functional form is often encoded as a Gaussian process (or *Kriging* model) (Krige (1951); Rasmussen & Williams (2006)), and this approach has become popular for use in engineering applications (De Ath et al. (2021); Forrester et al. (2008); Wang & Shan (2007)).

Despite the apparent promise of BOAs, uncertainty remains over their ability to handle common features of real-world engineering applications such as engine design problems. Such problems typically feature a mix of continuous and discrete design variables; however BOAs tend to assume continuous variables and, to our knowledge, no BOAs have yet been developed for mixed-integer variables. In addition to objectives, engineering problems tend also to feature a large number of constraints. However, as will be shown in the next section, constraints handling is a recent innovation in BOAs, with demonstrations of effectiveness only for very simple benchmark problems and no clear conclusion as to the most effective method to use. Since, for Kriging-based BOAs, the number of data points (i.e., evaluated designs) that can be used to build the surrogate model is limited by the computational complexity of the estimation process, the choice of candidate designs to inform the surrogate is also an open question. In this paper, motivated by our real-world engineering problem, we aim to advance BOA research by developing and testing the first mixed-integer BOA and examining alternative constraint handling and surrogate-building strategies, with a view to making firmer recommendations for other analysts considering using a BOA on their problem. Our algorithmic development builds on the seminal *efficient global optimization* (EGO) and *Pareto efficient global*

optimization (ParEGO) methods by Jones et al. (1998) and Knowles (2005) respectively and is available to the community as open-source software (Duro et al. (2020)).

The remainder of this paper is organised as follows. A literature review that covers the application of BOAs to constrained multi-objective problems is conducted in Section 2. Some preliminaries and background on Bayesian optimization are in Section 3. The proposed methodology in Section 4 covers the strategy to handle mixed continuous and discrete variables (Section 4.1); a proposed algorithm to select a subset of training points for constructing a surrogate model (Section 4.2); and the constraint handling strategies for ParEGO (Section 4.3). The empirical analysis setup used in this paper is in Section 5. The demonstration of the proposed methodology on benchmark problems is in Section 6. The demonstration on the real engineering problem is in Section 7, and the paper concludes with Section 8.

2. Bayesian optimization algorithms for constrained multi-objective optimization

In this review we consider Kriging based BOAs that have been applied to multi-objective problems with constraints, and the focus is on their constraint handling strategy, the type of optimization problems that they have been applied to, and which strategy (if any) is used to manage the number of training points for constructing the surrogate model. In the text below, unless stated, the decision variables are all continuous.

2.1. Probability of feasibility based approaches

This section discusses approaches that have used the *probability of feasibility* (PoF) which is often combined with an infill criterion such as the *probability of improvement* (PoI) or the *expected improvement* (EI) for handling constraints.

Emmerich and colleagues (Emmerich (2005); Emmerich et al. (2008); Emmerich et al. (2006)) have shown how to integrate several infill strategies into a multi-objective evolutionary algorithm. Amongst them is the PoI, EI, and lower confidence bound. To determine the EI in the multi-objective context, their approach relies on the hypervolume indicator, and this is referred to as the hypervolume-based EI. In addition, the PoI and EI (including the hypervolume-based EI) have both been extended for handling constraints; this is achieved by multiplying their corresponding expressions by the PoF. To estimate the hypervolume the authors proposed the use of a Monte Carlo approach, which requires decomposing the non-dominated region into a uniform grid of cells, and performing a hypervolume calculation separately for each cell.

Couckuyt et al. (2014) have shown that the hypervolume indicator could be also used to determine the PoI, leading to the hypervolume-based PoI. The same authors have also proposed an improvement to the hypervolume estimation which scales better with the number of objectives and number of solutions when compared with the approach used by Emmerich et al. (2006). The work by Couckuyt et al. (2014) was only demonstrated on unconstrained multi-objective problems, and this led Singh et al. (2014) to integrate the PoF into the hypervolume-based PoI. Feliot et al. (2017) have proposed an improvement to the hypervolume estimation which compares solutions based on their objective value and constraint values, and have demonstrated this on the hypervolume-based EI with the PoF. Other authors such as Martínez-Frutos & Herrero-Pérez (2016) and Do et al. (2021), have combined the PoF with the hypervolume-based EI and the hypervolume-based PoI, respectively.

On a separate research direction Forrester and colleagues (Forrester & Keane (2009); Forrester et al. (2008)) have proposed a multi-objective version of the EI that relies on determining the probability that a new design improves (or dominates) over existing non-dominated solutions. Although the proposed formulation is limited to two objectives, they have combined with the PoF for dealing with constraints.

In terms of how to manage the number of training points for the surrogate model, Emmerich et al. (2006) used local surrogate models as opposed to global models. The latter is used if the number of points is not higher than $2d$, where d is the number of decision variables, but once the number of points exceeds this threshold a local surrogate model is constructed around each point by using the closest (in terms of Euclidean distance) $2d$ points in the decision space. In Forrester & Keane (2009), strategies for selecting a subset of training points are not discussed, but they recommend using other surrogate techniques that are cheaper-to-build (e.g. polynomial models) in case the number of points is higher than 500 or if there are

more than 20 decision variables. Moreover, the number of training points is not fixed in Do et al. (2021); Feliot et al. (2017); Martínez-Frutos & Herrero-Pérez (2016); Singh et al. (2014)).

In terms of the optimization problems, Emmerich et al. (2006) have demonstrated their approach on many unconstrained single- and multi-objective problems, and also on a multipoint airfoil optimization problem with three objectives and 10 constraints. Forrester & Keane (2009) have demonstrated their approach on a variant of the classic Nowacki beam problem (Nowacki (1980)) with two objectives and five constraints. Singh et al. (2014) used the same Nowacki problem, and also a microwave filter design problem with two objectives and seven constraints. Martínez-Frutos & Herrero-Pérez (2016) used three test problems with two objectives and two constraints. Feliot et al. (2017) used several test problems, including two-objective problems with 2–6 constraints, and a five-objective problem with 7 constraints. Do et al. (2021) have used a steel frame design problem with uncertain parameters, three objectives and a scalable number of constraints depending on several structural parameters, up to a maximum of 13 probabilistic constraints and 10 deterministic constraints (although it seems that only the probabilistic constraints are being surrogated). Their approach was also demonstrated on a real-world two-bar truss problem with two objectives and two constraints. The authors have noted that different runs of the optimization algorithm could lead to different Pareto-optimal solutions when the feasible region of the optimization problem is large.

2.2. Penalty function based approaches

The approaches described in this section rely on the use of penalty functions to handle constraints. This often means that the performance of the solutions is penalised depending on their constraint violation.

Chugh et al. (2016) have proposed an extension to a surrogate-assisted evolutionary multi-objective optimization algorithm, known as K-RVEA, for dealing with constraints by making use of a penalty function based approach taken from Miettinen et al. (2003). K-RVEA uses reference direction vectors to decompose the original problem into a number of subproblems, and the subproblems are solved simultaneously to generate a set of solutions that approximate the entire Pareto front. The computationally expensive objective functions are replaced by surrogate models, but the constraints are not surrogated and therefore are assumed to be inexpensive to evaluate. The training set used to construct the surrogates has a maximum fixed size. The major contribution of this work is to show the effect of infeasible solutions in the training set. For this, two approaches have been considered: in one case only feasible solutions are chosen; while in the other case both feasible and infeasible solutions are used. The simulation results show that having a mix of feasible and infeasible solutions in the training set produces better results than just relying on feasible solutions. This approach has been demonstrated on three benchmarked problems with the number of objectives ranging from 3 to 10 and the number of constraints from 1 to 10.

Hussein & Deb (2016) have also used a decomposition-based evolutionary algorithm with a penalty-based approach for handling constraints. The search is conducted over the EI, and a single Pareto optimal solution is found at a time. The fitness of each infeasible solution is replaced by the worst objective function value of all feasible solutions plus the product of all normalised constraint breaches. A subset of training points is chosen from an archive by selecting those with the shortest orthogonal distance to the reference direction vector. This algorithm has been demonstrated on several unconstrained test problems, and four two-objective test problems with 2–6 constraints.

Koziel & Pietrenko-Dabrowska (2022) have used an infill criterion that relies on an inverse surrogate—predicts the decision variable values for a given objective vector. A point is chosen in objective space by the use of a triangulation approach, where the point in the middle of the largest simplex is chosen to be evaluated next. It is expected for points generated in this way to cover the Pareto front in a uniform manner. To handle the constraints a penalty function approach is combined with a local search method. There is no strategy to manage the size of the training set, which increases at the end of each iteration. The algorithm has been demonstrated on two two-objective microwave circuit problems, a Branch-line coupler problem with three constraints, and a impedance matching transformer problem with one constraint.

2.3. Others

Jeong & Obayashi (2005) utilise the EI of objective functions directly in the optimization process. This means that a surrogate model is constructed for each objective function (the constraints are not surrogated),

and a genetic algorithm finds a solution that maximises the EI, individually for each objective function. The authors claim that their selection mechanism helps to identify non-dominated solutions. The constraints are handled by selecting only feasible solutions to construct the surrogate model. Initially the training points are generated by *Latin Hypercube Sampling* (LHS), and during the optimization more points are added to the training set by selecting those with the largest EI with respect to each objective. The number of training points is not fixed. This algorithm has been demonstrated on a transonic airfoil design problems with two objectives and one constraint.

Li et al. (2009, 2008) builds a surrogate model of each objective, and the optimization is conducted by a multi-objective evolutionary algorithm inspired by NSGA-II. The Kriging variance is used as a measure of correctness for the predicted responses. At each generation the surrogate is used to evaluate all individuals in the population, and expensive evaluations take place if the Kriging variance is below than some threshold. To handle the constraints, a surrogate model of each constraint is constructed and a criterion that relies on the Kriging variance is used to indicate whether or not a solution is said to be feasible. The number of training points is not fixed. The algorithm has been demonstrated on five two-objective test problems: two problems with 2 and 6 constraints; and three unconstrained problems where one of the problems has discrete decision variables, while all the others only have continuous ones. It is not clear how the discrete nature of the optimization problem is handled in this approach.

Mlakar et al. (2015) have proposed a multi-objective optimization algorithm based on the concept of *differential evolution* known as GP-DEMO. This algorithm relies on a sparse–approximation method which is more computationally efficient than having to compute the inverse of the entire covariance matrix during the process of learning the surrogate model. The approach is to retain most information contained in the full training set by reducing the size of the covariance matrix, as opposed to selecting a subset of training points. This algorithm has been applied to several optimization problems with two objectives, where 12 are benchmark problems (three of them with constraints), and two real-world problems. It is unclear how constraints are handled.

Garrido-Merchán & Hernández-Lobato (2019) rely on the concept of entropy as an alternative to the EI and proposed an algorithm called Predicted Entropy Search for Multi-objective Optimization with Constraints (PESMOC). The constraints are integrated into the predicted distribution of the Gaussian process models in a similar way as done for the EI with PoF (see Emmerich et al. (2006)). This algorithm has been demonstrated on several two-objective problems, including four test problems and three real-world problems, with the number of constraints ranging from 2 to 6.

2.4. Research gaps

Following the above review, the research gaps identified are as follows:

1. The BOAs reviewed above have been applied to several benchmark and real-world problems, where the number of objectives ranges from 2 to 10, and the number of constraints from 1 to 23. Most are continuous problems, and only in Li et al. (2009, 2008) there is a problem with solely discrete variables. Therefore, to the best of the authors' knowledge, existing BOAs have not yet been applied to expensive constrained multi-objective problems with a mix of continuous and discrete decision variables.
2. The most popular constraint handling approach identified above is to use a combination of the EI with the PoF, known as constrained EI. An alternative approach found in two publications (Chugh et al. (2016); Hussein & Deb (2016)) is to use penalty functions, which have been extensively used in the past to solve constrained optimization problems (Farmani & Wright (2003); Miettinen et al. (2003)). However, it is not known which approach provides the best transformation to the cost landscape that would allow a BOA to offer the fastest convergence towards the Pareto-optimal front.
3. Most BOAs reviewed above do not impose a fixed limit on the number of training points to construct the surrogate model. The approach adopted in Emmerich et al. (2006) is to construct a separate surrogate model for each point in the population, implying that the computational cost is expected to increase after each iteration as more points are added to the population. The approach in Chugh et al. (2016) relies only on a single surrogate model where the training set has a maximum fixed size, but their approach assumes that constraint functions are computationally inexpensive. Hence, a

criterion to manage the training set that could be applied to problems with computationally expensive constraints is still an open challenge.

3. Preliminaries and background

This section introduces some definitions in Section 3.1, the Kriging model in Section Appendix A.1, and the algorithm of ParEGO is described in Section Appendix A.2.

3.1. Definitions

We focus on the following form of constrained multi-objective problem with continuous and discrete variables:

$$\begin{aligned} \text{Minimise} \quad & f_m(\mathbf{x}), && m = 1, \ldots, M; \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq c_j, && j = 1, \ldots, J; \\ & x_{r_i} \in [x_{r_i}^{\min}, x_{r_i}^{\max}], && i = 1, \ldots, n_r; \\ & x_{o_i} \in O^{(i)}, && i = 1, \ldots, n_o. \end{aligned} \tag{1}$$

The above terms are defined as follows:

1. A total of n_r continuous decision variables are denoted as $x_{r_1}, \ldots, x_{r_{n_r}}$, and $x_{r_i}^{\min}$ and $x_{r_i}^{\max}$ denote the lower and upper bound, respectively, of the ith decision variable. Discrete variables can be either of type ordered or unordered. The values that ordered decision variables are allowed to take have ordered relations (e.g. cold/mild/hot), while no such relations exist for unordered ones (e.g. colours). In this paper we only consider ordered discrete variables since the target application (see description in Section 7.1) does not have unordered ones. Let a total of n_o ordered decision variables be denoted as $x_{o_1}, \ldots, x_{o_{n_o}}$, each of which is taken from a finite domain, respectively denoted by $O^{(1)}, \ldots, O^{(n_o)}$, where $O^{(i)} \equiv \{x_{o_i}^{\min}, \ldots, x_{o_i}^{\max}\}$ for all $i = 1, \ldots, n_o$, and $x_{o_i}^{\min}$ and $x_{o_i}^{\max}$ denotes the lower and upper bound, respectively. The domain defined by all decision variables constitutes a decision variable space $\mathcal{D}$, also known as the search space. A solution to the problem in Equation 1 is represented by an n-dimensional vector that contains all decision variables, where $n = n_r + n_o$, and is denoted as $\mathbf{x} = (x_{r_1}, \ldots, x_{r_{n_r}}, x_{o_1}, \ldots, x_{o_{n_o}})^T \subset \mathcal{D}$.
2. A total of M objective functions to be minimised are defined as $f_m : \mathcal{D} \to \mathbb{R}$, $m = 1, \ldots, M$, and the multi-dimensional space that the objective functions constitute $\mathbb{R}^M$ is called the objective space.
3. A total of J constraint functions to be satisfied are defined as $g_j : \mathcal{D} \to \mathbb{R}$, $j = 1, \ldots, J$.

The corresponding objective and constraint vectors of a solution $\mathbf{x}$ are denoted by $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))^T$ and $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_J(\mathbf{x}))^T$, respectively. A point in objective space is denoted by $\mathbf{f}(\mathbf{x}) = \mathbf{z} = (z_1, \ldots, z_M)^T$, and a point in constraint space is denoted by $\mathbf{g}(\mathbf{x}) = \mathbf{z}_c = (z_{c_1}, \ldots, z_{c_J})$. In this paper the term solution and point will be used interchangeably and they both refer to a decision vector $\mathbf{x}$. A solution $\mathbf{x}$ is said to be *feasible* if $\mathbf{x} \in \mathcal{D}$ and $g_j(\mathbf{x}) \leq c_j$ for all $j = 1, \ldots, J$, where c_j is the constraint limit for the jth constraint function. The feasible set of all solutions is defined as:

$$\mathcal{F} = \{\mathbf{x} \in \mathcal{D} : g_j(\mathbf{x}) \leq c_j, j = 1, \ldots, J\}. \tag{2}$$

When multiple conflicting objectives are involved, besides each objective having its own optimal solution, there is also a set of trade-off solutions where a gain in one objective leads to a sacrifice in the other. The best trade-offs among objectives can be identified in terms of Pareto optimality. Consider the following two feasible solutions $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{F}$, and let $\mathbf{x}_1 \preceq \mathbf{x}_2$ denote that solution $\mathbf{x}_1$ dominates $\mathbf{x}_2$. This implies that solution $\mathbf{x}_1$ is no worse than $\mathbf{x}_2$ in all objectives ($f_i(\mathbf{x}_1) \not> f_i(\mathbf{x}_2)\ \forall i = 1, \ldots, M$), and $\mathbf{x}_1$ is better than $\mathbf{x}_2$ in at least one objective ($\exists j \in \{1, \ldots, M\}$ such that $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$). When comparing solutions in a set (say $\mathcal{F}' \subset \mathcal{F}$), those solutions that are not dominated by any other solution in $\mathcal{F}'$ are said to be non-dominated. The non-dominated solutions of the entire feasible set $\mathcal{F}$ form the Pareto-optimal set. In other words, a feasible solution $\mathbf{x}^* \in \mathcal{F}$ is said to be Pareto-optimal if there is no other solution in $\mathcal{F}$ that dominates it (i.e.

iff $\nexists \mathbf{x} \in \mathcal{F}$ such that $\mathbf{x} \preceq \mathbf{x}^*$). The set of all Pareto-optimal solutions is the Pareto-optimal set. $\mathbf{z}^* = \mathbf{f}(\mathbf{x}^*)$ is a Pareto-optimal objective vector, and the set that contains them all is the *Pareto-optimal front* (PF). Moreover, we assume that $\mathbf{f}$ and $\mathbf{g}$ are both expensive to evaluate and are black box functions—that is, given $\mathbf{x}$ they return $\mathbf{z}$ and $\mathbf{z}_c$.

3.2. Bayesian optimization

For dealing with expensive-to-evaluate problems some optimization algorithms adopt the *surrogate model* approach. The surrogate (or also known as metamodel or emulator) is a cheap-to-evaluate model that emulates the input-output relationship of the expensive high fidelity simulation. A search procedure that replaces the original function with the surrogate is expected to run faster, but depending on how accurate the surrogate is, there is still some necessity to conduct expensive evaluations. These evaluations could be used to improve the accuracy of the model in regions of the search space where there is lack of information, but we need to balance this with the task of finding the optimal solution.

Kriging is a type of surrogate model based on Gaussian Process theory (Krige (1951); Rasmussen & Williams (2006)) that has become very popular and is widely used for engineering applications (see e.g. Forrester et al. (2008); Wang & Shan (2007)). In particular, the *efficient global optimization* (EGO) algorithm (Jones et al. (1998)), an optimizer for single-objective unconstrained problems, uses the *design and analysis of computer experiments* (DACE) model (Sacks et al. (1989)). This is a Kriging based model parameterised by *maximum-likelihood estimation* (MLE) which, besides being capable of predicting responses at untried inputs, also quantifies the uncertainty (or variance) at those predictions through the *mean squared error* (MSE). More details about the Kriging model used in this study are found in the Appendix A.1.

EGO uses the EI infill criterion, which takes into account the uncertainty of the model to provide an estimate of where an improvement is more likely to be. Due to its success in dealing with expensive problems, EGO has been extended for handling constraints (Schonlau et al. (1998)), and later to handle multiple objectives in an algorithm known as ParEGO (Knowles (2005)). More details about EGO and how to extend it to the multi-objective domain are provided in Appendix A.2.

As reviewed in Section 2 there are many BOAs in the literature which, in general, have in common an infill criterion such as the EI combined with the PoF. Although this paper focusses on how to extend ParEGO for handling constraints and mixed-integer variables, the proposed methodology can also be applied to other BOAs.

