A novel algorithm for fast representation of a Pareto front with adaptive resolution: application to multi-objective optimization of a chemical reactor

I. Hashem[a], D. Telen[a], P. Nimmegeers[a], F. Logist[a], J. Van Impe[a]

[a]*KU Leuven, Chemical Engineering Department, BioTeC & OPTEC, Gebroeders De Smetstraat 1, 9000 Ghent, Belgium*

Abstract

Solving a multi-objective optimization problem yields an infinite set of points in which no objective can be improved without worsening at least another objective. This set is called the Pareto front. A Pareto front with adaptive resolution is a representation where the number of points at any segment of the Pareto front is directly proportional to the curvature of this segment. Such representations are attractive since steep segments, i.e., knees, are more significant to the decision maker as they have high trade-off level compared to the more flat segments of the solution curve. A simple way to obtain such representation is the a posteriori analysis of a dense Pareto front by a smart filter to keep only the points with significant trade-offs among them. However, this method suffers from the production of a large overhead of insignificant points as well as the absence of a clear criterion for determining the required density of the initial dense representation of the Pareto front. This paper's contribution is a novel algorithm for obtaining a Pareto front with adaptive resolution. The algorithm overcomes the pitfalls of the smart filter strategy by obtaining the Pareto points recursively while calculating the trade-off level between the obtained points before moving to a deeper recursive call. By using this approach, once a segment of trade-offs insignificant to the decision maker's needs is identified, the algorithm stops exploring it further. The improved speed of the proposed algorithm along with its intuitively simple solution process make it a more attractive route to solve multi-objective optimization problems in a way that better suits the decision maker's needs.

Keywords: Multi-objective optimization, Pareto front representation, Divide and conquer strategy, Dynamic optimization

1. Introduction

Biochemical processes are usually dynamic systems that are described by a set of differential equations. The optimization of such processes is carried out using control variables to achieve either one objective, yielding a Single Objective Optimization Problem (SOOP) or multiple objectives, yielding a Multi-Objective Optimization Problem (MOOP). Finding the optimal control trajectories as a function of time is a situation that is frequently encountered in chemical engineering applications, e.g., finding the time optimal feed rate for a fed-batch reactor. The problem is typically solved by discretizing the continous control variable into a large number of discrete variables over the time/space interval making it a relatively computationally expensive problem to solve. This makes a multi-objective setting particularly challenging as the control problem has to be repeatedly solved during the solution process. For MOOPs, a set of mathematically equivalent optimal points exists, called the

Postprint version of paper published in Computers and Chemical Engineering, vol. 106, p. 544-558.
The content is identical to the published paper, but without the final typesetting by the publisher.

Pareto front. There are two classes of techniques to obtain an approximation of the Pareto set: vectorization and scalarization methods. Vectorization methods ([13]) are stochastic techniques that tackle the multi-objective optimization problem directly. However, their time consuming nature and the difficulty of incorporating state constraints make them less attractive to be applied in optimal control problems ([19]). On the other hand, scalarization methods ([24]) have been frequently implemented to solve Multi-Objective Optimal Control Problems (MOOCPs). They work by parameterizing the original MOOP into a series of SOOPs. Solving each SOOP yields a point on the Pareto front such that a Pareto front representation can be obtained for the Decision Maker (DM) to examine. It is natural to assume that not all segments of the Pareto front are equally important to a potential DM ([22], [2]). Examples of techniques for the a posteriori analysis of obtained solutions are the order of efficiency filter ([2]), which ranks solutions according to how balanced their overall performance is, and the smart filter ([22]). The motivation for using a smart filter is to emphasize the segments more attractive to the DM in the final representation at the expense of the less significant segments of the curve. The steeper segments, the "knees" of the representation, have higher trade-off level than the more flat "plateau" segments. For a prespecified trade-off level, the filter removes the solutions deemed insignificant to the DM, keeping only solutions which have significant trade-offs between each other. However, the main disadvantage of this approach is the need to produce a dense representation with excess of insignificant solutions for the filter to act on. Considering the large computational cost for solving an instance of a MOOCP, this hinders applying the smart filter strategy to this class of computationally intensive problems. In this paper, an alternative approach is introduced to obtain a Pareto front with adaptive resolution. The novel algorithm utilizes a recursive paradigm in exploring the Pareto front. This way, once a segment of insignificant trade-off level to the decision maker is identified, the algorithm stops generating more solutions within this segment. The paper is structured as follows: in Section 2, the mathematical formulations for solving a MOOCP are introduced. The algorithm's concept of operation is developed in Section 3. Several numerical problems as well as a dynamic benchmark example are presented in Section 4 while the obtained simulation results are discussed in Section 5. Finally, Section 6 summarizes the paper's conclusions.

2. Mathematical formulations

This section is structured as follows: first, the general formulation of multi-objective optimal control problems is discussed. Then, an overview of multi-objective solution algorithms is presented as well as the formulation of a smart filter for the a posteriori analysis of the Pareto front. Finally, Pomodoro, an in-house library used in this paper for solving dynamic optimization problems is introduced.

2.1. Multi-objective optimization formulation

A multi-objective optimal control problem (MOOCP) can be formulated as a minimization problem as follows ([18]):

$$\min_{u(\epsilon),x(\epsilon),p,\epsilon_f} \{J_1, J_2, ..., J_m\} \tag{1}$$

subject to:

$$\frac{dx}{d\epsilon} = F(x(\epsilon), u(\epsilon), p, \epsilon_\mathrm{f}) \quad \epsilon \in [0, \epsilon_\mathrm{f}] \tag{2}$$

$$0 = b_\mathrm{i}(x(0), p) \tag{3}$$

$$0 = b_\mathrm{t}(x(\epsilon_\mathrm{f}), p) \tag{4}$$

$$0 \geq c_\mathrm{p}(x(\epsilon), u(\epsilon), p, \epsilon) \tag{5}$$

$$0 \geq c_\mathrm{t}(x(\epsilon_\mathrm{f}), p, \epsilon_\mathrm{f}) \tag{6}$$

where m is the number of objectives, ϵ is the independent variable, usually time and typically ranging from 0 to ϵ_f, x are the state variables, u represents the control variables and p the time-invariant parameters of the process. The (nonlinear) model equations are denoted by F. The vectors b_i and b_t represent the initial and terminal conditions, respectively. The vectors c_p and c_t denote the path and terminal inequality constraints. In this work, an individual objective function J_i is generally formulated as follows:

$$J_\mathrm{i} = M_\mathrm{i}(x(\epsilon_\mathrm{f}), p, \epsilon_\mathrm{f}) + \int_0^{\epsilon_\mathrm{f}} L_\mathrm{i}(x(\epsilon), u(\epsilon), p, \epsilon) d\epsilon \tag{7}$$

with $M_\mathrm{i}(x(\epsilon_\mathrm{f}), p, \epsilon_\mathrm{f})$ the Mayer term, which represents the terminal cost, e.g., the final conversion at the end of the process and $\int_0^{\epsilon_\mathrm{f}} L_\mathrm{i}(x(\epsilon), u(\epsilon), p, \epsilon) d\epsilon$ the Lagrange term, representing the integral cost over the interval $[0, \epsilon_\mathrm{f}]$, e.g., total fuel consumption during the process.

Finally, for conciseness, a vector of the optimization variables is defined as $y = [x(\cdot)^\top, u(\cdot)^\top, p^\top, \epsilon_\mathrm{f}]^\top$. The individual objective functions are grouped in a vector as $J(y) = [J_1(y), J_2(y), ..., J_\mathrm{m}(y)]^\top$ and the set of feasible solutions S is defined as all vectors y that satisfy the imposed constraints (2)-(6) ([18]).

In multi-objective optimization no single optimal solution exists so the notion of Pareto optimality is adopted. As formulated in, [24], a vector y^* is said to be Pareto optimal if there exists no other $y \in S$ such that $J_\mathrm{i}(y) \leq J_\mathrm{i}(y^*)$ for $\mathrm{i} = 1, 2, ..., \mathrm{m}$ and $J_\mathrm{i}(y) < J_\mathrm{i}(y^*)$ for at least one J_i. A Pareto point is said to be not dominated by any other point in the objective space. This means that y^* is a Pareto optimal point iff there exists no other feasible point that would improve a certain objective without causing a simultaneous increase in another objective. Unless all the objectives are not conflicting, an infinite set of solutions will exist. The complete set of Pareto solutions is called the Pareto front, ([24, 18]).