4. Proposed methodology

4.1. Strategy to handle mixed continuous and discrete variables

Conventional ParEGO can only handle problems with a continuous search space. Problems with discrete, or a mix of continuous and discrete variables are not directly supported. Our strategy to handle such problems relies on a:

1. a direct conversion approach (Pelamatti et al. (2020)) used for Gaussian process-based surrogate models, where the Kriging correlation function is extended to handle both continuous and discrete variables (Section 4.1.1);
2. a simultaneous approach (Li et al. (2013))[1] where discrete and continuous variables are optimised together when searching the surrogate model with an evolutionary algorithm (Section 4.1.2).

For all operations within the optimization algorithm, the set of values that ordinal variables can take are mapped into an integer set by using an *integer coding*. Before a candidate design is evaluated, the discrete variables are mapped back to their original values. For instance, the set $\{10, 10.5, 11, 11.5, 12\}$ that contains all allowed values for a given discrete variable, is converted to the set $\{1, 2, 3, 4, 5\}$.

[1] In Li et al. (2013) the authors mention two general approaches, one is an hierarchical approach where discrete and continuous variables are optimized in separate "hierarchical" problems, and the other pursued in this paper is the simultaneous approach.

4.1.1. Direct conversion approach for the Kriging correlation function

The changes to the procedure used to construct a surrogate model involve the use of a direction conversion approach (Pelamatti et al. (2020)) that treats discrete variables as continuous. For this, the Kriging correlation function (Equation A.1) is modified as follows:

$$\mathcal{K}(||\mathbf{x}_a - \mathbf{x}_b||) = \exp\left(-\sum_{i=1}^{n_r}\theta_i|x_{r_i}^{(a)} - x_{r_i}^{(b)}|^2 - \sum_{i=1}^{n_o}\theta_{n_r+i}|x_{\hat{o}_i}^{(a)} - x_{\hat{o}_i}^{(b)}|^2\right), \tag{3}$$

where $x_{r_i}^{(a)}$ is the ith continuous variable of the point $\mathbf{x}_a$, and $x_{\hat{o}_i}^{(a)}$ represents the continuous variable resulting from the integer coding of the ordinal variable $x_{o_i}^{(a)}$.

4.1.2. Mixed-integer solver for searching the surrogate model

In a BOA, an evolutionary algorithm (EA) is used to search the surrogate model to find the solution that, e.g., maximises both the EI and the PoF. The simultaneous approach will be used by an optimization algorithm to search the surrogate model when both continuous and discrete variables are involved. For this task we have chosen ACROMUSE (McGinley et al. (2011)), a genetic algorithm for continuous single-objective problems, but note that any other evolutionary algorithm that relies on crossover and mutation operators (e.g. differential evolution), could be used instead.

The simultaneous approach described by Li et al. (2013) consists of using multiple specialised crossover and mutation operators in the same evolutionary algorithm. By specialised we mean that some crossover operators have been designed to operate only on continuous variables (e.g. simulated binary crossover) and may produce undesirable behaviour when applied to discrete variables. The same applies to mutation operators. The main idea is to use multiple crossover or mutation operators; each one will operate either on the continuous and discrete part of the decision vector. Li et al. (2013) have proposed an evolutionary algorithm known as MIES in the process of describing the simultaneous approach, but we do not use MIES in our study, rather we only borrow the simple concept of using multiple crossover and mutation operators to extend ACROMUSE. Nevertheless, we use the same crossover and mutation operators for discrete variables in ACROMUSE that have been integrated into MIES, but we rely on different operators for handling continuous variables. Some details about the crossover and mutation operators used by ACROMUSE in this paper are as follows:

1. Continuous variables: single-point crossover and polynomial mutation (Deb & Goyal (1996)). Single-point crossover generates two offspring from every two parents, and for every pair of parent values a and b the offspring values are $\beta a + (1-\beta)b$ and $(1-\beta)a + \beta b$, where $\beta \sim U(0,1)$.
2. Discrete variables: we use the uniform crossover (Syswerda (1989)) with a chance of 0.5 at each position, which is also called discrete crossover (Yu & Gen (2010)). The discrete mutation is borrowed from Rudolph (2005). Some details about these two discrete operators are as follows.

In the discrete crossover, for every two parent solutions selected for crossover, each variable in the offspring solution is chosen randomly from either the first or second parent solution with equal probability. The discrete mutation relies on the difference between two geometrical distributed variables, and the resulting distribution has some attractive characteristics, such as:

1. symmetry: no additional bias is introduced by the operator;
2. accessibility: it is possible to reach any point within the search space by applying the operator a finite number of times to an individual solution.

The operator has a step size parameter that controls the strength of the mutation, and more details (including a step-by-step procedure) can be found in Li et al. (2013) (Algorithm 3 page 38).

Another important aspect is the procedure used for generating the initial set of solution involving both continuous and discrete variables, which is now described. To generate the initial set of solutions we use the space-filling method LHS and propose an extension for dealing with discrete variables. First of all, each dimension of the search space is divided into N equal intervals, where N is the number of points to generate.

Only one point is allowed inside each box defined by the intervals. The procedure generates a $N \times n$ matrix where n is the number of decision variables, and each column contains the elements $0, 1, \ldots, N-1$. After all elements per column are randomly permuted, this matrix becomes a Latin hypercube and it is denoted by $\mathbf{L}$. Then, let a total of N solutions be denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_N$, and the decision variables of solution $\mathbf{x}_k$, where $k \in \{1, \ldots, N\}$, are initialised as follows:

$$\begin{aligned} x_{r_i} &= x_{r_i}^{\min} + \frac{x_{r_i}^{\max} - x_{r_i}^{\min}}{N}(U(0,1) + \mathrm{L}_{ki}),\ i = 1, \ldots, n_r, \\ x_{o_i} &= x_{o_i}^{\min} + \mathrm{L}_{ki} \mod (x_{o_i}^{\max} - x_{o_i}^{\min} + 1),\ i = 1, \ldots, n_o, \end{aligned} \tag{4}$$

where the top equation is for continuous variables, and the bottom one is for discrete. L_{ki} is the element from $\mathbf{L}$ for the kth solution and ith decision variable, $U(0,1)$ draws a number from the continuous uniform distribution bounded between 0 and 1, and mod denotes the modulo operation. Note that in Equation 4 both x_{r_i} and x_{o_i} can only take values from within their limits, that is, $x_{r_i} \in [x_{r_i}^{\min}, x_{r_i}^{\max}]$ and $x_{o_i} \in \{x_{o_i}^{\min}, \ldots, x_{o_i}^{\max}\}$.

ACROMUSE is used by ParEGO for training and searching the surrogate model; however the discrete crossover and integer mutation mentioned above only operate for the second task. Training the surrogate corresponds to the task of solving the single-objective optimization problem in (A.3) where all variables are continuous. Searching the surrogate model corresponds to the task of finding the solution that maximises the EI function (Equation A.7), and there could be a mix between continuous and discrete variables.

4.2. Choosing a subset of solutions to construct a surrogate model

The most computational demanding task of constructing the Kriging model (Appendix A.1) is to estimate $\boldsymbol{\theta}$ in the Kriging correlation function (Equation A.1) that best fits the data. The approach used requires solving the single-objective optimization problem in (A.3) by using a numerical optimization algorithm, where the inverse of $\boldsymbol{R}$ needs to be determined for each function evaluation. Given that $\boldsymbol{R}$ is a $N \times N$ dimensional matrix where N is the number of solutions, it is expected for the computational cost to increase as more solutions are added to $\mathcal{X}$. To ensure that the construction of the surrogate model does not become prohibitively expensive, in this section we discuss a computationally cheap strategy with a criterion that exploits the presence of both feasible and infeasible solutions. This builds on the findings from Chugh et al. (2016) where the simulation results indicate it is preferable to have a mix of feasible and infeasible solutions in the training set, rather than relying just on feasible solutions.

The criterion used not only preserves the best performing solutions but also those that could inform us about the location of infeasible regions across the search space. Let $N^{\max}$ denote the maximum number of solutions that can be used to construct the surrogate model, and let half of the required solutions be denoted by $H = \lceil N^{\max}/2 \rceil$, then consider the following cases with respect to the solutions in $\mathcal{X}$:

1. All solutions are feasible: select H solutions with the best fitness value; and select the remaining solutions with the shortest distance to the operating reference direction vector. For the latter, the performance vector $\mathbf{z}$ of each solution $\mathbf{x}$ is projected into the $M-1$ simplex, and then its distance to the reference direction vector is measured by the Euclidean norm (this procedure is shown mathematically in algorithm 1 lines 26-30). In this situation there are no infeasible solutions in $\mathcal{X}$, therefore the constraints can be ignored and the best performing solutions are chosen.
2. All solutions are infeasible: select H solutions with the lowest infeasibility score; and select the remaining solutions by using the criterion from point 1.
3. There is a mix of feasible and infeasible solutions. The objective here is to have a good balance of feasible and infeasible solutions in the subset. Consider the following three cases:
   (a) The number of infeasible and the number of feasible solutions are each not smaller than H: select a total of H solutions by applying the criterion from point 1 to all feasible solutions; and select the remaining $(N^{\max} - H)$ solutions from amongst all infeasible ones by choosing those with the lowest infeasibility score. Given that there is a sufficient number of solutions from both categories (feasible and infeasible) to ensure an equal distribution, the first priority is to focus on

the best performing feasible solutions, following which the remaining solutions are chosen from the infeasibility ones based on their infeasibility score (less infeasible first).

(b) The number of infeasible solutions is less than H, and the number of feasible solutions is not smaller than H: Select all infeasible solutions; and select the remaining solutions from amongst all feasible ones by applying criterion from point 1.

(c) The number of infeasible solutions is not smaller than H, and the number of feasible solutions is less than H: Select all feasible solutions; and select the remaining solutions from amongst all infeasible ones by choosing those with the lowest infeasibility score.

The steps of the above procedure are shown in algorithm 1. For this, the solutions in $\mathcal{X}$ are divided into two sets: $\mathcal{X}_f$ contains the feasible solutions, and $\mathcal{X}_i$ contains the infeasible ones. The output from the algorithm is a subset $\mathcal{X}'$ such that $\mathcal{X}' \subseteq \mathcal{X}$. A resume of the steps is as follows. The subset $\mathcal{X}'$ contains all solutions if the number of solutions in $\mathcal{X}$ is not higher than $N^{\max}$ (lines 2-3). Point 1 above corresponds to lines 4-5, where the procedure to select the best performing solutions is shown in lines 21-30. The remaining cases are as follows: point 2 above corresponds to lines 6-8; point 3a to lines 9-11; point 3b to lines 12-14; and point 3b to lines 15-17.

4.3. Constraint handling strategies

4.3.1. ParEGO with penalty function based constraint handling (ParEGO-C1)

The main idea is to transform a constrained problem into an unconstrained one by adding a penalty to infeasible solutions based on the degree of constraint violation. The information provided by infeasible solutions can be used to steer the search towards feasible and optimal regions. For instance, a solution that violates the constraints only by a small margin might be more valuable than a feasible solution with very poor performance. The approach described here is derived from Farmani & Wright (2003). The scalarised fitness of infeasible solutions is penalised according to the degree of infeasibility and the fitness of the solutions in $\mathcal{X}$. The penalty is applied in two stages. At first, the fitness of all infeasible solutions that are better than the "best" solution is set equal to the fitness of the "best" solution. The best solution is the feasible solution with the best fitness. If no feasible solutions exists in $\mathcal{X}$, the best solution is the one with least constraint violation. In the second stage, all infeasible solutions are further penalised according to their original fitness and the degree of constraint violation. More formally this approach is as follows.

We use an infeasibility score given by:

$$\dot{\xi}(\mathbf{x}) = \frac{1}{J}\sum_{j=1}^{J} v_j(\mathbf{x})/v_j^{\max}, \tag{5}$$

where $v_j^{\max}$ is the highest constraint violation for the jth constraint found with respect to all infeasible solutions in $\mathcal{X}$. Following this, the procedure identifies the best solution, as follows:

$$\mathbf{x}^* = \begin{cases} \arg\min_{\mathbf{x}\in\mathcal{X}_f} s(\mathbf{x}), \text{ if } |\mathcal{X}_f| > 0, \\ \arg\min_{\mathbf{x}\in\mathcal{X}} \dot{\xi}(\mathbf{x}), \text{ otherwise}, \end{cases} \tag{6}$$

where the first case selects the feasible solution with the best fitness, and in case there are no feasible solutions then the second case selects the least infeasible solution with respect to the modified infeasibility score (Equation 5). The fitness of all infeasible solutions are now penalised as follows. The first penalisation ensures that their fitness is not better than the best solution, that is:

$$\dot{s}(\mathbf{x}) = \begin{cases} s(\mathbf{x}^*) \text{ if } s(\mathbf{x}) < s(\mathbf{x}^*) \\ s(\mathbf{x}), \text{ otherwise} \end{cases} \tag{7}$$

where $\dot{s}(\mathbf{x})$ is the penalised fitness score of $\mathbf{x}$. To determine the second penalisation the original fitness and modified infeasibility score are normalised so that the values lie in the range between 0 and 1, as follows:

$$\begin{aligned} \bar{s}(\mathbf{x}) &= (s(\mathbf{x}) - s^{\min})/(s^{\max} - s^{\min}), \\ \bar{\xi}(\mathbf{x}) &= (\dot{\xi}(\mathbf{x}) - \dot{\xi}^{\min})/(\dot{\xi}^{\max} - \dot{\xi}^{\min}), \end{aligned} \tag{8}$$

Algorithm 1 Choose a subset of solutions to construct a surrogate model

Require: a feasible set of solutions $\mathcal{X}_f$, an infeasible set of solutions $\mathcal{X}_i$, subset size $N^{\max}$, current reference direction vector $\mathbf{d}$
Ensure: a subset $\mathcal{X}'$ of size $N^{\max}$

$H = \lceil N^{\max}/2 \rceil$ ▷ Half of the required solutions
if $|\mathcal{X}_i| + |\mathcal{X}_f| \leq N^{\max}$ **then** ▷ All solutions are selected
  $\mathcal{X}' \leftarrow \mathcal{X}_i \cup \mathcal{X}_f$
else if $|\mathcal{X}_i| = 0$ **then** ▷ All solutions are feasible
  $\mathcal{X}' \leftarrow$ ChooseBestPerformingSolutions$(\mathcal{X}_f, N^{\max}, \mathbf{d})$
else if $|\mathcal{X}_f| = 0$ **then** ▷ All solutions are infeasible
  $\mathcal{X}' \leftarrow H$ solutions from $\mathcal{X}_i$ with the smallest infeasibility score (Equation A.13)
  $\mathcal{X}' \leftarrow \mathcal{X}' \cup$ ChooseBestPerformingSolutions$(\mathcal{X}_i \setminus \mathcal{X}', N^{\max} - |\mathcal{X}'|, \mathbf{d})$
else if $|\mathcal{X}_i| \geq H$ and $|\mathcal{X}_f| \geq H$ **then** ▷ Both $\mathcal{X}_f$ and $\mathcal{X}_i$ have at least H solutions
  $\mathcal{X}' \leftarrow$ ChooseBestPerformingSolutions$(\mathcal{X}_f, H, \mathbf{d})$
  $\mathcal{X}'' \leftarrow N^{\max} - |\mathcal{X}'|$ solutions from $\mathcal{X}_i$ with the smallest infeasibility score (Equation A.13)
  $\mathcal{X}' \leftarrow \mathcal{X}' \cup \mathcal{X}''$
else if $|\mathcal{X}_i| < H$ and $|\mathcal{X}_f| \geq H$ **then** ▷ $\mathcal{X}_i$ has less than H solutions
  $\mathcal{X}' \leftarrow \mathcal{X}_i$
  $\mathcal{X}' \leftarrow \mathcal{X}' \cup$ ChooseBestPerformingSolutions$(\mathcal{X}_f, N^{\max} - |\mathcal{X}'|, \mathbf{d})$
else if $|\mathcal{X}_i| \geq H$ and $|\mathcal{X}_f| < H$ **then** ▷ $\mathcal{X}_f$ has less than H solutions
  $\mathcal{X}' \leftarrow \mathcal{X}_f$
  $\mathcal{X}'' \leftarrow N^{\max} - |\mathcal{X}'|$ solutions from $\mathcal{X}_i$ with the smallest infeasibility score (Equation A.13)
  $\mathcal{X}' \leftarrow \mathcal{X}' \cup \mathcal{X}''$
else
  $\mathcal{X}' \leftarrow \mathcal{X}_i \cup \mathcal{X}_f$ ▷ This point should never be reached but is kept for completion
end if

procedure ChooseBestPerformingSolutions($\mathcal{X}$,N,$\mathbf{d}$)
  $\mathcal{X}' \leftarrow \lceil N/2 \rceil$ solutions from $\mathcal{X}$ with the best fitness value (Equation A.11)
  $\mathcal{X}'' \leftarrow \mathcal{X} \setminus \mathcal{X}'$
  for all $\mathbf{x} \in \mathcal{X}''$ **do**
    $\hat{\mathbf{z}}(\mathbf{x}) \leftarrow$ project $\mathbf{z}(\mathbf{x})$ to the $M-1$ simplex, implying that $\hat{\mathbf{z}}(\mathbf{x}) = \mathbf{z}(\mathbf{x})||\mathbf{z}(\mathbf{x})||_1^{-1}$
    $\Delta(\mathbf{x}, \mathbf{d}) \leftarrow ||\hat{\mathbf{z}}(\mathbf{x}) - \mathbf{d}||_2$
  end for
  $\mathcal{X}''' \leftarrow N - \lceil N/2 \rceil$ solutions from $\mathcal{X}''$ with the smallest Δ distance
  $\mathcal{X}' \leftarrow \mathcal{X}' \cup \mathcal{X}'''$
  return $\mathcal{X}'$
end procedure

where $s^{\min}$ and $s^{\max}$ are the lowest and highest original fitness of all solutions, and $\dot{\xi}^{\min}$ and $\dot{\xi}^{\max}$ are the lowest and highest modified infeasibility score of all infeasible solutions. Finally, all infeasible solutions are further penalised according to their original cost function and the degree of constraint violation as follows:

$$\ddot{s}(\mathbf{x}) = \dot{s}(\mathbf{x}) + \frac{e^{2(\bar{s}(\mathbf{x})+\bar{\xi}(\mathbf{x}))-1}}{e^2-1}, \tag{9}$$

and their current fitness is replaced by $\ddot{s}(\mathbf{x})$. The exponential function in Equation 9 ensures that solutions of low infeasibility get a slight reduction in the rate of penalty, and this helps to maintain the fitness of solutions that slightly violated the constraints.

ParEGO with penalty-based constraint handling is now referred to as ParEGO-C1 for short, and a pseudo-code is shown in algorithm 2. The general steps are as follows. An initial set of solutions is generated by using some space filling design strategy (line 1), and once all solutions are evaluated the objective and constraints values are stored respectively in $\mathcal{Z}$ and $\mathcal{G}$ (lines 2-3). Following the construction of a set of reference direction vectors $\mathbf{D}$ (line 4), an iterative procedure starts where the reference directions are processed one at the time (line 7). Each time all reference directions are visited, the set $\mathbf{D}$ is shuffled (line 6). This is to avoid any bias due to using the same sequence of reference directions repeatedly during the optimization process. A scalarised fitness value is determined for each solution and stored in $\mathcal{S}$ (line 9). The fitness values are updated by penalising infeasible solutions according to their degree of constraint violation (line 10). Following the selection of a subset of solutions (line 11), these are used as training data to construct a surrogate model (line 12). The surrogate model is used by the EI function and a solution is found by maximizing it (line 13). Following its evaluation, the new solution is added to $\mathcal{X}$, its objective and constraint values are store respectively in $\mathcal{Z}$ and $\mathcal{G}$, and this completes one iteration (line 14).

Algorithm 2 ParEGO with penalty function based constraint handling (ParEGO-C1)

Parameters: initial set size N^{init}, surrogate model maximum set size $N^{\max}$
1: $\mathcal{X} \leftarrow$ generate initial set of solutions with size N^{init}
2: $\mathcal{Z} \leftarrow \mathbf{f}(\mathcal{X})$ ▷ evaluate the objectives of the solutions in the initial set
3: $\mathcal{G} \leftarrow \mathbf{g}(\mathcal{X})$ ▷ evaluate the constraints of the solutions in the initial set
4: $\mathbf{D} \leftarrow$ set of all reference direction vectors (Equation A.10)
5: **while** termination criteria not satisfied **do**
6: Shuffle the set $\mathbf{D}$
7: **for all** $\mathbf{d} \in \mathbf{D}$ **do**
8: update objective lower and upper bounds (Equation A.8)
9: $\mathcal{S} \leftarrow$ for each solution calculate a scalar fitness value (Equation A.11)
10: $\mathcal{S}' \leftarrow$ penalise infeasible solutions according to their degree of constraint violation (Equation 9)
11: $\mathcal{X}' \leftarrow$ select a subset of solutions from $\mathcal{X}$ with maximum size $N^{\max}$ (Section 4.2)
12: $\hat{s} \leftarrow$ fit a surrogate model to $\mathcal{X}'$ and corresponding fitness values in $\mathcal{S}'$
13: $\mathbf{x}^{\text{new}} \leftarrow$ find a solution that maximises the EI (Equation A.7)
14: $\mathcal{X} \leftarrow \mathcal{X} \cup \mathbf{x}^{\text{new}}$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathbf{f}(\mathbf{x}^{\text{new}})$ and $\mathcal{G} \leftarrow \mathcal{G} \cup \mathbf{g}(\mathbf{x}^{\text{new}})$ ▷ evaluate and store new solution
15: **end for**
16: **end while**

4.3.2. ParEGO with constrained expected improvement based constraint handling (ParEGO-C2)

The EI formulation in Equation A.7 is extended to handle constraints. A surrogate model of the constraint function can be used to influence the expectation of improvement, that is: in case the surrogate indicates a constraint violation with a low error, the EI should have a low value; and if the surrogate error is high, then we cannot be sure if the constraint is violated, and therefore the EI should be higher. Based on this premise, the approach described in Forrester & Keane (2009) calculates the probability that the prediction of the constraint model is smaller than the constraint limit, that is, the probability that the

constraint is met. More formally, let the mean and variance of the model at $\mathbf{x}$ be given by $\hat{g}(\mathbf{x})$ and $\hat{\varepsilon}^2(\mathbf{x})$, respectively. The probability that the constraint is met is given by:

$$P[g(\mathbf{x}) < c] = \Phi\left(\frac{c - \hat{g}(\mathbf{x})}{\hat{\varepsilon}(\mathbf{x})}\right), \tag{10}$$

where $g(x)$ is the constraint function, and c is the constraint limit. The constraint EI is obtained by multiplying $E[I(\mathbf{x})]$ by $P[g(\mathbf{x}) < c]$, and this can be easily extended for dealing with multiple (J) constraints as follows:

$$E[I(\mathbf{x})]\Pi_{j=1}^{J} P[g_j(\mathbf{x}) < c_j]. \tag{11}$$

In problems with a constrained optima (different from the unconstrained optima), most improvement reported by the EI function (Equation A.7) corresponds to infeasible regions. This implies that the improvement in the feasible region is likely to be close to zero, which makes it more difficult to search for the constrained optima when using the constraint EI function (Equation 11). To address this we have adopted a recommendation by Bagheri et al. (2017), which consists of replacing the current best "known" function value in Equation A.7 by the feasible solution with the best fitness, or the least infeasible solution in case all solutions are infeasible. For the latter, the chosen solution has the lowest infeasibility score, as determined by Equation A.13.

ParEGO with probabilistic-based constraint handling is now referred to as ParEGO-C2 for short, and a pseudo-code is shown in algorithm 3. Algorithm 2 and 3 share many steps in common, hence, we focus only on the differences between the two algorithms. In algorithm 3 the fitness values are no longer penalised as in line 10 of algorithm 2. Instead, a surrogate model is constructed for each constraint function (line 12) based on the subset of solutions chosen in line 10. All learnt surrogate models are combined into the constraint EI function, and following its maximization, a new solution is found (line 13). Two points that are noteworthy:

1. A total of J surrogate models are constructed in line 12 algorithm 3, and this task is expected to be computationally more expensive than the approach used that penalises the fitness values in line 10 algorithm 2.
2. The surrogate modelling task in line 12 algorithm 3 relies on the subset $\mathcal{X}'$. This prevents the construction of the surrogates from becoming prohibitively expensive since the number of solutions in $\mathcal{X}$ can be too high. However, note that the approach in line 10 algorithm 2 relies on the entire solution set $\mathcal{X}$ which could be advantageous in case the subset $\mathcal{X}'$ lacks important information about where infeasible solutions are meant to exist in the search space.

5. Empirical analysis setup

This section provides details about the parameter setup for ParEGO and NSGA-II (Deb et al. (2002)), and also about the quality indicators used to evaluate the solutions sets obtained by the optimization algorithms. Note that NSGA-II is chosen in this study to be compared with ParEGO due to its current popularity for the study of multi-objective internal combustion engine design problems, as in Corre et al. (2019); DErrico et al. (2011); Lotfan et al. (2016). Some details about NSGA-II including its implementation and how it handles constraints are provided in Appendix A.3.

5.1. Optimisation algorithms setup

The budget for both optimization algorithms is set to $N = 500$ function evaluations, which includes the evaluation of the initial set of solutions. Note that relying on 500 or fewer function evaluations is considered the general agreement on what constitutes a constrained budget (Chugh et al. (2019); Keane (2006); Knowles & Hughes (2005)). All solutions evaluated during the optimization run are saved in an archive, and each run is repeated 21 times with a different random seed. The initial set for each optimization algorithm is generated by LHS, and the set size is fixed to $N^{\text{init}} = 100$. The justification for this lies in the need to validate the optimization model used by the internal combustion engine design problem, which is the primary focus

Algorithm 3 ParEGO with constrained EI based constraint handling (ParEGO-C2)

Parameters: initial set size N^{init}, surrogate model maximum set size $N^{\max}$
$\mathcal{X} \leftarrow$ generate initial set of solutions with size N^{init}
$\mathcal{Z} \leftarrow \mathbf{f}(\mathcal{X})$ ▷ evaluate the objectives of the solutions in the initial set
$\mathcal{G} \leftarrow \mathbf{g}(\mathcal{X})$ ▷ evaluate the constraints of the solutions in the initial set
$\mathbf{D} \leftarrow$ set of all reference direction vectors (Equation A.10)
while termination criteria not satisfied **do**
    Shuffle the set $\mathbf{D}$
    for all $\mathbf{d} \in \mathbf{D}$ **do**
        update objective lower and upper bounds (Equation A.8)
        $\mathcal{S} \leftarrow$ for each solution calculate a scalar fitness value (Equation A.11)
        $\mathcal{X}' \leftarrow$ select a subset of solutions from $\mathcal{X}$ with maximum size $N^{\max}$ (Section 4.2)
        $\hat{s} \leftarrow$ fit a surrogate model to $\mathcal{X}'$ and corresponding fitness values in $\mathcal{S}$
        $\{\hat{g}_j\}_{j=1}^{J} \leftarrow$ fit a separate surrogate model to each constraint function w.r.t. $\mathcal{X}'$ and $\mathcal{G}$
        $\mathbf{x}^{\text{new}} \leftarrow$ find a solution that maximises the constraint EI (Equation 11)
        $\mathcal{X} \leftarrow \mathcal{X} \cup \mathbf{x}^{\text{new}}$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathbf{f}(\mathbf{x}^{\text{new}})$ and $\mathcal{G} \leftarrow \mathcal{G} \cup \mathbf{g}(\mathbf{x}^{\text{new}})$ ▷ evaluate and store new solution
    end for
end while

of this study. It was agreed that 100 designs provide sufficient coverage on a 7 decision variable problem for validation purposes. For consistency, the initial set used by the benchmark problems was also generated by LHS with a set size of 100.

In NSGA-II, the size of both the parent and offspring populations are set to $N^P = N^{\text{init}}/2$, and the number of generations is given by $(N - N^P)/N^P$. The crossover and mutation operators for continuous variables are simulated binary crossover (Deb & Kumar (1995)) and polynomial mutation (Deb & Goyal (1996)). The distribution index for simulated binary crossover and polynomial mutation is set to 15 and 20, respectively. For the discrete variables we use the same discrete crossover and discrete mutation operators mentioned in Section 4.1.2 for ACROMUSE. The step size parameter in the discrete mutation operators is set to be equal to 10% of the length defined by the upper and lower bound of each decision variable. The probability of crossover and mutation are respectively set to 90% and 10%. The same probability of crossover is used by Knowles (2005) when comparing NSGA-II with ParEGO. The probability of mutation is suggested by Li et al. (2013) for dealing with mixed integer problems.

Specifically for ParEGO, the number of reference direction vectors is set to 10 (this ensures that all direction vectors are processed exactly 50 times for the given budget, and is a very close number to the 11 chosen by Knowles (2005)). To construct the surrogate model we used a maximum of $N^{\max} = 100$ solutions. ACROMUSE is used for training and searching the surrogate model. Training the surrogate corresponds to the task of solving the single-objective optimization problem in (A.3). Searching the surrogate corresponds to the task of finding the solution that maximises the EI function. This either corresponds to Equation A.7 in case ParEGO-C1 is used (algorithm 2), or Equation 11 in case ParEGO-C2 is used (algorithm 3). The following setup is used by ACROMUSE. The crossover and mutation operators for continuous variable are single-point crossover and polynomial mutation, and for discrete variables we use the same operators mentioned above for NSGA-II. The probability of crossover and mutation is the same as defined for NSGA-II. More details about the crossover and mutation operators in ACROMUSE are found in Section 4.1.2. Based on several experiments we found that setting an initial population size to $20n$, where n is the number of decision variables, provides the best convergence. The termination criterion for ACROMUSE is as follows:

1. Determine the difference between the current best fitness in the population and the best fitness of the previous generation, and store the result in an archive;
2. Determine the mean of the last 20 elements in the archive, and terminate the optimization run if the result is below a very small number (i.e. 10^{-6});
3. To ensure that the optimization terminates in case the above condition cannot be satisfied, a maximum

number of generations is set to $100n$.

5.2. Quality indicators

There are several quality indicators in the literature for comparing the performance of optimization algorithms (Audet et al. (2021)), and we have used the following ones in this study:

1. The hypervolume indicator is used to measure the convergence and diversity across the Pareto front, and its exact value is determined by a dimension-sweep algorithm (Fonseca et al. (2006)). The hypervolume is calculated with respect to a non-dominated set that is identified from the archive that contains all evaluated solutions found during an optimization run. In order to show the hypervolume progress during an optimization run, a non-dominated set is determined after each evaluation, which serves as an input to the dimension-sweep algorithm. In some optimization runs the non-dominated solutions may not include any feasible solutions, and this is more likely to happen during the first few evaluations. In case there are no feasible solutions in a solution set then the hypervolume is calculated with respect to all infeasible non-dominated solutions, in order to provide some insight into the dynamics of the optimization algorithms. To test the statistical significance of the results we employ the Wilcoxon's rank-sum test (Conover (1999)) with a significance level of 5%. This provides an indication of whether the hypervolume performance of an algorithm is significantly different from another. In addition, when two algorithms are compared, we will state the *relative percentage difference* (RPD)[2], or simply percentage improvement, with respect to their median hypervolume values across the 21 replications. The reference point used by the hypervolume computation is $\{-0.1, 2630.0\}$, $\{7100, 1700\}$ and $\{0.9035, 0.67\}$, corresponding to the OSY, speed reducer, and engine design problems, respectively. The hypervolume values are then normalised in the range between 0 and 1, and for this we divide it by 7.15×10^5, 4.5×10^6 and 120, corresponding to the OSY, speed reducer and engine design problems, respectively. These reference points have been obtained by taking the worst case observed across the union of all runs.
2. We will show the differences between the empirical attainment functions (EAFs) (López-Ibáñez et al. (2010)) when comparing two algorithms. This provides visual information of where an algorithm has done better than the other across the Pareto front by considering multiple optimization runs. It also shows the best, median and worst attainment surfaces, corresponding to the lines in the bottom, centre and top, respectively (assuming minimisation).
3. The performance of the constraint handling approaches is evaluated during the optimization run by showing the infeasibility score (Equation 5), and the number of feasible solutions that have been found. We have observed that the infeasibility score can be very noisy between evaluations, and as a smoother, we employ a moving mean with a sliding window of length 10. The mean of each point is calculated over the neighbouring points, and the sliding window is centred in the current and previous elements. Following the smoothing of each optimization run, the mean and standard deviation is then taken. A similar approach was used for the number of feasible solutions, where each point along the optimization run gives the number of feasible solutions found in the previous 10 evaluations, including the current point.
4. We use a heatmap to show which constraints are more likely to be violated during the optimization run. Each row in the heatmap represents a constraint and the columns correspond to the evaluations across the optimization run. To compute the heatmap each constraint is processed separately as follows. At any given evaluation during the optimization run a constraint may have been violated (1) or not (0). We take the average across all runs to calculate a probability that the constraint is likely to be violated, for instance, consider a total of 5 runs then the vector $(0, 1, 0, 1, 1)$ indicates that the constraint has been violated in the second, fourth and fifth run, but it is not violated in the first and third runs. The average of the vector gives 0.6, which indicates that the probability of violating the constraint at this

[2]The relative percentage difference between two numbers v_1 and v_2 is determined by taking their absolute difference divided by their arithmetic mean, i.e., $|v_1 - v_2|/[(v_1 + v_2)/2] \times 100$.

Table 1: Comparison of subset selection strategies on the performance of ParEGO-C1 and ParEGO-C2 when applied to the OSY and speed reducer problems. The performance is measured by the hypervolume indicator (the higher the better). The statistics have been determined based on 21 runs. The term Std is an abbreviation for standard deviation, Max for maximum, and Min for minimum. Results are shown to 4 significant figures, and the best ones have been underlined.

Hypervolume	OSY				Speed reducer			
Optimizer	Median	Mean (Std)	Max	Min	Median	Mean (Std)	Max	Min
ParEGO-C1(PA)	0.7257	0.6709 (0.1693)	0.8982	0.3090	0.9359	0.9342 (0.0058)	0.9413	0.9192
ParEGO-C1(BP)	0.6933 [≈]	0.6656 (0.1331)	0.8496	0.3504	0.9359 [≈]	0.9357 (0.0034)	0.9410	0.9295
ParEGO-C1(RD)	0.6366 [≈]	0.6504 (0.1187)	0.8736	0.3983	0.9349 [≈]	0.9348 (0.0035)	0.9397	0.9240
ParEGO-C2(PA)	0.9434	0.9450 (0.0119)	0.9657	0.9282	0.9397	0.9399 (0.0029)	0.9460	0.9353
ParEGO-C2(BP)	0.9113 [-]	0.9158 (0.0146)	0.9461	0.8920	0.9409 [≈]	0.9408 (0.0019)	0.9444	0.9369
ParEGO-C2(RD)	0.9325 [-]	0.9324 (0.0125)	0.9554	0.9118	0.9390 [≈]	0.9391 (0.0033)	0.9483	0.9343

particular evaluation is 60%. For smoothing purposes, these probabilities are then averaged across 10 consecutive evaluations, implying that there is a total 50 cells in the heatmap shows across the x-axis corresponding to a total of 500 function evaluations.

6. Optimization of benchmark problems

6.1. Test suite

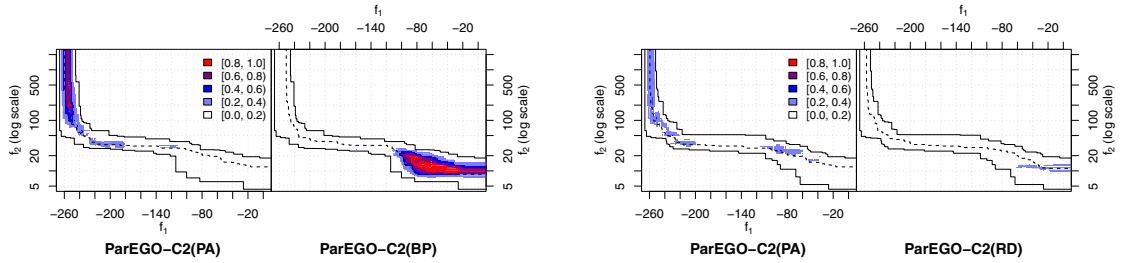
The constrained ParEGO proposed in this paper is intended to be used in solving expensive constrained mixed-integer problems, such as the target *internal combustion engine* (ICE) problem (investigated in Section 7.1), which has 4 ordinal plus 3 continuous decision variables, 2 objectives and 5 constraints. However an obstacle to comprehensive analysis of such genuinely expensive problems is that the computational budget required for extensive empirical experimentation is infeasible. To conduct this type of analysis, we require benchmark problems with comparatively low evaluation times. Unfortunately there are very few benchmark problems in existence with similar characteristics to the ICE problem. To the authors' knowledge, there is only one constrained mixed-integer benchmark problem in existence and one further constrained (but continuous variable) problem which has a similar number of constraints to the ICE problem.

The first is a quasi-real-world problem known as the speed reducer (Gunawan et al. (2003)). There are several formulations available for this problem in the literature, including, Coello & Pulido (2005) and Gong et al. (2009). The problem has two objectives, seven decision variables and eleven constraints. The decision variables are all continuous apart from the third decision variable which is discrete (i.e. x_3 in Equation B.2). The problem formulation is available in Equation B.2 in Appendix B. Based on our own numerical experiments, we discovered that the fourth, fifth, sixth and eleventh constraints are not active, which means that there are only seven active constraints in this problem. The second problem is known as OSY (Osyczka & Kundu (1995)) and it is a bi-objective optimization problem with six decision variables and six constraints. The decision variables are all continuous, and four of the constraints are linear. The problem formulation is available in Equation B.1 in Appendix B.

6.2. Comparative analysis between subset selection strategies

In this section we study the impact that different strategies for selecting a subset of solutions for constructing the surrogate model have on the performance of ParEGO-C1 and ParEGO-C2 when applied to the OSY and speed reducer problems. The criterion adopted by the subset selection strategies considered is as follows:

1. Proposed strategy in algorithm 1 (PA);
2. Best performing solutions (BP). In this case the solutions are chosen based on the procedure ChooseBestPerformingSolutions in algorithm 1. This criterion only takes into account the fitness of the solutions (as determined by Equation A.11), and their constraint values are not taken into account. First, half of the allowed number of solutions with the best fitness value are selected, and second, select the remaining solutions with the shortest distance to the reference direction vector;



(a) PA vs. BP (b) PA vs. RD

Figure 1: Comparison of subset selection strategies for ParEGO-C2 by showing the differences between the empirical attainment functions (EAFs) of the feasible non-dominated solutions for a total of 21 runs on the OSY problem. For each subfigure, the plot in the left highlights the differences in favour of algorithm 1, and the plot in the right highlights the differences in favour of algorithm 2. The colour level encodes the magnitude of the observed differences. The lines in the left, centre and right correspond to the best, median and worst attainment surfaces, respectively.

3. Random (RD). The order of all solutions is shuffled and the first $N^{\max}$ solutions are selected.

Table 1 shows the hypervolume obtained by ParEGO-C1 and ParEGO-C2 with different subset selection strategies. Based on the Wilcoxon's rank-sum test results, we use the following notation in Table 1, where [-], [+] and [≈] corresponds respectively to significantly worse than, better than, and equal to PA. Consider the following observations:

1. OSY: PA reports the best median hypervolume performance for both ParEGO-C1 and ParEGO-C2. For ParEGO-C1, the percentage improvement of PA over BP is 4.57%, and PA over RD is 13.09%. For ParEGO-C2, the percentage improvement of PA over BP is 3.46%, and PA over RD is 1.16%. The results for ParEGO-C2 are statistically significant, but the results for ParEGO-C1 are not.
2. Speed reducer: BP reports the best median hypervolume performance for both ParEGO-C1 and ParEGO-C2. The percentage improvement of BP over PA is only 0.001% and 0.13% for ParEGO-C1 and ParEGO-C2, respectively. PA performs marginally better than RD with a percentage improvement of just 0.1% and 0.07%, for ParEGO-C1 and ParEGO-C2, respectively. These results are not statistically significant.