2.2. Multi-objective optimization solution algorithms

According to a review by [21], two major classes of methods exist to obtain a Pareto front: vectorization methods and scalarization methods. Vectorization methods ([13]) work by solving the multi-objective optimization problem directly using stochastic algorithms. The drawbacks of these methods, as explained in [18] are their inability to handle complex constraints, being time consuming and being limited to low dimensional search spaces. On the other hand, scalarization methods ([24]) are deterministic and can handle a large number of decision variables and constraints. However, they are prone to converging to local optima. In this class of methods, the multi-objective optimization problem is converted to a series of parametrized single objective optimization problems. This set is typically generated by varying a parameters/weights vector. This way, solving each sub-problem gives a point on the Pareto front. Since, dynamic optimization problems usually involve a high number of constraints, scalarization methods are more suited to solve them, several successful applications can be found in [10, 28, 25]. Three of the most widely

used scalarization methods are discussed in this section: *weighted sum* method, *normal boundary intersection* and *(enhanced) normalized normal constraint*.

2.2.1. Weighted sum method (WS)

The (convex) weighted sum method is one of the most widely applied scalarization techniques in practice, mainly due to its simplicity. It is based on combining the multiple objectives into a single convex function composed of their weighted sums as follows, [18]:

$$\min_{y} \sum_{\mathrm{i}=1}^{\mathrm{m}} w_{\mathrm{i}} J_{\mathrm{i}}(y) = w^{\top} J(y) \tag{8}$$

where the weights w_{i} can be grouped in w. Furthermore, $w_{\mathrm{i}} \geq 0$ with $\mathrm{i} = 1, 2, ..., \mathrm{m}$ and $\sum_{\mathrm{i}=1}^{\mathrm{m}} w_{\mathrm{i}} = 1$. The solution of this minimization problem is obtained at y^*. Since the obtained point is a Pareto optimal solution, it lies on the Pareto front of the feasible objectives space. The procedure of the weighted sum method is solving the minimization problem repeatedly using different combinations of w to obtain multiple points on the Pareto front. However, despite its simplicity, the weighted sum method suffers from two major flaws, as explained in [8]:

- An even spread of weights does not produce an even spread of points on the Pareto front.
- It is impossible to detect the non-convex parts of the Pareto front. The algorithm will not produce a complete representation of Pareto fronts that contain non-convex regions, since it is geometrically impossible to find a weight combination that can produce a point in these regions ([8]).

So, despite its simplicity, there is no way to circumvent the drawbacks of the weighted sum method. Therefore, the following alternative techniques have been proposed that do not suffer from these drawbacks.

2.2.2. Normal boundary intersection (NBI)

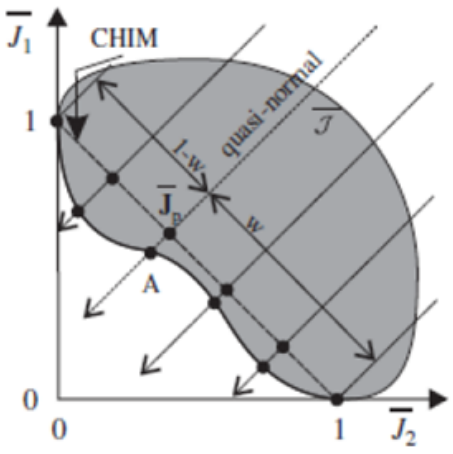


Figure 1: A geometrical illustration of the normal boundary intersection method, after [19].

This method was first proposed by [9] to overcome the drawbacks associated with the weighted sum method. It is based on reformulating the multi-objective optimization problem into a series of parametric optimization problems where an

even distribution of parameters/weights will produce an even distribution of points on the Pareto front. The following concepts are required for explaining the technique. The *utopia point* J^* is defined as a vector whose components are all the individual minima J_i^* of the different objectives. It is an unattainable point which is impossible to produce for any problem with conflicting objectives ([9]). The *convex hull of individual minima* (CHIM) is a hyperplane in the objective space that includes all the individual minima of different objectives. The NBI starts with rescaling the objectives by shifting the Utopia point to the origin. The method proceeds by constructing a set of quasi-normal lines to the CHIM. The distance between the CHIM and the utopia point is sought to be maximized along these lines. The furthest distance from a point on the CHIM towards the utopia point J^* in the feasible objectives space is a Pareto optimal point. Therefore, a uniform distribution of the quasi-normal lines should provide a uniform distribution of the Pareto points on the front ([9]). A major drawback of the NBI technique is its inability to vary the resolution of the Pareto front according to the trade-off content in the segment. Thus, representing areas with insignificant trade-offs with the same point's density as areas with significant trade-offs ([22]). Another reported disadvantage is the production of non-Pareto optimal points in some cases, hence the need of invoking a Pareto filter a posteriori to remove these points ([23]).