So far we have compared the algorithms by using the hypervolume indicator, and these are now compared by visualising the differences between their EAFs. This is only shown for the statistical significant results, that is, it considers only the comparison between PA and BP with ParEGO-C2 for the OSY problem. Figure 1 shows the EAFs differences and their magnitude is encoded by the colour level. The obtained results in Figure 1a show that PA has better attainment on the upper left region of the PF when compared with BP, but BP has better attainment across the lower left region of the PF. A similar trend is captured in Figure 1b between PA and RD, but the region attained by RD is much smaller than the region attained by BP.

Based on the above results, PA has better hypervolume when compared with BP and PA for the OSY problem, although the results are not statistically significant for ParEGO-C1, but they are for ParEGO-C2. The EAFs indicates that this overall improvement comes at the cost of some regions across the PF over others. For the speed reducer problem the results are not statistical significant, meaning that for this problem the subset selection strategies have a low influence on the performance of the algorithms. These results suggest that ensuring a good balance between feasible and infeasible solutions can be better than simply relying on their performance. In the following section, the constrained ParEGO algorithms with PA will be compared to NSGA-II.

Table 2: Comparison between ParEGO-C1, ParEGO-C2 and NSGA-II when applied to the OSY and speed reducer problems. The performance is measured by the hypervolume indicator (the higher the better). The first three rows show the performance obtained after 500 function evaluations. The second last row shows the performance obtained by NSGA-II after 10000 function evaluations. The last row shows the number of function evaluations required by each successful run from NSGA-II to achieve the median performance obtained by ParEGO-C2, and in the same row, the numbers between {∗} indicate the number of successful runs out of 21 that have obtained a better hypervolume value than ParEGO-C2. The statistics have been determined based on 21 runs. The term Std is an abbreviation for standard deviation, Max for maximum, and Min for minimum. Results are shown to 4 significant figures, and the best ones have been underlined.

Hypervolume	OSY				Speed reducer			
Optimizer	Median	Mean (Std)	Max	Min	Median	Mean (Std)	Max	Min
ParEGO-C1	0.7257	0.6709 (0.1693)	0.8982	0.3090	0.9359	0.9342 (0.0058)	0.9413	0.9192
ParEGO-C2	<u>0.9434</u>	<u>0.9450</u> (<u>0.0119</u>)	<u>0.9657</u>	<u>0.9282</u>	<u>0.9397</u>	<u>0.9399</u> (<u>0.0029</u>)	<u>0.9460</u>	<u>0.9353</u>
NSGA-II (500)	0.6840	0.6593 (0.1576)	0.8850	0.3549	0.8799	0.8593 (0.0663)	0.9269	0.6764
NSGA-II (10000)	0.9001	0.9037 (0.0962)	0.9994	0.5373	0.9515	0.9465 (0.0138)	0.9537	0.9055
	2265 {6}	2315 (765)	3190	1413	1798 {18}	2268 (1745)	7944	781

Table 3: p-values obtained by the Wilcoxon's rank-sum test. A p-value lower than 0.05 (corresponding to a significance level of 5%) indicates rejection of the null hypothesis that the two samples being compared have equal medians. The values above and below the main diagonal corresponds to the OSY and speed reducer problems, respectively. The numbers 500 and 10000 after NSGA-II indicate the number of function evaluations after which the results have been obtained.

Optimizer	ParEGO-C1	ParEGO-C2	NSGA-II (500)	NSGA-II (10000)
ParEGO-C1	—	3.13×10^{-8}	0.7247	9.32×10^{-7}
ParEGO-C2	6.23×10^{-4}	—	3.13×10^{-8}	0.01931
NSGA-II (500)	8.41×10^{-8}	3.13×10^{-8}	—	7.21×10^{-7}
NSGA-II (10000)	2.66×10^{-5}	7.05×10^{-5}	1.67×10^{-7}	—

6.3. Comparative analysis between proposed constrained ParEGO and NSGA-II

This section presents results for the constraint ParEGO algorithms (ParEGO-C1 and ParEGO-C2), and NSGA-II when applied to the OSY and speed reducer problems. Both ParEGO algorithms use the subset selection strategy PA.

Table 2 shows a comparison between the algorithms in term of hypervolume. In the third row, the results obtained for NSGA-II correspond to 500 function evaluations, and the results in the second last row correspond to a very generous 10000 function evaluations. For now we will only focus on the 500 function evaluations case. ParEGO-C2 outperforms ParEGO-C1, with a percentage improvement of 26.1% for the OSY problem, and 0.42% for the speed reducer problem. The percentage improvement of ParEGO-C2 and ParEGO-C1 over NSGA-II is respectively 31.9% and 5.9% for the OSY problem, and respectively 6.57% and 6.17% for the speed reducer problem. Results are statistically significant (see Table 3), with the only exception of ParEGO-C1 versus NSGA-II for the OSY problem.

Figures 2 and 3 show the evolution of three indicators along the optimization run, and also the EAFs differences between the algorithms for the OSY and speed reducer problems, respectively. The indicators are the hypervolume, infeasibility score (Equation 5), and number of feasible solutions found during the optimization run. In addition, the three bottom subfigures in Figures 2 and 3, show all solutions evaluated during a single optimization run, and the chosen run has the closest hypervolume value to the median from a total of 21 runs. Consider the following observations:

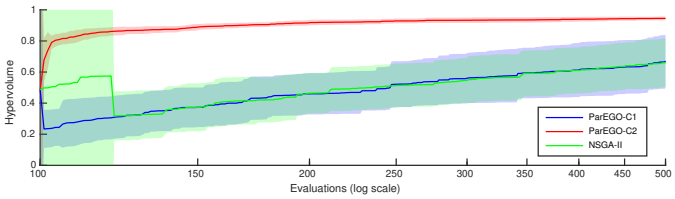
1. ParEGO-C2 shows faster convergence in terms of hypervolume when compared with ParEGO-C1 and NSGA-II (Figures 2a and 3a).
   In addition, ParEGO-C2 has the lowest standard deviation when compared with ParEGO-C1 and NSGA-II for both problems, while NSGA-II shows the highest. The relatively high standard deviation by NSGA-II during first ≈25 evaluations is caused by the fact that some optimization runs have not found feasible solutions up-to this point, and in case there are no feasible solutions then the

hypervolume is calculated with respect to the infeasible solutions, which may cause the hypervolume to show unrealistic high performance.

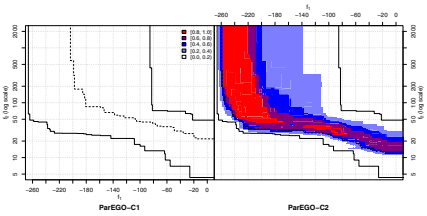
2. The optimizers show different trends when in comes to the infeasibility score and number of feasible solutions found during the optimization run as follows:
   (a) For the OSY problem in Figure 2c, ParEGO-C2 has the lowest infeasibility score during the first ≈30 evaluations, and subsequently reports the highest infeasibility score including the highest standard deviation. Whereas the infeasibility score of ParEGO-C1 and NSGA-II decreases along the optimization run.
   (b) For the speed reducer problem, the infeasibility score of ParEGO-C2 remains comparatively low across the entire optimization run when compared with the other optimizers as shown in Figure 3c, but the infeasibility score of ParEGO-C1 remains very high when compared with NSGA-II and ParEGO-C2 during the entire optimization run.
3. All evaluated solutions in a single optimization run are shown in the bottom plots of Figures 2 and 3, corresponding to the OSY and speed reducer problems, respectively. Although it is true that variance can be very high across different optimization runs, especially for NSGA-II, we have observed that these solutions capture the general trend across most runs. Consider the following observations:
   (a) Most solutions found by ParEGO-C2 for the speed reducer problem are feasible (Figure 3h). In the OSY problem, there is an area in the objective space (top-left region in Figure 2h) with many infeasible solutions. These solutions are responsible for the high infeasibility score shown by ParEGO-C2 and have mostly been found after evaluation ≈40.
   (b) ParEGO-C1 found many feasible solution when compared with the other algorithms as shown in Figure 2g for the OSY problem, but many are found in a sub-optimal region when compared with those found by ParEGO-C2. In the speed reducer problem, ParEGO-C1 found considerably less feasible solutions than ParEGO-C2 (Figure 3g), and there are many infeasible solutions found that are in sub-optimal regions in the objective space far from the edges of the PF.

The probability of violating an individual constraint along the optimization run is shown in Figures 4 and 5, corresponding to the OSY and speed reducer problems, respectively. The first 100 evaluations shows the evaluation of the initial set of solutions which is common for all optimization algorithms. For the OSY problem, g_2 and g_5 have the highest probability of being violated. This is evident during the evaluation of the initial set of solutions but also in subsequent evaluations, and in particular for ParEGO-C2 as shown in Figure 4b. For the speed reducer problem, g_7 is the most active constraint, while g_9 and g_{10} are the second most active constraints. This is more evident for ParEGO-C1 as shown in Figure 5a than it is for ParEGO-C2 and NSGA-II. Constraints such as g_4, g_5, g_6 and g_{11} are never active, and others such as g_1, g_2 and g_3, and g_8, show very low level of activity in comparison. This indicates that out of 11 constraints there are only 3 (i.e. g_7, g_9 and g_{10}) that seem to be posing difficulties to the optimization algorithms. In contrast, for the OSY problem there are at least 4 out of 6 constraints (i.e. g_2, g_3, g_5 and g_6) that seem to be posing difficulties to the optimization algorithms (in particular to ParEGO-C2). The above results suggest that the constraints in the speed reducer problem were less active when compared with the constraints from the OSY problem. In such a scenario it is expected for a subset selection strategy that takes into account the constraint violation of the solutions to show less impact. This is corroborated by the lack of statistical significance shown by the results when comparing the subset selection strategies in Table 1 for the speed reducer problem.

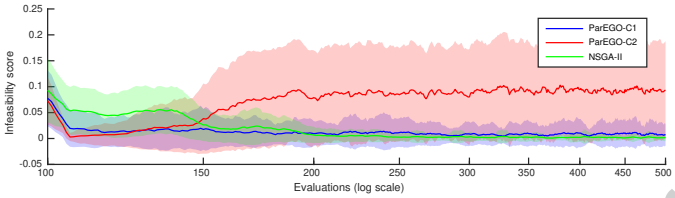
So far we have only discussed results obtained by the optimizers up to a maximum of 500 function evaluations. For this tight budget it is clear that the two ParEGO algorithms, in particular ParEGO-C2, have better performance than NSGA-II. We therefore ask the questions how many more function evaluations would be required for NSGA-II to show a similar performance to that obtained by ParEGO-C2, and how good are the results obtained by ParEGO-C2? To answer these two questions, the second last row in Table 2 shows the hypervolume obtained by NSGA-II after 10000 function evaluations (a prohibitively expensive budget for many real-world applications, in particular ICE design problems), and the last row in Table 2 shows the number of function evaluations required by each optimization run of NSGA-II to obtain a better hypervolume value than the median performance of ParEGO-C2. In that:



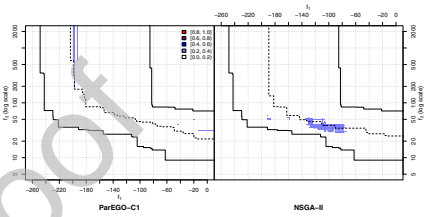
(a) Hypervolume



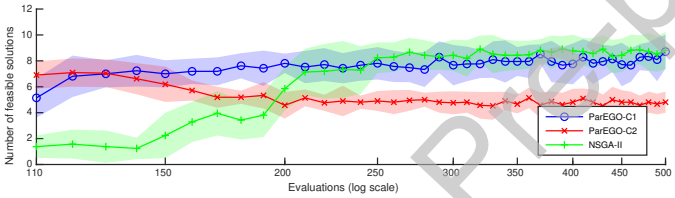
(b) EAFs: ParEGO-C1 vs. ParEGO-C2



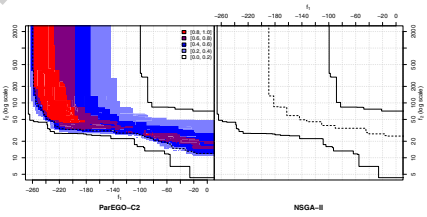
(c) Infeasibility score (Equation 5)



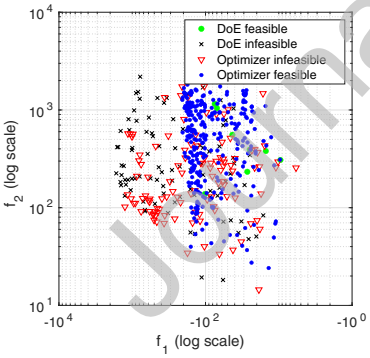
(d) EAFs: ParEGO-C1 vs. NSGA-II



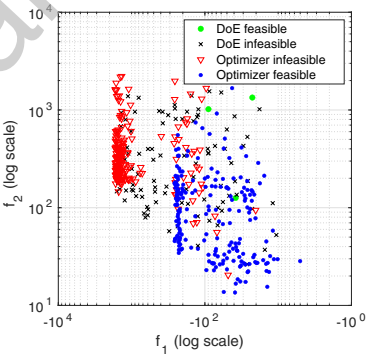
(e) Number of feasible solutions



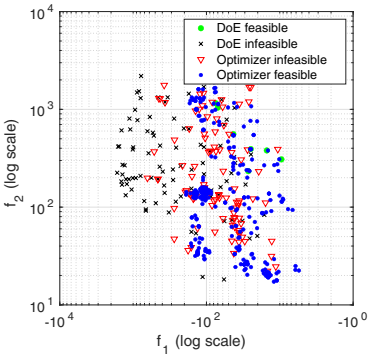
(f) EAFs: ParEGO-C2 vs. NSGA-II



(g) ParEGO-C1

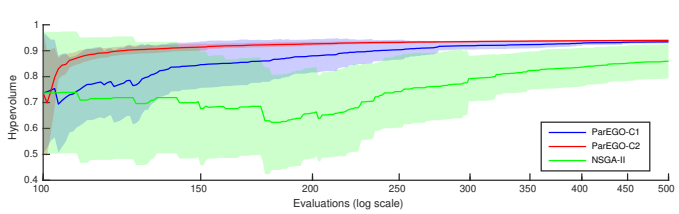


(h) ParEGO-C2

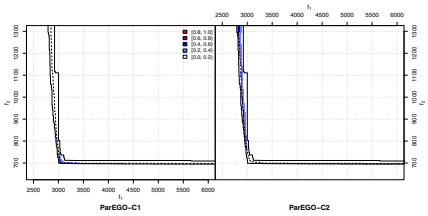


(i) NSGA-II

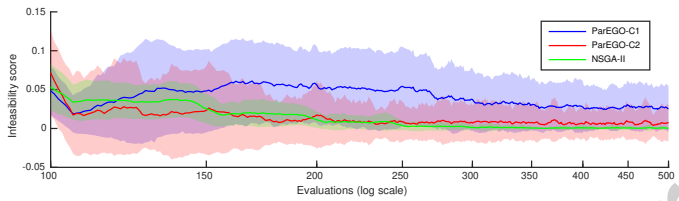
Figure 2: Performance of ParEGO-C1, ParEGO-C2 and NSGA-II on the OSY problem. Subfigures in the left show the hypervolume indicator, infeasibility score, and the number of feasible solutions found. For each performance indicator the mean and variance are represented by the middle line and surrounding shaded region, respectively. Subfigures in the right show the differences between the empirical attainment functions (EAFs) of the non-dominated solutions. The plots in the left highlights the differences in favour of algorithm 1, and the plots in the right highlights the differences in favour of algorithm 2. The colour level encodes the magnitude of the observed differences. The lines in the left, centre and right correspond to the best, median and worst attainment surfaces, respectively. The statistics and EAFs have been determined based on 21 runs. The three subfigures on the bottom show all solutions generated by each optimizer in a single run, represented in the objective space. The solutions are taken from the median run with respect to hypervolume out of 21 runs.



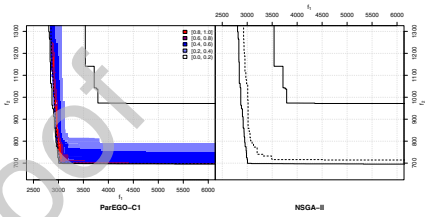
(a) Hypervolume



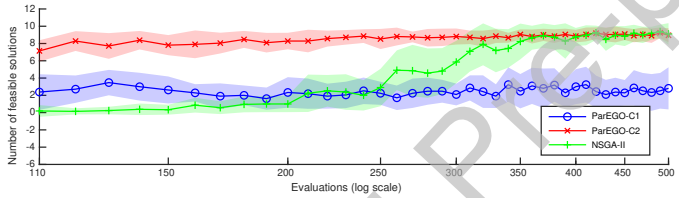
(b) EAFs: ParEGO-C1 vs. ParEGO-C2



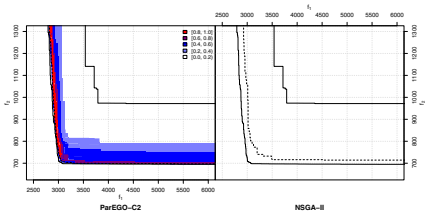
(c) Infeasibility score (Equation 5)



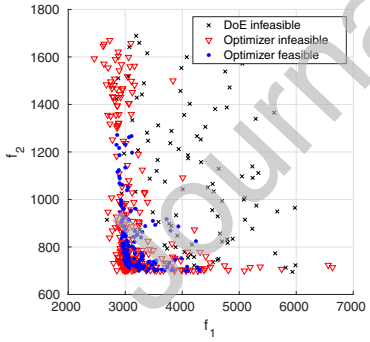
(d) EAFs: ParEGO-C1 vs. NSGA-II



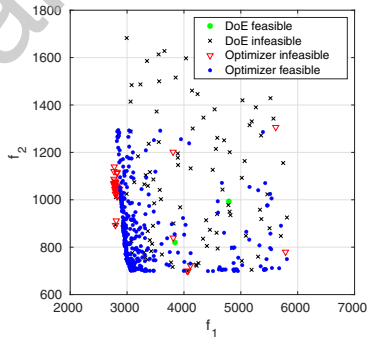
(e) Number of feasible solutions



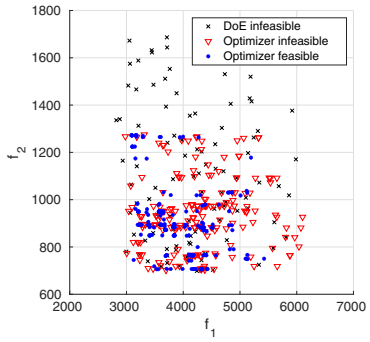
(f) EAFs: ParEGO-C2 vs. NSGA-II



(g) ParEGO-C1

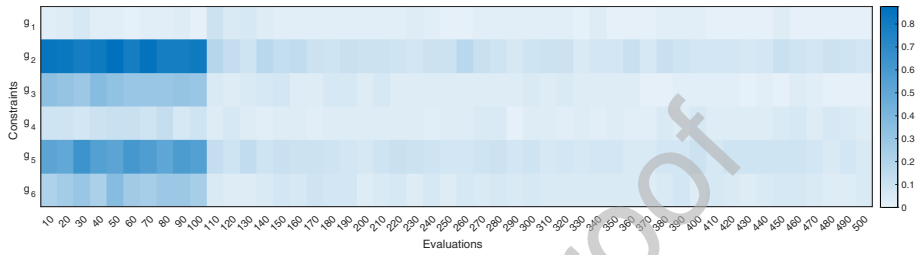


(h) ParEGO-C2

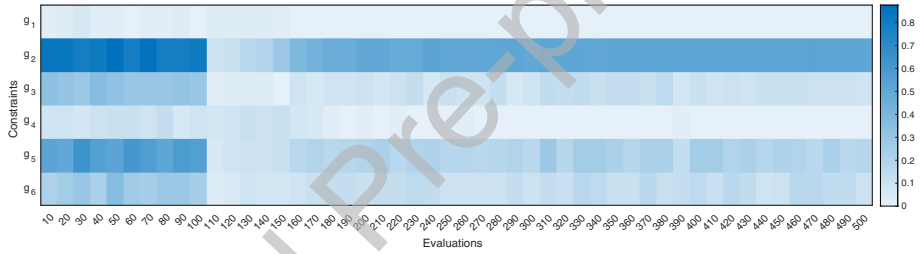


(i) NSGA-II

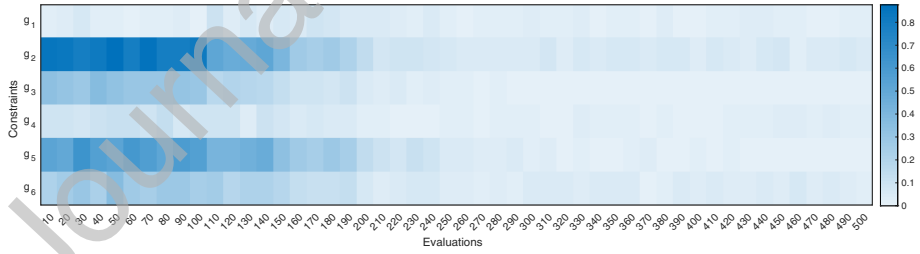
Figure 3: Performance of ParEGO-C1, ParEGO-C2 and NSGA-II on the speed reducer problem. Subfigures in the left show the hypervolume indicator, infeasibility score, and the number of feasible solutions found. For each performance indicator the mean and variance are represented by the middle line and surrounding shaded region, respectively. Subfigures in the right show the differences between the empirical attainment functions (EAFs) of the non-dominated solutions. The plots in the left highlights the differences in favour of algorithm 1, and the plots in the right highlights the differences in favour of algorithm 2. The colour level encodes the magnitude of the observed differences. The lines in the left, centre and right correspond to the best, median and worst attainment surfaces, respectively. The statistics and EAFs have been determined based on 21 runs. The three subfigures on the bottom show all solutions generated by each optimizer in a single run, represented in the objective space. The solutions are taken from the median run with respect to hypervolume out of 21 runs.



(a) ParEGO-C1

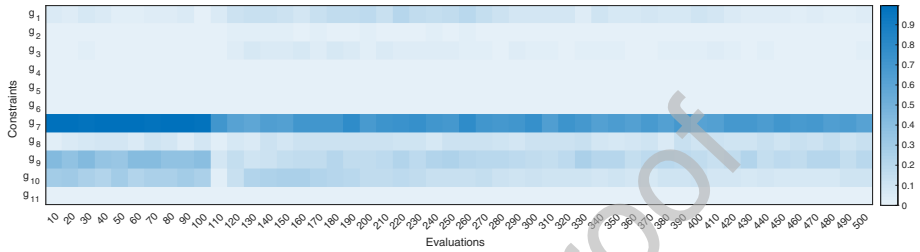


(b) ParEGO-C2

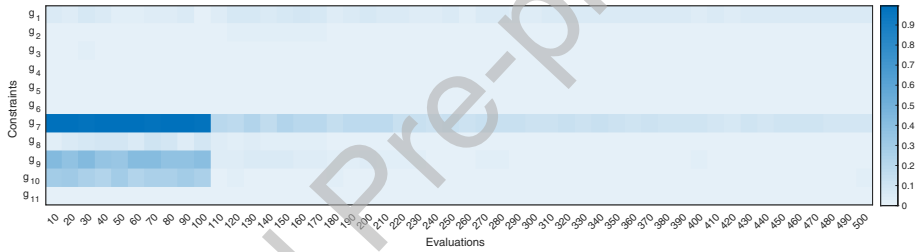


(c) NSGA-II

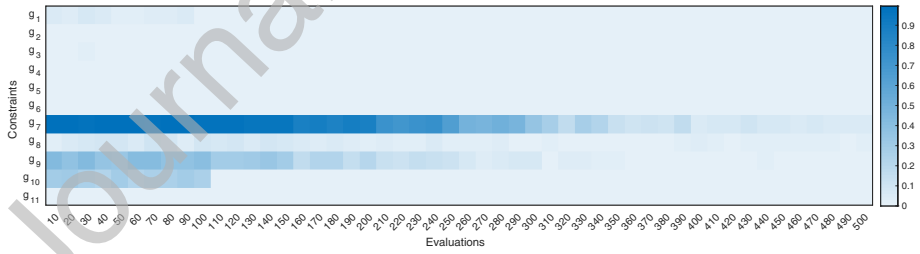
Figure 4: Heatmap showing the constraint violation prevalence for the OSY problem.



(a) ParEGO-C1



(b) ParEGO-C2



(c) NSGA-II

Figure 5: Heatmap showing the constraint violation prevalence for the speed reducer problem.

1. Considering first the hypervolume values obtained by the optimizers. ParEGO-C2 has better hypervolume median performance than NSGA-II for the OSY problem with a percentage improvement of 4.70%, and NSGA-II shows better performance in the speed reducer problem with a percentage improvement of 1.25%. These results give some indication that the solution sets obtained by ParEGO-C2 offers a good approximation to the true PF for both problems.
2. Consider now the number of function evaluations required by each run from NSGA-II to achieve the same median performance as that obtained by ParEGO-C2. NSGA-II requires a median of 2265 function evaluations for the OSY problem, where only 6 out of 21 runs successfully achieve a better performance than ParEGO-C2 within the limit of 10k evaluations. This suggests that some NSGA-II runs may require more than 10000 function evaluations in order to achieve the same median performance that was obtained by ParEGO-C2. For the speed reducer problem, 18 runs out of 21 from NSGA-II successfully achieve a better performance than ParEGO-C2, requiring a median of 1798 function evaluations.

Overall, the above results provide some evidence that ParEGO endowed with a constraint handling approach can be effective at dealing with constrained multi-objective problems with a mix of continuous and discrete variables on a tight budget scenario, and in such circumstances ParEGO has shown to be more effective than NSGA-II. In the following section we will show results for the application of these optimization algorithms to an expensive-to-evaluate ICE design problem.

7. Optimization of an internal combustion engine design problem

7.1. Internal combustion engine design problem

The approach is applied to a real engineering problem consisting of the optimisation of an internal combustion engine design and controller algorithm. The aim of this application is to demonstrate the potential and run time of the approach when applied to a real problem. A model based approach is adopted, whereby a representative engine model of the 1.0 litre 3-cylinder *gasoline direct injection* (GDI) turbocharged engine with a *low-pressure exhaust gas recirculation* (LP EGR) system was used (Figures 6). The engine model was validated by dynamometer test data at various engine speeds and loads for the baseline configuration given in Table 4. The optimization problem is typical of the process that engineering teams would need to undertake during the early development of a new engine version platform. The problem is a constrained multi-objective problem seeking to minimize fuel consumption and NOx emissions over a 2–minute dynamic duty cycle. Seven decision variables are defined. The first four define the hardware choices of cylinder compression ratio (Giles et al. (2018)), turbo machinery and EGR cooler sizing (Dimitriou et al. (2018)). The last three relate to control variables that parameterise the engine control logic (Giles (2018)). Specifically, the optimization problem consists of seven decision variables:

$$\begin{aligned} x_1 &\equiv \text{Compression ratio} \\ x_2 &\equiv \text{EGR cooler size} \\ x_3 &\equiv \text{Turbine flow capacity} \\ x_4 &\equiv \text{Compressor size} \\ x_5 &\equiv \text{Spark timing [CAD]} \\ x_6 &\equiv \text{Inatake and Exhaust Valve timing [CAD]} \\ x_7 &\equiv \text{Manifold pressure target [in Hg]} \end{aligned} \tag{12}$$

The first four decision variables are discrete, meaning that they only take values from the following sets: $x_1 \in \{10, 10.5, 11, 11.5, 12\}$, $x_2 \in \{0.25, 1, 1.75\}$, and $x_3, x_4 \in \{0.75, 1, 1.25\}$. The last three decision variables are continuous, and can take any values from the following intervals: $x_5 \in [-8, 8]$, $x_6 \in [-10, 10]$, and $x_7 \in [0, 20]$. For brevity, let the vector that contains all decision variables be given by $\mathbf{x} = (x_1, x_2, \ldots, x_7)^T$.

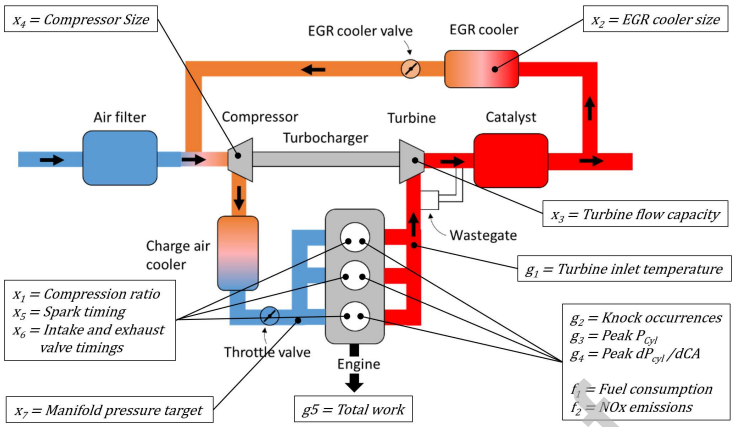


Figure 6: Schematic diagram of a turbocharged GDI engine with low pressure EGR routing.

The physical meaning of the objective and constraint functions is as follows:

$$\begin{aligned}
\text{Minimise} \quad & f_1(\mathbf{x}) \equiv \text{Normalised cumulative fuel [-]} \\
& f_2(\mathbf{x}) \equiv \text{Normalised cumulative NOx [-]} \\
\text{subject to} \quad & g_1(\mathbf{x}) \equiv \text{Turbine inlet temperature [K]} \ \leq m_{limit} \\
& g_2(\mathbf{x}) \equiv \text{Number of knock occurrences} \ \leq g_{b2} \\
& g_3(\mathbf{x}) \equiv \text{Peak cylinder pressure [bar]} \ \leq g_{b3} \\
& g_4(\mathbf{x}) \equiv \text{Peak cylinder pressure rise [bar/rad]} \ \leq g_{b4} \\
& g_5(\mathbf{x}) \equiv \text{Total work [kJ]} \geq g_{b5}
\end{aligned} \tag{13}$$

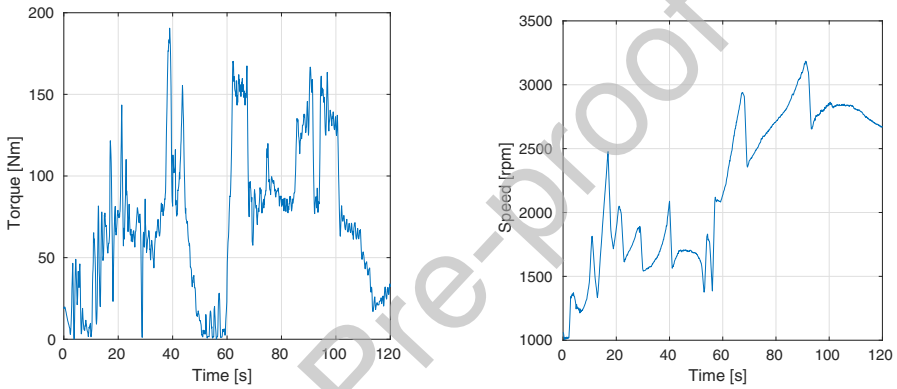
The above optimization problem has two objectives to be minimised, namely fuel consumption and *nitrogen oxides* (NOx) emissions. Both fuel consumption and NOx are measured over a period of time and we take the cumulative values as the final performance of a candidate design $\mathbf{x}$. Constraints g_1 to g_4 are defined as hardware protection limits, defining upper bounds on temperatures and pressures to ensure the engine operates within its thermal and mechanical limits (Parsons et al. (2021); Tang (2016); Tornatore et al. (2019); Zhen et al. (2012)). Constraint g_5 is defined to ensure that a given configuration can meet the overall work output specified by the duty cyle which represents a design requirement. The threshold on g_1 corresponds to the material limit for fixed turbocharger (m_{limit}). The thresholds for the other constraints are defined with respect to a baseline configuration design. The decision variable values for this design are shown in Table 4 and the corresponding outputs are shown in Figure 11[3]. This baseline design will be compared to the best designs obtained by the optimizers in Section 7. However, we would like to highlight that this comparison may not be totally fair since the precise set of criteria used to develop the baseline design is unavailable, and the baseline may have been established using a broader set of objectives and constraints. The model runs in Matlab Simulink version 2017b environment and includes a physics-based model developed in Ricardo WAVE-RT[4] version 2016.2.

The sensitivity of the optimization objectives and constraints are checked by two-level full factorial design for the minimum and maximum values given in Table 4 (Figure 8). The main purpose of giving the sensitivities for fuel and NOx is to demonstrate the engine model is sensitive to changes in the parameters. The minimum and maximum values for control parameters given in Table 4 are far from the optimum values to obtain the maximum brake torque and efficiency.

[3]Case study is commercial sensitive and performance is anonymised through normalisation over the range of the values seen.
[4]WAVE-RT, part of the Ricardo Software product family Fluid Dynamics. Available from: `https://software.ricardo.com/software-updates/ricardo-software-2020-2`

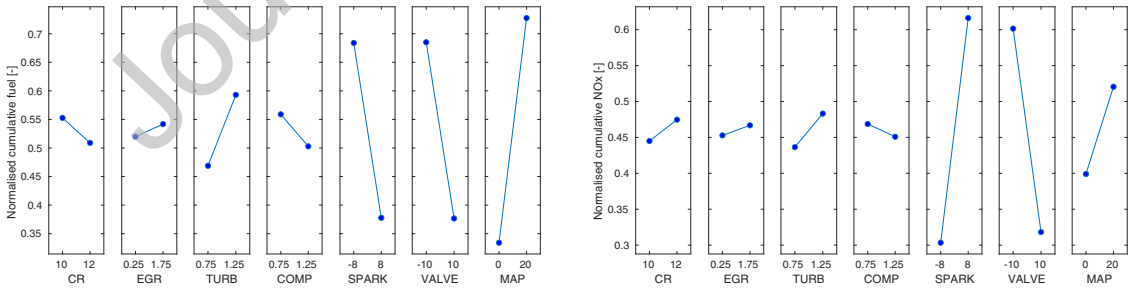
Table 4: Maximum and minimum values for the full factorial design of experiment and sensitivity analysis

	Compression ratio	EGR cooler size	Turbine mass flow multiplier	Compressor mass flow multiplier	Spark offset [CAD]	Valve offset [CAD]	MAP offset [in Hg]
Baseline	10.5	1.0	1.0	1.0	0	0	0
Min	10	0.25	0.75	0.75	-8	-10	0
Max	12	1.75	1.25	1.25	8	10	20



(a) Torque [Nm] (b) Speed [rpm]

Figure 7: Desired engine torque and speed signals used in engine simulation for the 2-minute segment of the WLTP test cycle.



(a) Normalised cumulative fuel (b) Normalised cumulative NOx

Figure 8: Main effects (sensitivity) analysis of architecture and control parameters. CR: Compression ratio, EGR: EGR cooler size, TURB: Turbine mass flow multiplier, COMP: Compressor mass flow multiplier, SPARK: Spark timing offset, VALVE: Valve timing offset, MAP: Manifold absolute pressure offset.

7.2. Optimization results

Figure 9 and Table 5 show a comparison between the two proposed constrained ParEGO algorithms and NSGA-II when applied to the ICE design problem.

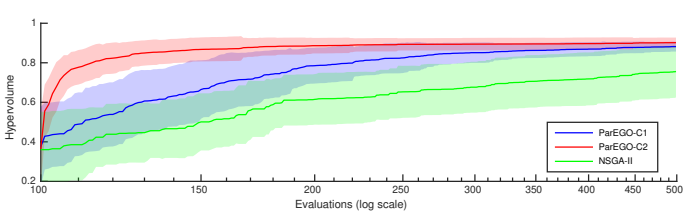
ParEGO-C2 outperforms ParEGO-C1 across the entire optimization run as shown in Figure 9a, with tight confidence intervals and a percentage improvement of 2.70% at the end of the optimization run. Both ParEGO algorithms outperform NSGA-II, where the percentage improvement of ParEGO-C2 over NSGA-II is 11.43%, and for ParEGO-C1 over NSGA-II is 8.73%. All results are statistically significant. Moreover, the EAFs in Figure 9b shows that ParEGO-C2 offers better attainment across several regions of the PF when compared with ParEGO-C1, whilst Figures 9d and 9f show that ParEGO-C1 and ParEGO-C2, respectively, offer better attainment across the entire PF when compared with NSGA-II.

Figures 9c and 9e show the infeasibility score and number of feasible solutions found during the optimization run, respectively. ParEGO-C2 reports the lowest infeasibility score during the first ≈150 evaluations, and the general trend shows that the infeasibility score increases during the optimization run. Although ParEGO-C1 and NSGA-II report higher infeasibility score values than ParEGO-C2 in the first ≈150 evaluations, their values decreases during the optimization run and remain stable after evaluation ≈220. A similar trend is captured by the number of feasible solutions found, as shown in Figure 9e, where ParEGO-C2 finds the highest number during the beginning of the optimization run, but as its number decreases, the number of feasible solutions found by ParEGO-C1 and NSGA-II increases. This is a similar trend observed for the OSY problem, where the infeasibility score of ParEGO-C2 keeps increasing during the optimization run, but decreases for both ParEGO-C1 and NSGA-II as shown in Figure 2c. The same can be said about the number of feasible solutions as shown in Figure 2e, where ParEGO-C2 finds the highest number in the beginning of the optimization run but reports a small reduction as the number of evaluations progresses, while the number of feasible solutions found by both ParEGO-C1 and NSGA-II keeps increasing.

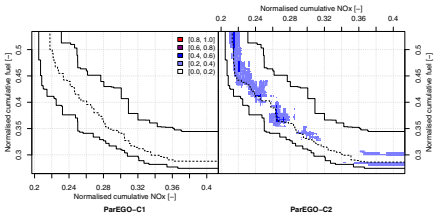
Figures 9g, 9h and 9i shows all solutions evaluated, including the initial set of solutions, taken from the median run with respect to hypervolume. An interesting point about the initial set of solutions is that many of the solutions are infeasible. In fact, in some runs we have noted that there were no feasible solutions in the initial set of solutions, but in all cases the optimizers have managed to find feasible solutions during the optimization run. Moreover, Figure 9g shows that ParEGO-C1 found many feasible solutions in a region close to the PF, and many other infeasible solutions found are in close proximity to this region. Figure 9h shows that ParEGO-C2 only found a few feasible solutions close to the PF region (although with very good convergence), and found many high-performance infeasible solutions further away from the feasible region. This is a similar trend to that captured in Figure 2h for the OSY problem, where many infeasible solutions are generated in a region father away in objective space from the feasible region, while ParEGO-C1 is more likely to generate infeasible solutions closer to the feasible region. Finally, Figure 9i shows that NSGA-II found many feasible solutions but most are sub-optimal.

The probability of violating an individual constraint along the optimization run is shown in Figure 10 for all optimization algorithms. For ParEGO-C1 in Figure 10a, CPR is relatively problematic across the optimization run. CPR is also problematic for ParEGO-C2 as shown in Figure 10b but only after ≈100 evaluations, while Work is always problematic along the entire optimization run for ParEGO-C2. For NSGA-II, in general there are no problematic constraints as shown in Figure 10c, although CPR persists in early stages.

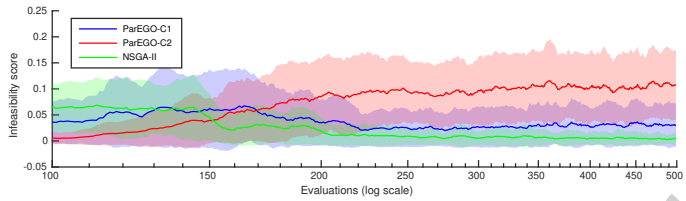
Figure 11 shows in a parallel coordinates plot the values obtained for the objectives, constraints and decision variables. The solutions obtained by the optimization algorithms are taken from the median run with respect to hypervolume, and these are compared with the baseline engine design. Notably, ParEGO offers a better spread of solutions when compared with NSGA-II, and most solutions reported by NSGA-II seem to be clustered around three regions in the objective space. This is due to the narrower set of values obtained by NSGA-II as shown in Figure 11, in particular with respect to the continuous decision variables (i.e. spark timing offset, valve timing offset, and manifold absolute pressure offset). All solutions found by the optimization algorithms have better NOx emissions than the baseline design, and both ParEGO algorithms have found solutions with better fuel consumption than the baseline design. However, the baseline design has better NOx emissions than the solutions found by NSGA-II.



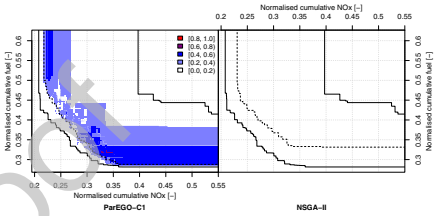
(a) Hypervolume



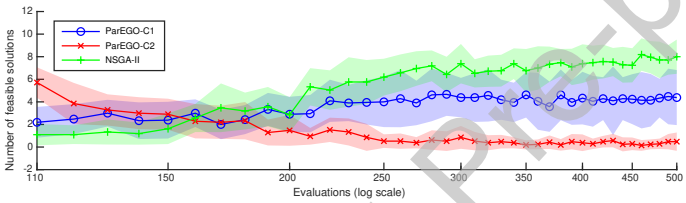
(b) EAFs: ParEGO-C1 vs. ParEGO-C2



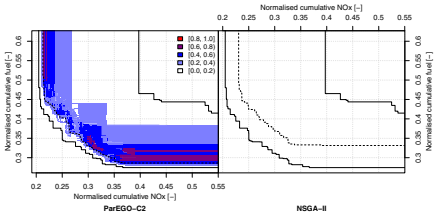
(c) Infeasibility score (Equation 5)



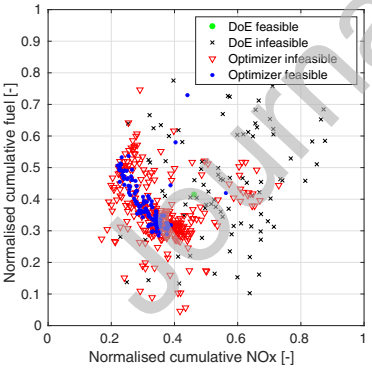
(d) EAFs: ParEGO-C1 vs. NSGA-II



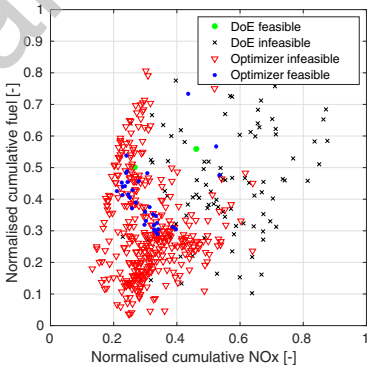
(e) Number of feasible solutions



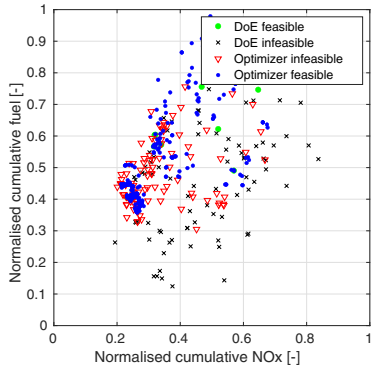
(f) EAFs: ParEGO-C2 vs. NSGA-II



(g) ParEGO-C1

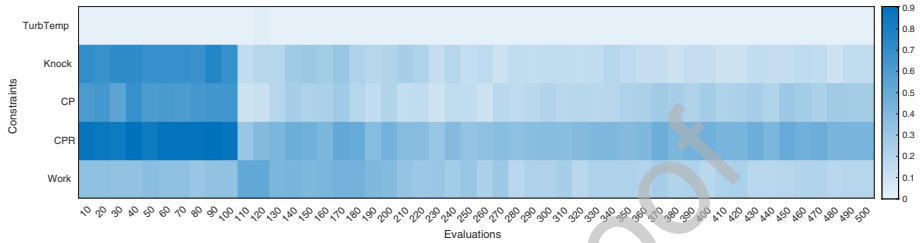


(h) ParEGO-C2

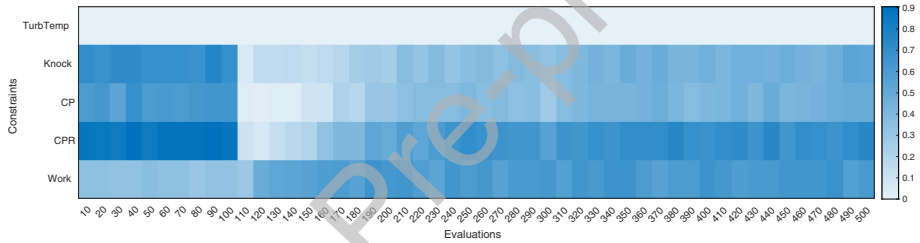


(i) NSGA-II

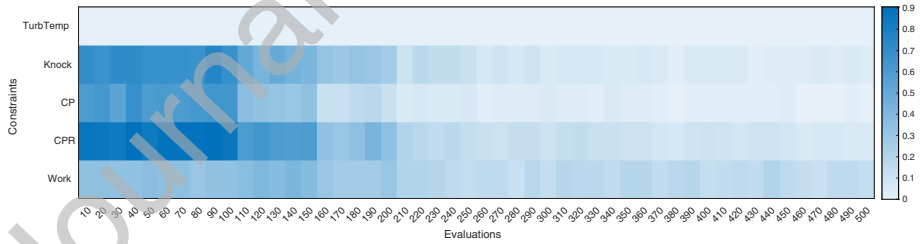
Figure 9: Performance of ParEGO-C1, ParEGO-C2 and NSGA-II on the ICE design problem. Subfigures in the left show the hypervolume indicator, infeasibility score, and the number of feasible solutions found. For each performance indicator the mean and variance are represented by the middle line and surrounding shaded region, respectively. Subfigures in the right show the differences between the empirical attainment functions (EAFs) of the non-dominated solutions. The plots in the left highlights the differences in favour of algorithm 1, and the plots in the right highlights the differences in favour of algorithm 2. The colour level encodes the magnitude of the observed differences. The lines in the left, centre and right correspond to the best, median and worst attainment surfaces, respectively. The statistics and EAFs have been determined based on 21 runs. The three subfigures on the bottom show all solutions generated by each optimizer in a single run, represented in the objective space. The solutions are taken from the median run with respect to hypervolume out of 21 runs.



(a) ParEGO-C1



(b) ParEGO-C2



(c) NSGA-II

Figure 10: Heat map showing the constraint violation prevalence for the ICE design problem. Determined based on 21 runs. The following constraint names have been abbreviated. TurbTemp: Turbine temperature, Knock: Number of knock occurrences, CP: Cylinder pressure, CPR: Cylinder pressure rise, Work: Total work.

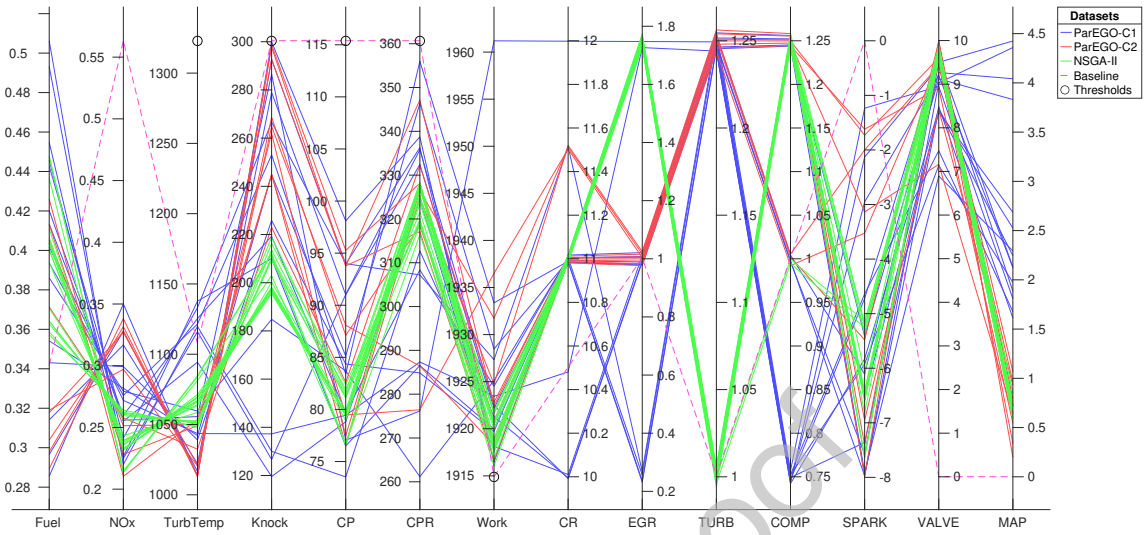


Figure 11: Feasible non-dominated solutions shown in a parallel coordinates plot. The solutions are taken from the median run with respect to hypervolume out of 21 runs. The baseline design is depicted by the dashed line, and the circles indicate the thresholds for the constraints. A small tolerance is applied to the discrete variable values so that they are unlikely to overlap perfectly along the coordinates rulers. Fuel: Normalised cumulative fuel, NOx: Normalised cumulative NOx, TurbTemp: Turbine temperature, Knock: Number of knock occurrences, CP: Cylinder pressure, CPR: Cylinder pressure rise, Work: Total work, CR: Compression ratio, EGR: EGR cooler size, TURB: Turbine mass flow multiplier, COMP: Compressor mass flow multiplier, SPARK: Spark timing offset, VALVE: Valve timing offset, MAP: Manifold absolute pressure offset.

Table 5: Performance of ParEGO-C1, ParEGO-C2, and NSGA-II on the ICE design problem. The performance is measured by the hypervolume indicator (the higher the better). The results on this table also includes the number of solutions that dominate the baseline engine design found during the entire optimization run. The statistics have been determined based on 21 runs. The term Std is an abbreviation for standard deviation, Max for maximum, and Min for minimum.

(a) Hypervolume

Optimizer	Median	Mean (Std)	Max	Min
ParEGO-C1	0.8855	0.8820 (0.0252)	0.9161	0.8061
ParEGO-C2	0.9098	0.9017 (0.0258)	0.9275	0.8284
NSGA-II	0.8114	0.7561 (0.1305)	0.9031	0.4550

(b) Number of solutions that dominate the baseline design

Optimizer	Median	Mean (Std)	Max	Min
ParEGO-C1	61	57 (36)	126	0
ParEGO-C2	14	12 (5)	18	1
NSGA-II	6	24 (27)	70	0

The performance of the optimization algorithms is now compared against the baseline engine design by considering all the optimization runs. The percentage improvement of the best designs found by ParEGO-C1, ParEGO-C2 and NSGA-II over the baseline design are respectively 36.1%, 36.4% and 35.8% for NOx emissions; and 1.82%, 2.03% and 1.56% for fuel consumption. Table 5b shows the number of solutions found during the optimization run that dominate the baseline design. ParEGO-C1 found more solutions that dominate the baseline design, and is the most consistent of the three optimizers. NSGA-II could not find solutions that dominate the baseline design in 8 out of 21 runs, while ParEGO-C2 found at least one solution that dominates the baseline design across all runs.

7.3. Discussion

To analyse the obtained results we have used a combination of different quality indicators. Besides the hypervolume and EAFs, which are considered standard quality indicators in the literature (Audet et al. (2021)), we have looked into understanding more about the constraint handling techniques of the algorithms by monitoring how infeasible new solutions are (via the infeasibility score (Equation 5)), the

number of feasible solutions being generated, and also the constraint violation prevalence across the different constraints. This has revealed that some constraints are more active than others, for instance, in the speed reduce problem from a total of 11 constraints, 8 could be considered inactive (i.e. $g_1 - g_6$, g_8 and g_{11}); in the OSY problem from a total of 6 constraints, two could be considered as inactive (i.e. g_1 and g_4), and; the ICE design problem from a total of 5 constraints only one is inactive (i.e. g_1).

ParEGO-C2 exhibits a lack of consistency in finding feasible solutions for the OSY and ICE design problems when compared with ParEGO-C1 and NSGA-II after ≈100 evaluations, and found many high-performance infeasible solutions further away from the feasible region for these problems. For the speed reducer problem the opposite trend was captured, that is, ParEGO-C2 was more effective at finding feasible solutions when compared with the other two optimizers. The reason for this inconsistency remains unclear but this trend was also captured by the infeasibility score, and constraint violation prevalence across the constraints. This suggests that in problems with very active constraints (such as OSY and ICE design problems), the feasible region is not well captured by the surrogate models of the constraints, and in such situation a penalty function technique can be more effective at generating feasible solutions when compared with constrained EI. Despite this, it was revealed that the feasible solutions found by ParEGO-C2 offer better convergence towards and across the Pareto front when compared with ParEGO-C1 and NSGA-II, even when the latter was allowed to run with a much larger evaluation budget.

The reason why NSGA-II found more feasible solutions than the two ParEGO instances could be that variations operators such as crossover are more likely to produce nearby solutions to the existing ones. On the other hand, ParEGO is more likely to find infeasible solutions as it queries the constraint models in an attempt to learn where the boundaries of the feasible region are located. Eventually, as the surrogate models become more accurate, ParEGO is able to reveal the location of the Pareto optimal solutions.

8. Conclusion

This paper has studied the application of a state-of-the-art BOA, known as ParEGO, to solve an expensive constrained multi-objective optimization problem characterised for having a mix of continuous and discrete variables. For this, a real non-linear and transient *internal combustion engine* (ICE) design problem has been formulated, and the conventional ParEGO algorithm has been extended to handle such type of problems. This includes: the integration of a strategy to handle mixed continuous and discrete variables; a new strategy to select a subset of training points to construct a surrogate model and a criterion that exploits the presence of both feasible and infeasible solutions in the training set; and the incorporation of two constraint handling strategies—one based on penalty functions (ParEGO-C1) and the other based on the constrained EI (ParEGO-C2). The proposed ParEGO variants have been compared to a popular optimization algorithm for the multi-objective study of ICE design problems, known as NSGA-II. For this comparison to take place, we also equipped NSGA-II for dealing with problems with a mix of continuous and discrete variables. This comparative analysis was conducted on a limited budget of 500 function evaluations, and involved two benchmark problems taken from the literature (i.e. speed reducer and OSY), and the proposed ICE design problem.

To analyse the obtained results we have employed a combination of different quality indicators. This includes the popular hypervolume and EAFs to measure the convergence towards and diversity across the Pareto front. In addition, to better understand the performance of the constraint handling techniques we monitor the progress of several indicators during the optimization run, such as the number of feasible solutions found, their degree of infeasibility, and the prevalence of constraint violation across the different constraints. This has revealed that some constraints are more active than others, and such information could be used for instance to simplify the optimization problems since inactive constraints could be omitted without affecting the feasible Pareto optimal front. The key findings from this work are:

1. When selecting a subset of solutions to construct a surrogate model, the obtained results suggest that ensuring a good balance between feasible and infeasible solutions may not always be better than simply relying on the solutions performance, and this depends on how active the constraints in the problem are.

2. Constrained ParEGO convergence towards the PF is better than NSGA-II, and ParEGO-C2 is better than ParEGO-C1, confirmed by both hypervolume and EAF.
3. NSGA-II is more sensitive to randomness (e.g. initial solutions) when compared with ParEGO-C2. This has been quantified by the standard deviation of the hypervolume indicator (see Tables 2 and 5). The higher standard deviation obtained by NSGA-II has been linked to lack of convergence. Running NSGA-II for 10000 function evaluations improved convergence and lead to a reduction in the standard deviation (see Table 2).
4. ParEGO-C2 generated many infeasible solutions in the OSY and ICE design problems when compared with ParEGO-C1 and NSGA-II, and this tends to happens after generating very good quality feasible solutions. However, for the speed reducer problem, ParEGO-C2 generated more feasible solutions than the other two algorithms.
5. A comparison with a baseline engine design revealed that the optimization algorithms are able to find improvements of up-to 36.4% for NOx emissions, and up-to 2.03% for fuel consumption. The results shown in Table 5b have also highlighted that: (i) constrained ParEGO found solutions, in 21 out of 21 runs, that dominate the baseline design while NSGA-II fail to do so in 8 out of 21 runs; and (ii) ParEGO-C1 found more solutions that dominate the baseline design than the other two optimizers.

Based on the above key findings, we conclude that constrained ParEGO methods are effective for solving modern ICE design optimization problems, with expensive evaluations due to the new test procedures for real-world engine conditions. This has also been shown on similar benchmark problems.

Although ParEGO-C2 as shown better performance than ParEGO-C1 across all optimization problems, many of the solutions found by ParEGO-C2 during the later stages of the optimization were infeasible. One potential solution to improve the performance of ParEGO-C2 is to endow the algorithm with a local search operator. This however would require that some amount of the computational budget be devoted to perform local search. There are several factors that could have an influence on the performance of both constraint handling approaches and this takes us to discuss their pros and cons. A major difference is that penalty function methods do not need to build a surrogate of each constraint function, which besides avoiding all the computational burden of having to construct several surrogates, also avoids having to deal with the following possible issues, such as: is the same set of training points used to build the surrogate of the objective function suitable for the constraints? Or knowing that the constraint functions can have different levels of complexity is the same surrogate modelling approach appropriate for all constraints? Although penalty functions are simple to apply, there are drawbacks associated with the discontinuities that could potentially be created between the feasible and infeasible region, which could cause issues to the surrogate modelling approaches.

For future work, the endeavour of the authors is to extend the current work to handle uncertainties, which could arise from multiple sources including fidelity of evaluation functions and manufacturing tolerances. Uncertainty quantification and management is important in identifying designs that offer robust performance across objectives and constraints. Directions for future work also include, explore the application of different paradigms, such as the use of Bayesian inference as in Tsionas (2019), where the initial decision variable ranges are considered to represent a prior belief in the location of the Pareto front and Markov Chain Monte Carlo converges on a posterior belief for the optimal values of the decision variables, using the scalarised objective functions as a likelihood. Further extend the ParEGO algorithms for dealing with categorical variables, since currently it is limited to discrete variables of the type ordinal. In ParEGO all reference direction vectors are equally important, that is, the same reference direction vector is only visited twice once all the others have been processed. Another key endeavour is to test the proposed ParEGO extensions on other real engineering problems, and also on other benchmark problems, in particular with more tunable parameters, such as the number of levels in the discretization.

Acknowledgements

This work was conducted under the Advanced Propulsion Centre (UK) project DYNAMO, with funding from Innovate UK under grant number 113130. The authors would like to thank Dr Byron Mason, Dr Edward

Windward and Dr Sam Le-Corre from Loughborough University and Dr Tomasz Duda from University of Bath for their contribution in development of the engine control model, Robert Norris from Ricardo plc for his contribution to the development of the Ricardo WAVE-RT model, Roshan Mathew from University of Bath for his role in setting up the WAVE-RT Ricardo Software on the Balena High Performance Computing (HPC) Service at the University of Bath, and Ricardo plc for their support, including the provision of licenses for the WAVE-RT software, which has been instrumental for the generation of simulation results.

Contributions

The following uses the abbreviated names of the authors. JAD and UO have collaborated in writing the paper; UO and RB have formulated the real-world internal combustion engine design optimization problem; JAD and SS have collaborated in the implementation of the ParEGO algorithm and in the design and implementation of the approach to handle a mix of continuous and discrete variables; JAD designed and implemented the subset selection strategies and implemented the Kriging model; DCO implemented the probabilistic constraint handling approach; SS implemented the penalty based approach; RJL led the industrial support at Ford Motor Company; RB provided expert knowledge about the real-world problem, and led the University of Bath DYNAMO studies; RCP provided expert optimization advice; contributed to paper development, and led the University of Sheffield DYNAMO studies.

Appendix A. Background

Appendix A.1. Kriging

The term Kriging was first coined by Matheron (1963) in honour of Daniel Krige, a mining engineer from South Africa interested in the application of mathematical statistics to ore validation, and whose work lead to the development of Kriging from a geostatistics perspective. Sacks et al. (1989) introduced the use of Kriging to engineering problems developing the *design and analysis of computer experiments* (DACE) model. This section provides some details about the Kriging modelling approach used in this paper, and our implementation follows some recommendations from Forrester et al. (2008) and Kleijnen (2017) to construct and search the surrogate model.

Given the initial design $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of N points where each $\mathbf{x} \in \mathbb{R}^n$ is a n-vector of continuous decision variables, and a vector of corresponding scalar evaluations $\mathbf{y} = (y_1, \ldots, y_N)^T$ we seek to learn a mapping $y = f(\mathbf{x})$, where f is our expensive-to-evaluate function. This mapping is approximated by a sample path of a *Gaussian stochastic process* (GP) with unknown mean μ and covariance $\sigma^2\boldsymbol{R}$; σ^2 is the variance of the GP; and $\boldsymbol{R}$ is an $N \times N$ matrix parameterised by the n-dimensional vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T$ and having elements:

$$R_{ab} = \mathcal{K}(||\mathbf{x}_a - \mathbf{x}_b||) = \exp\left(-\sum_{i=1}^{n} \theta_i |x_i^{(a)} - x_i^{(b)}|^2\right), \tag{A.1}$$

where $a, b \in \{1, \ldots, N\}$, $x_i^{(a)}$ is the ith decision variable of the point $\mathbf{x}_a$ and the function $\mathcal{K}(||\mathbf{x}_a - \mathbf{x}_b||)$ measures the correlation between the responses at two design points. To learn $\boldsymbol{\theta}$ a common approach is to use *maximum likelihood estimation* (MLE), which provides estimates for the mean and variance, respectively:

$$\hat{\mu} = \frac{\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{y}}{\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{1}} \quad \text{and} \quad \hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^T\boldsymbol{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{N}, \tag{A.2}$$

where $\mathbf{1}$ is a n-dimensional column vector of ones. It is however required to solve the following unconstrained optimization problem:

$$\begin{aligned} &\text{Maximise} && -\frac{N}{2}\ln(\hat{\sigma}^2) - \frac{1}{2}\ln|\boldsymbol{R}|, \\ &\text{subject to} && \theta_i = 10^{\tau_i} \text{ where } -3 \le \tau_i \le 2, \qquad i = 1, \ldots, n, \end{aligned} \tag{A.3}$$

which cannot be solved analytically and thus requires a numerical optimization algorithm. The most computational expensive task in (A.3) is to determine the inverse of $\boldsymbol{R}$. For this, we use Cholesky decomposition followed by forward and back substitution. In case the decomposition fails, which might happen if $\boldsymbol{R}$ is close to singular, we assign a very small value to the objective function in (A.3). Based on our experiments we have observed that this approach always lead to a well conditioned matrix at the end of the optimization process.

To make a new prediction at $\mathbf{x}$ we use the MLE predictor function given by:

$$\hat{f}(\mathbf{x}) = \hat{\mu} + \boldsymbol{r}^T \boldsymbol{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}), \tag{A.4}$$

where $\boldsymbol{r} = (\mathcal{K}(||\mathbf{x}_1 - \mathbf{x}||), \ldots, \mathcal{K}(||\mathbf{x}_N - \mathbf{x}||))^T$ is a vector of correlations between all known points and $\mathbf{x}$. One of the key advantages of Kriging is that each prediction has its own error estimate. This is known as the *mean squared error* (MSE) and is given by the following expression:

$$\hat{\varepsilon}^2(\mathbf{x}) = \hat{\sigma}^2 \left[1 - \boldsymbol{r}^T \boldsymbol{R}^{-1} \boldsymbol{r} + \frac{(1 - \mathbf{1}^T \boldsymbol{R}^{-1} \boldsymbol{r})^2}{\mathbf{1}^T \boldsymbol{R}^{-1} \mathbf{1}}\right]. \tag{A.5}$$

Appendix A.2. Background on ParEGO

ParEGO is a surrogate-based multi-objective optimization algorithm that exhibits a promising performance for scenarios where the number of evaluations is restricted in number. The algorithm itself builds on the EGO algorithm, a single-objective optimizer specifically designed for expensive objective functions. Some details about the EGO algorithm are as follows.

Appendix A.2.1. EGO

The first step of EGO is to generate an initial set of solutions. Different space-filling schemes could be used here, some of the most popular ones are Latin Hypercube Sampling (Mckay et al. (2000)) or simply generate new solutions randomly inside the decision space. Following the evaluation of the initial solutions, a surrogate model is learnt by using the solution/fitness pairs, where fitness corresponds to the value of the objective function. The next step is to use the surrogate model to estimate where "best" to sample the next point. However, this does not imply that the next solution to be evaluated corresponds literally to the location that improves the estimated fitness, since we need to take into account the model's accuracy. In fact, a solution that has a good fitness and low uncertainty, might not be as desirable as a solution with a poor predicted fitness but with a high uncertainty. Hence, we need some criterion that promotes a balance between exploration and exploitation, where exploration is associated with areas in the search space with high uncertainty, whereas exploitation corresponds to areas with better fitness. For this, EGO relies on the EI function (Equation A.7), and also on the prediction and error estimation properties of the Kriging model (refer to Equations A.4 and A.5).

More formally, the uncertainty in the model prediction is the variance of a normal distributed random variable $F(\mathbf{x}) \sim N(\hat{f}(\mathbf{x}), \hat{\varepsilon}^2(\mathbf{x}))$ at point $\mathbf{x}$ with mean $\hat{f}(\mathbf{x})$ and variance $\hat{\varepsilon}^2(\mathbf{x})$. Due to the fact that $F(\mathbf{x})$ could take different values, controlled by the size of $\hat{\varepsilon}^2(\mathbf{x})$, we rely on the EI to balance $\hat{f}$ and $\hat{\varepsilon}$. Let $\mathcal{X}$ denote a set that contains all solutions evaluated so far, and let $f(\mathbf{x}^+)$ denote the current best "known" function value that is located at point $\mathbf{x}^+$, where:

$$\mathbf{x}^+ = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}). \tag{A.6}$$

The improvement at $\mathbf{x}$ is $I(\mathbf{x}) = \max(f(\mathbf{x}^+) - F(\mathbf{x}), 0)$, and the EI is obtained by taking the expected value as given by $E[I(\mathbf{x})] \equiv E[\max(f(\mathbf{x}^+) - F(\mathbf{x}), 0)]$. A closed form expression for this expectation is given by:

$$E[I(\mathbf{x})] = \begin{cases} (f(\mathbf{x}^+) - \hat{f}(\mathbf{x}))\Phi\left(\dfrac{f(\mathbf{x}^+) - \hat{f}(\mathbf{x})}{\hat{\varepsilon}(\mathbf{x})}\right) + \hat{\varepsilon}(\mathbf{x})\phi\left(\dfrac{f(\mathbf{x}^+) - \hat{f}(\mathbf{x})}{\hat{\varepsilon}(\mathbf{x})}\right) & \text{if } \hat{\varepsilon}(\mathbf{x}) > 0 \\ 0 & \text{if } \hat{\varepsilon}(\mathbf{x}) = 0 \end{cases} \tag{A.7}$$

where $\Phi(.)$ and $\phi(.)$ are the cumulative distribution function (CDF) and probability density function (PDF), respectively. Note that, the first term in Equation A.7 before the summation controls the exploitation, and the second term after the summation controls the exploration. The next point to be sampled is found by maximising the EI function, which involves using a numerical optimization approach since it cannot be solved analytically. The EGO algorithm completes one iteration once the new point is evaluated. In the next iteration the surrogate model is updated, and a new point is found again by conducting a search over the EI and finding the point that maximises it. This procedure repeats itself until some termination criterion is satisfied.

The above approach has been extended for dealing with multi-objective problems by ParEGO. This is achieved by relying on scalarisation functions which have been widely used to convert multi-objective optimization problems into a single objective optimization problem.

Appendix A.2.2. Scalarisation

Before we delve into scalarisation, the objectives need to be converted to non-dimensional units. This requires estimating a lower and upper bound with respect to each objective, respectively, as follows:

$$\begin{aligned} z_i^l &= \min_{\mathbf{x}\in\mathcal{X}} f_i(\mathbf{x}),\ i = 1, \ldots, M; \\ z_i^u &= \max_{\mathbf{x}\in\mathcal{X}} f_i(\mathbf{x}),\ i = 1, \ldots, M. \end{aligned} \tag{A.8}$$

Note that the lower and upper bounds have been estimated from solutions in $\mathcal{X}$, meaning that as more solutions are added to $\mathcal{X}$, the estimated bounds are likely to change[5]. The objectives are then normalised as follows:

$$\hat{z}_i = (z_i - z_i^l)/(z_i^u - z_i^l),\ i = 1, \ldots, M. \tag{A.9}$$

The use of scalarisation functions requires the use of reference direction vectors, and each vector targets the PF from a different direction. Let $\mathbf{d}$ be a reference direction vector that exists in the set $\mathbf{D}$ of suitable reference direction vectors (Equation A.10). $\mathbf{D}$ is constructed by using a $\{M, h\}$ *simplex lattice design* that consists of M-dimensional points defined by the following coordinate settings: the proportions assumed by each component d_i where $i \in \{1, \ldots, M\}$ take the $h + 1$ equally spaced values from 0 to 1.

$$\mathbf{D} = \left\{ \mathbf{d} = (d_1, \ldots, d_M)^T \mid \sum_{i=1}^{M} d_i = 1 \text{ and } d_i \in \left\{ 0, \frac{1}{h}, \frac{2}{h}, \ldots, 1 \right\} \text{for all } i = 1, \ldots, M \right\} \tag{A.10}$$

In an iterative approach starting from the first reference direction vector, a scalar fitness value is calculated for each solution by using a scalarizing function. This function maps an objective vector $\mathbf{z}$ corresponding to solution $\mathbf{x}$ into a scalar value with respect to some weight vector $\mathbf{w} = (w_1, \ldots, w_M)^T$. The scalarizing function used is called the weighted Chebyshev Augmented and has the following form:

$$s(\mathbf{x}) = \max_{1\le i\le M} \{w_i \hat{z}_i\} + \rho \sum_{i=1}^{M} w_i \hat{z}_i, \tag{A.11}$$

where ρ is a small positive number which we set to 0.05 as suggested in Knowles (2005). It was mentioned in Giagkiozis et al. (2013) that when using certain scalarizing functions (including the Chebyshev function), the objective vectors of the solutions on the PF might not directly match with the projection of their corresponding reference direction vectors. This is because an evenly distributed set of weighting vectors can only produce well distributed Pareto-optimal solutions when the given scalarisation function is linear in the

[5]In the literature it is common to use the ideal and nadir objective vectors to normalise objectives. These objective vectors are defined with respect to the feasible space $\mathcal{F}$. However, they cannot be used in practice when there are no feasible solutions, which is a strong possibility in our case given the highly constrained nature of the optimization problem. The alternative is to rely on all solutions irrespective of their feasibility status.

weights $\mathbf{w}$ (Giagkiozis et al. (2014)), which is not the case for the Chebyshev function. However, there is an optimal weight vector[6] that accounts for this discrepancy which is defined as:

$$w_i = t_i \Big/ \sum_{i=1}^{M} t_i, \quad \text{where } t_i = (d_i + \epsilon)^{-1}, \quad i = 1, \ldots, M, \tag{A.12}$$

and ϵ is a small number set to 0.01. Each component in Equation A.12 is normalised to ensure that the sum of all components adds up-to one. The transformation in Equation A.12 is only optimal for the weighted sum and weighted Chebyshev Augmented scalarising functions. To use other scalarising functions there is a more general approach suggested in Giagkiozis et al. (2013).

A surrogate model can now be learnt by using the existing solution/fitness pairs. A new solution is then found by maximising the EI function, which is then added to $\mathcal{X}$, and following its evaluation one iteration is completed. In the next iteration the same procedure is applied to a new sub-problem, where all solutions are scalarised with respect to a different reference direction vector. This is similar to the procedure described above for EGO, the main difference is that as more solutions are generated by traversing all reference direction vectors, multiple trade-off solutions are expected to be found.

Appendix A.3. Background on NSGA-II

NSGA-II (Deb et al. (2002)) is a well known Pareto-dominance based multi-objective evolutionary algorithm capable of handling multi-objective problems. NSGA-II is chosen in this study due to its current popularity in the study of multi-objective ICE design problems, as in Corre et al. (2019); DErrico et al. (2011); Lotfan et al. (2016). In this section we provide some details about NSGA-II, including how it handles constraints. The extension to handle problems with a mix of continuous and discrete variables will be described in Section 4.1.

NSGA-II evolves a set of solutions (also known as population[7]) by relying on genetic operators that conduct selection, crossover and mutation. The main loop of the NSGA-II algorithm is as follows. Initially a parent population P is created by using some design of experiments technique (e.g. Latin Hypercube sampling). Following this, an offspring population Q is created by applying binary tournament selection, recombination, and mutation operators to P. Next, an elitist-preserving approach and a parameterless niching operator are applied to the combined population $R = P \cup Q$, and a subset of solutions from R are chosen to replace the population in P by applying selection operators. This completes the first generation, and in subsequent generations the whole process repeats itself by first creating a new Q from the current P and each generation ends once P is updated. This iterative process repeats itself until some termination criterion is satisfied (e.g. the maximum number of generations has been exceeded).

There is a small difference between the above main loop, as described in Deb et al. (2002), and our own implementation. In the first generation the combined population R is treated as the initial population, which is initialised by some design of experiments technique. Following this, P is created by applying selection operators to R as mentioned above, Q is created in the usual way, and the two are combined to generate a new R. This completes the first generation, and the same process repeats again for more generations having R as the input for each generation. Some details about the operators as as follows:

1. The elitist-preserving approach divides R into non-dominated ranks and starting from the best rank to the worst one, only the best ranks are allowed into P until its maximum size is exceeded.
2. The niching operator, known as crowding distance, is applied to the solutions in last rank of P. It then determines how crowded each solution is by measuring their distances in the objective space, and the most crowded solutions are removed from the last rank until the size of P is no longer exceeded.

[6] Based on our simulation results, the optimal $\mathbf{w}$ for the Chebyshev function obtained by solving the optimization problem in Equation 11 by Giagkiozis et al. (2013) can be approximated instead by using the transformation in Equation A.12.

[7] The term population is used in the context of population-based search methods employed by evolutionary optimization algorithms and others, and refers to a container of fixed size with solutions that are updated after each generation. But this excludes BOAs, such as ParEGO, which relies on all evaluated solutions (kept in $\mathcal{X}$) during the course of the optimization run.

3. The binary tournament selects two solutions from P at random, and generates several pairs. A single solution is chosen from each pair as the winner of a tournament selection, and the selected solutions are used to create a new Q. The tournament selection is as follows. The solutions are first compared by looking into their non-dominated rank, and the solution with the best rank is chosen. In case both solutions have the same rank, then the least crowded solution is chosen (as determined by the crowding distance operator).
4. The crossover and mutation are also called variation operators and are applied to Q to make it more distinct from P. There are many crossover and mutation operators in the literature to choose from, and for continuous problems it is common to use simulated binary crossover (Deb & Kumar (1995)) and polynomial mutation (Deb & Goyal (1996)) when using NSGA-II.

NSGA-II handles constraints by relying on the concept of infeasibility score. The constraint violation of $\mathbf{x}$ with respect to the jth constraint is $v_j(\mathbf{x}) = \max(g_j(\mathbf{x}) - c_j, 0)$, and the infeasibility score of $\mathbf{x}$ is:

$$\xi(\mathbf{x}) = \sum_{j=1}^{J} v_j(\mathbf{x}). \tag{A.13}$$

Notably, the higher the infeasibility score, the higher is the degree of constraint violation. The infeasibility score is used by a modified definition of dominance, which states that solution $\mathbf{x}_1$ dominates solution $\mathbf{x}_2$ if any of the following conditions is true:

1. Solution $\mathbf{x}_1$ is feasible and solution $\mathbf{x}_2$ is not.
2. Solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ are both infeasible, but solution $\mathbf{x}_1$ has a smaller infeasibility score.
3. Solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ are both feasible, and solution $\mathbf{x}_1$ dominates $\mathbf{x}_2$.

The above modified definition of dominance is used by the elitist-preserving approach and it will interfere in the composition of the non-dominated ranks. It is expected for solutions that are feasible to be promoted to better ranks when compared with infeasible ones, and solutions which are closer to the feasible region will be promoted to better ranks when compared with more infeasible solutions.

Appendix B. OSY and speed reducer design problem formulations

The OSY problem (Osyczka & Kundu (1995)) is a bi-objective problem with six decision variables and six constraints. The decision variables are all continuous, and four of the constraints are linear. The problem formulation is as follows:

$$\begin{aligned}
\text{Minimise} \quad & f_1(\mathbf{x}) = -[25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2], \\
\text{Minimise} \quad & f_2(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2, \\
\text{subject to} \quad & g_1(\mathbf{x}) \equiv x_1 + x_2 - 2 \geq 0, \\
& g_2(\mathbf{x}) \equiv 6 - x_1 - x_2 \geq 0, \\
& g_3(\mathbf{x}) \equiv 2 - x_1 + x_1 \geq 0, \\
& g_4(\mathbf{x}) \equiv 2 - x_1 + 3x_2 \geq 0, \\
& g_5(\mathbf{x}) \equiv 4 - (x_3-3)^2 - x_4 \geq 0, \\
& g_6(\mathbf{x}) \equiv (x_5-3)^2 + x_6 - 4 \geq 0, \\
& 0 \leq x_1, x_2, x_6 \leq 10, \quad 1 \leq x_3, x_5 \leq 5, \quad 0 \leq x_4 \leq 6.
\end{aligned} \tag{B.1}$$

The Speed reducer design problem (Gunawan et al. (2003)) has two objectives, seven decision variables and eleven constraints. The decision variables are all continuous apart from one which is discrete (i.e. x_3).

The problem formulation is as follows:

$$
\begin{aligned}
\text{Minimise} \quad & f_{\text{weight}} = f_1(\mathbf{x}) = 0.7854 x_1 x_2^2 (10x_3^2/3 + 14.933x_3 - 43.0934) - \\
& \qquad 1.508 x_1 (x_6^2 + x_7^2) + 7.477 (x_6^3 + x_7^3) + 0.7854 (x_4 x_6^2 + x_5 x_7^2), \\
\text{Minimise} \quad & f_{\text{stress}} = f_2(\mathbf{x}) = \frac{\sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 1.69 \times 10^7}}{0.1 x_6^3}, \\
\text{subject to} \quad & g_1(\mathbf{x}) \equiv 1/(x_1 x_2^2 x_3) - 1/27 \leq 0, \\
& g_2(\mathbf{x}) \equiv 1/(x_1 x_2^2 x_3^2) - 1/397.5 \leq 0, \\
& g_3(\mathbf{x}) \equiv x_4^3/(x_2 x_3 x_6^4) - 1/1.93 \leq 0, \\
& g_4(\mathbf{x}) \equiv x_5^3/(x_2 x_3 x_7^4) - 1/1.93 \leq 0, \\
& g_5(\mathbf{x}) \equiv x_2 x_3 - 40 \leq 0, \\
& g_6(\mathbf{x}) \equiv x_1/x_2 - 12 \leq 0, \\
& g_7(\mathbf{x}) \equiv 5 - x_1/x_2 \leq 0, \\
& g_8(\mathbf{x}) \equiv 1.9 - x_4 + 1.5 x_6 \leq 0, \\
& g_9(\mathbf{x}) \equiv 1.9 - x_5 + 1.1 x_7 \leq 0, \\
& g_{10}(\mathbf{x}) \equiv f_2(\mathbf{x}) \leq 1300, \\
& g_{11}(\mathbf{x}) \equiv \frac{\sqrt{\left(\frac{745 x_5}{x_2 x_3}\right)^2 + 1.575 \times 10^8}}{0.1 x_7^3} \leq 1100, \\
& 2.6 \leq x_1 \leq 3.6, \quad 0.7 \leq x_2 \leq 0.8, \quad x_3 \in \{17, \ldots, 28\}, \quad 7.3 \leq x_4, x_5 \leq 8.3, \\
& 2.9 \leq x_6 \leq 3.9, \quad 5.0 \leq x_7 \leq 5.5.
\end{aligned}
\tag{B.2}
$$

References

Alcaraz, J., Landete, M., Monge, J. F., & Sainz-Pardo, J. L. (2020). Multi-objective evolutionary algorithms for a reliability location problem. *European Journal of Operational Research*, *283*, 83–93. doi:https://doi.org/10.1016/j.ejor.2019.10.043.

Alonso, J. M., Alvarruiz, F., Desantes, J. M., Hernandez, L., Hernandez, V., & Molto, G. (2007). Combining neural networks and genetic algorithms to predict and reduce diesel engine emissions. *IEEE Transactions on Evolutionary Computation*, *11*, 46–55. doi:10.1109/TEVC.2006.876364.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European Journal of Operational Research*, *292*, 397–422. doi:https://doi.org/10.1016/j.ejor.2020.11.016.

Avilés, J., Mayo-Maldonado, J., & Micheloud, O. (2020). A multi-objective evolutionary approach for planning and optimal condition restoration of secondary distribution networks. *Applied Soft Computing*, *90*, 106182. doi:10.1016/j.asoc.2020.106182.

Bagheri, S., Konen, W., Allmendinger, R., Branke, J., Deb, K., Fieldsend, J., Quagliarella, D., & Sindhya, K. (2017). Constraint handling in efficient global optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference* GECCO 17 (pp. 673–680). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3071178.3071278.

Bird, J. J., Wanner, E., Ekárt, A., & Faria, D. R. (2020). Optimisation of phonetic aware speech recognition through multi-objective evolutionary algorithms. *Expert Systems with Applications*, *153*, 113402. doi:10.1016/j.eswa.2020.113402.

Chen, Y., Zhou, A., & Das, S. (2021). Utilizing dependence among variables in evolutionary algorithms for mixed-integer programming: A case study on multi-objective constrained portfolio optimization. *Swarm and Evolutionary Computation*, *66*, 100928. doi:10.1016/j.swevo.2021.100928.

Chugh, T., Sindhya, K., Hakanen, J., & Miettinen, K. (2019). A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, *23*, 3137–3166. doi:10.1007/s00500-017-2965-0.

Chugh, T., Sindhya, K., Miettinen, K., Hakanen, J., & Jin, Y. (2016). On Constraint Handling in Surrogate-Assisted Evolutionary Many-Objective Optimization. In J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, & B. Paechter (Eds.), *International Conference on Parallel Problem Solving from Nature (PPSN)* (pp. 214–224). Cham: Springer International Publishing. doi:10.1007/978-3-319-45823-6_20.

Coello, C. C., & Pulido, G. (2005). Multiobjective structural optimization using a microgenetic algorithm. *Structural and Multidisciplinary Optimization*, *30*, 388–403. doi:10.1007/s00158-005-0527-z.

Conover, W. J. (1999). *Practical Nonparametric Statistics*. (3rd ed.). John Wiley & Sons.

Corre, S. D. L., Mason, B., Steffen, T., Winward, E., Yang, Z., Childs, T., Cary, M., & Lygoe, R. (2019). Application of Multi-Objective Optimization Techniques for Improved Emissions and Fuel Economy over Transient Manoeuvres. In *WCX SAE World Congress Experience*. SAE International. doi:10.4271/2019-01-1177.

Couckuyt, I., Deschrijver, D., & Dhaene, T. (2014). Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. *Journal of Global Optimization*, *6*, 575–594. doi:10.1007/s10898-013-0118-2.

De Ath, G., Everson, R. M., Rahat, A. A. M., & Fieldsend, J. E. (2021). Greed is good: Exploration and exploitation trade-offs in bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, *1*, 1–22. doi:10.1145/3425501.

Deb, K., & Goyal, M. (1996). A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, *26*, 30–45.

Deb, K., & Kumar, A. (1995). Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems. *Complex Systems*, *9*, 431–454.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197. doi:10.1109/4235.996017.

Dimitriou, P., Turner, J., Burke, R., & Copeland, C. (2018). The benefits of a mid-route exhaust gas recirculation system for two-stage boosted engines. *International Journal of Engine Research*, *19*, 553–569. doi:10.1177/1468087417723782.

Do, B., Ohsaki, M., & Yamakawa, M. (2021). Bayesian optimization for robust design of steel frames with joint and individual probabilistic constraints. *Engineering Structures*, *245*. doi:10.1016/j.engstruct.2021.112859.

Drake, J. H., Starkey, A., Owusu, G., & Burke, E. K. (2020). Multiobjective evolutionary algorithms for strategic deployment of resources in operational units. *European Journal of Operational Research*, *282*, 729–740. doi:10.1016/j.ejor.2019.02.002.

Duro, J. A., Yan, Y., Giagkiozis, I., Giagkiozis, S., Salomon, S., Oara, D. C., Sriwastava, A. K., Morison, J., Freeman, C. M., Lygoe, R. J., Purshouse, R. C., & Fleming, P. J. (2020). Liger: A cross-platform open-source integrated optimization and decision-making environment. *Applied Soft Computing*, . doi:10.1016/j.asoc.2020.106851. In Press.

DErrico, G., Cerri, T., & Pertusi, G. (2011). Multi-objective optimization of internal combustion engine by means of 1d fluid-dynamic models. *Applied Energy*, *88*, 767–777. doi:10.1016/j.apenergy.2010.09.001.

Emmerich, M. (2005). *Single- and Multi-Objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels*. Ph.D. thesis Technical University Dortmund.

Emmerich, M., Deutz, A., & Klinkenberg, J.-W. (2008). *The computation of the expected improvement in dominated hypervolume of Pareto front approximations*. Technical Report Leiden Institute for Advanced Computer Science, Niels Bohrweg 1, Leiden University, The Netherlands. LIACS-TR 9-2008.

Emmerich, M. T. M., Giannakoglou, K. C., & Naujoks, B. (2006). Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, *10*, 421–439. doi:10.1109/TEVC.2005.859463.

Farmani, R., & Wright, J. A. (2003). Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, *7*, 445–455. doi:10.1109/TEVC.2003.817236.

Feliot, P., Bect, J., & Vazquez, E. (2017). A bayesian approach to constrained single- and multi-objective optimization. *Journal of Global Optimization*, *67*, 97–133. doi:10.1007/s10898-016-0427-3.

Fonseca, C. M., Paquete, L., & López-Ibáñez, M. (2006). An improved dimension–sweep algorithm for the hypervolume indicator. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'06)* (pp. 1157–1163). IEEE Press. doi:10.1109/CEC.2006.1688440.

Forrester, A. I., & Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, *45*, 50–79. doi:10.1016/j.paerosci.2008.11.001.

Forrester, A. I. J., Sóbester, D. A., & Keane, A. J. (2008). *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, Ltd. doi:10.1002/9780470770801.

Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive entropy search for multi-objective bayesian optimization with constraints. *Neurocomputing*, *361*, 50–68. doi:10.1016/j.neucom.2019.06.025.

Giagkiozis, I., Purshouse, R., & Fleming, P. (2014). Generalized decomposition and cross entropy methods for many-objective optimization. *Information Sciences*, *282*, 363–387. doi:10.1016/j.ins.2014.05.045.

Giagkiozis, I., Purshouse, R. C., & Fleming, P. J. (2013). Generalized Decomposition. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, & J. Shaw (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 428–442). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-37140-0_33.

Giles, K., , Lewis, A., Akehurst, S., Prakash, A., Redmann, J.-H., Cracknell, R., Aradi, A. A., & Turner, N. (2018). Octane response of a highly boosted direct injection spark ignition engine at different compression ratios. In *WCX World Congress Experience*. SAE International. doi:10.4271/2018-01-0269.

Giles, K. (2018). *Predicting Abnormal Combustion Phenomena In Highly Booted Spark Ignition Engines*. Ph.D. thesis Department of Mechanical Engineering, University of Bath, UK.

Gong, W., Cai, Z., & Zhu, L. (2009). An efficient multiobjective differential evolution algorithm for engineering design. *Structural and Multidisciplinary Optimization*, *38*, 137–157. doi:10.1007/s00158-008-0269-9.

Gunawan, S., Azarm, S., Wu, J., & Boyars, A. (2003). Quality-assisted multi-objective multidisciplinary genetic algorithms. *AIAA Journal*, *41*, 1752–1762. doi:10.2514/2.7293.

Hussein, R., & Deb, K. (2016). A Generative Kriging Surrogate Model for Constrained and Unconstrained Multi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* GECCO 16 (pp. 573–580). New York, NY, USA: Association for Computing Machinery. doi:10.1145/2908812.2908866.

IDE (2021). *The Future of the Automotive Industry through Digitalisation*. Technical Report Institute of Digital Engineering. https://roadmap.ide.uk/.

Jeong, S., & Obayashi, S. (2005). Efficient global optimization (EGO) for multi-objective problem and data mining. In *2005 IEEE Congress on Evolutionary Computation* (pp. 2138–2145). volume 3. doi:10.1109/CEC.2005.1554959.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, *13*, 455–492. doi:10.1023/A:1008306431147.

Karra, P., & Kong, S.-C. (2010). Application of Particle Swarm Optimization for Diesel Engine Performance Optimization. In *SAE 2010 World Congress & Exhibition*. SAE International. doi:10.4271/2010-01-1258.

Keane, A. J. (2006). Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, *44*, 879–891.

Keyes, D. E., McInnes, L. C., Woodward, C., Gropp, W., Myra, E., Pernice, M., Bell, J., Brown, J., Clo, A., Connors, J., Constantinescu, E., Estep, D., Evans, K., Farhat, C., Hakim, A., Hammond, G., Hansen, G., Hill, J., Isaac, T., Jiao, X., Jordan, K., Kaushik, D., Kaxiras, E., Koniges, A., Lee, K., Lott, A., Lu, Q., Magerlein, J., Maxwell, R., McCourt, M., Mehl, M., Pawlowski, R., Randles, A. P., Reynolds, D., Rivière, B., Rüde, U., Scheibe, T., Shadid, J., Sheehan, B., Shephard, M., Siegel, A., Smith, B., Tang, X., Wilson, C., & Wohlmuth, B. (2013). Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, *27*, 4–83. doi:10.1177/1094342012468181.

Kleijnen, J. P. (2017). Regression and kriging metamodels with their experimental designs in simulation: A review. *European Journal of Operational Research*, *256*, 1–16. doi:10.1016/j.ejor.2016.06.041.

Knowles, J. (2005). ParEGO: A Hybrid Algorithm With On-Line Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, *10*, 50–66. doi:10.1109/TEVC.2005.851274.

Knowles, J., & Hughes, E. J. (2005). Multiobjective optimization on a budget of 250 evaluations. In C. A. C. Coello, A. H. Aguirre, & E. Zitzler (Eds.), *Evolutionary Multi-Criterion Optimization. EMO 2005*. Springer, Berlin, Heidelberg volume 3410 of *Lecture Notes in Computer Science*. doi:10.1007/978-3-540-31880-4_13.

Koziel, S., & Pietrenko-Dabrowska, A. (2022). Constrained multi-objective optimization of compact microwave circuits by design triangulation and pareto front interpolation. *European Journal of Operational Research*, *299*, 302–312. doi:10.1016/j.ejor.2021.08.021.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, *52*, 119–139. URL: https://journals.co.za/content/saimm/52/6/AJA0038223X_479.

Kuk, J. N., Gonalves, R. A., Pavelski, L. M., Guse Scós Venske, S. M., de Almeida, C. P., & Ramirez Pozo, A. T. (2021). An empirical analysis of constraint handling on evolutionary multi-objective algorithms for the environmental/economic load dispatch problem. *Expert Systems with Applications*, *165*, 113774. doi:10.1016/j.eswa.2020.113774.

Li, G., Li, M., Azarm, S., Hashimi, S. A., Ameri, T. A., & Qasas, N. A. (2009). Improving multi-objective genetic algorithms with adaptive design of experiments and online metamodeling. *Structural and Multidisciplinary Optimization*, *37*, 447–461. doi:10.1007/s00158-008-0251-6.

Li, M., Li, G., & Azarm, S. (2008). A Kriging Metamodel Assisted Multi-Objective Genetic Algorithm for Design Optimization. *Journal of Mechanical Design*, *130*. doi:10.1115/1.2829879.

Li, R., Emmerich, M. T., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., & Reiber, J. (2013). Mixed Integer Evolution Strategies for Parameter Optimization. *Evolutionary Computation*, *21*, 29–64. doi:10.1162/EVCO_a_00059.

López-Ibáñez, M., Paquete, L., & Stützle, T. (2010). Exploratory analysis of stochastic local search algorithms in biobjective optimization. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, & M. Preuss (Eds.), *Experimental Methods for the Analysis of Optimization Algorithms* (pp. 209–222). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-02538-9_9.

Lotfan, S., Ghiasi, R. A., Fallah, M., & Sadeghi, M. (2016). ANN-based modeling and reducing dual-fuel engines challenging emissions by multi-objective evolutionary algorithm NSGA-II. *Applied Energy*, *175*, 91–99. doi:10.1016/j.apenergy.2016.04.099.

Martínez-Frutos, J., & Herrero-Pérez, D. (2016). Kriging-based infill sampling criterion for constraint handling in multi-objective optimization. *Journal of Global Optimization*, *64*. doi:10.1007/s10898-015-0370-8.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, *58*, 1246–1266. doi:10.2113/gsecongeo.58.8.1246.

McGinley, B., Maher, J., O'Riordan, C., & Morgan, F. (2011). Maintaining Healthy Population Diversity Using Adaptive Crossover, Mutation, and Selection. *IEEE Transactions on Evolutionary Computation*, *15*, 692–714. doi:10.1109/TEVC.2010.2046173.

Mckay, M. D., Beckman, R. J., & Conover, W. J. (2000). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics*, *42*, 55–61. doi:10.1080/00401706.2000.10485979.

Miettinen, K., Mäkelä, M. M., & Toivanen, J. (2003). Numerical Comparison of Some Penalty-Based Constraint Handling Techniques in Genetic Algorithms. *Journal of Global Optimization*, *27*, 427–446. doi:10.1023/A:1026065325419.

Millo, F., Arya, P., & Mallamo, F. (2018). Optimization of automotive diesel engine calibration using genetic algorithm techniques. *Energy*, *158*, 807–819. doi:10.1016/j.energy.2018.06.044.

Mlakar, M., Petelin, D., Tušar, T., & Filipič, B. (2015). GP-DEMO: Differential Evolution for Multiobjective Optimization based on Gaussian Process models. *European Journal of Operational Research*, *243*, 347–361. doi:10.1016/j.ejor.2014.04.011.

Nowacki, H. (1980). Modelling of design decisions for CAD. In J. Encarnacao (Ed.), *Computer Aided Design Modelling, Systems Engineering, CAD-Systems: CREST Advanced Course Darmstadt, 8.–19. September 1980* (pp. 177–223). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/BFb0040161.

Osyczka, A., & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, *10*, 94–99. doi:10.1007/bf01743536.

Parsons, D., Orchard, S., Evans, N., Ozturk, U., Burke, R., & Brace, C. (2021). A comparative study into the effects of pre and post catalyst exhaust gas recirculation on the onset of knock. *International Journal of Engine Research*, *22*, 2819–2829. doi:10.1177/1468087420962294.

Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.-G., & Guerin, Y. (2020). Overview and Comparison of Gaussian Process-Based Surrogate Models for Mixed Continuous and Discrete Variables: Application on Aerospace Design Problems. In T. Bartz-Beielstein, B. Filipič, P. Korošec, & E.-G. Talbi (Eds.), *High-Performance Simulation-Based Optimization* (pp. 189–224). Springer International Publishing. doi:10.1007/978-3-030-18764-4_9.

Ramos, A., noz, J. M., Andrés, F., & Armas, O. (2018). Nox emissions from diesel light duty vehicle tested under nedc and real-word driving conditions. *Transportation Research Part D: Transport and Environment*, *63*, 37–48. doi:10.1016/j.trd.

2018.04.018.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rudolph, G. (2005). An evolutionary algorithm for integer programming. In *International Conference on Parallel Problem Solving from Nature (PPSN)* (pp. 139–148). volume 866. doi:10.1007/3-540-58484-6_258.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and Analysis of Computer Experiments. *Statistical Science*, *4*, 409–423. URL: www.jstor.org/stable/2245858.

Schonlau, M., Welch, W. J., & Jones, D. R. (1998). Global versus local search in constrained optimization of computer models. In N. Flournoy, W. F. Rosenberger, & W. K. Wong (Eds.), *New developments and applications in experimental design* chapter New developments and applications in experimental design. (pp. 11–25). Hayward, CA: Institute of Mathematical Statistics volume 34 of *Lecture Notes–Monograph Series*. doi:10.1214/lnms/1215456182.

Singh, P., Couckuyt, I., Ferranti, F., & Dhaene, T. (2014). A Constrained Multi-Objective Surrogate-Based Optimization Algorithm. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 3080–3087). doi:10.1109/CEC.2014.6900581.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*. doi:10.5555/645512.657265.

Tadros, M., Ventura, M., & Soares, C. G. (2019). Optimization procedure to minimize fuel consumption of a four-stroke marine turbocharged diesel engine. *Energy*, *168*, 897–908. doi:10.1016/j.energy.2018.11.146.

Tang, H. (2016). *Application of Variable Geometry Turbine on Gasoline Engines and the Optimisation of Transient Behaviours*. Ph.D. thesis Department of Mechanical Engineering, University of Bath.

Togun, N., & Baysec, S. (2010). Genetic programming approach to predict torque and brake specific fuel consumption of a gasoline engine. *Applied Energy*, *87*, 3401–3408. doi:10.1016/j.apenergy.2010.04.027.

Tornatore, C., Bozza, F., Bellis], V. D., Teodosio, L., Valentino, G., & Marchitto, L. (2019). Experimental and numerical study on the influence of cooled egr on knock tendency, performance and emissions of a downsized spark-ignition engine. *Energy*, *172*, 968–976. doi:10.1016/j.energy.2019.02.031.

Tsionas, M. G. (2019). Multi-objective optimization using statistical models. *European Journal of Operational Research*, *276*, 364–378. doi:10.1016/j.ejor.2018.12.042.

Wang, G. G., & Shan, S. (2007). Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, *129*, 370–380. doi:10.1115/1.2429697.

Yu, X., & Gen, M. (2010). *Introduction to Evolutionary Algorithms*. Decision Engineering. Springer London. doi:10.1007/978-1-84996-129-5.

Zhen, X., Wang, Y., Xu, S., Zhu, Y., Tao, C., Xu, T., & Song, M. (2012). The engine knock analysis—An overview. *Applied Energy*, *92*, 628–636. doi:10.1016/j.apenergy.2011.11.079.