2.2.3. (Enhanced) normalized normal constraint ((E)NNC)

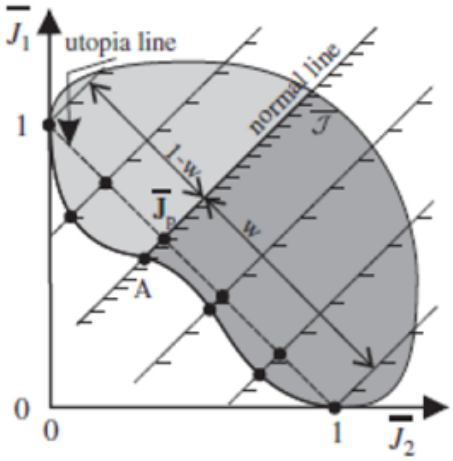


Figure 2: A geometrical illustration of normalized normal constraint method, after [19].

Another method that is capable of providing a uniform representation of the Pareto front is the (enhanced) normalized normal constraint method. Similar to the NBI, it starts with rescaling the objectives so that the utopia point is relocated to the origin. A hyperplane, here named the *utopia plane*, is defined that connects the individual minima points of each objective, analogous to the CHIM. The distinctive feature of the (E)NNC is that it proceeds by minimizing one objective function while the rest of the objective functions are incorporated to the problem as inequality constraints thus reducing the feasible objective space ([23]). The technique starts by normalizing all objectives. In the NNC the normalization is performed by shifting and scaling of all the individual objectives. However, this strategy suffered from limitations when handling problems with more than two objectives as it has not been able to produce a uniform distribution of points on the Pareto front. This has been addressed by [26], the resulting ENNC is an extension of NNC method where

adapted normalization strategies are used such that uniform Pareto Fronts can be obtained even at three or more objectives problems.

2.3. Posterior Pareto front analysis

As both NBI and ENNC can produce non-Pareto optimal points under certain conditions, there is a need for applying a Pareto filter to the generated set of solutions. This filter should remove all points that are dominated by other points in the computed set. A reduced set is produced containing only global Pareto optimal solutions ([23]). Before introducing the *global Pareto filter* algorithm, the concepts of a global Pareto point and local Pareto point are explained introduced ([23]): a solution point P^* is a global Pareto point if there are no other point P in the feasible design space such that $P_\mathrm{j} \leq P_\mathrm{j}^*, \forall\, \mathrm{j} \in \{1, 2, ..., \mathrm{m}\}$ **and** $\exists\, \mathrm{j} \in \{1, 2, ..., \mathrm{m}\} : P_\mathrm{j} < P_\mathrm{j}^*$. On the other hand, a solution point, P^* is a *local* Pareto point if there is no other point P in the *neighborhood* of P^* such that $P_\mathrm{j} \leq P_\mathrm{j}^*, \forall\, \mathrm{j} \in \{1, 2, ..., \mathrm{m}\}$ **and** $\exists\, \mathrm{j} \in \{1, 2, ..., \mathrm{m}\} : P_\mathrm{j} < P_\mathrm{j}^*$ ([23]). A global Pareto filter works by exhaustively comparing every pair of solutions of the initial Pareto set together. By removing any solution that gets dominated, the algorithm retains only the solutions that never get dominated by any other point in the set, thus, the output of the algorithm is a set of global Pareto points ([22], [23]). As all retained points are Pareto optimal, they can be considered as mathematically equivalent in the sense that no point is dominated by any other one. However, it is still possible to facilitate the analysis for a decision maker (DM) by reducing the number of the solutions in the set ([2], [22]). Knowing that the DM is usually interested in a specific significant trade-off level between the solutions, a smart filter can be used to keep only these solutions which exhibit significant trade-offs among them. Thus, reducing the size of the Pareto set by keeping only the points which have practical importance to the decision maker ([22]). It works by making pairwise comparisons on a set of globally Pareto optimal points. The goal of these comparisons is to filter out the solutions that can be deemed less useful by finding if a solution lies in the PIT-region, i.e., the region of *practically insignificant trade-offs*, of any other solution. The PIT-region is defined using two parameters Δt and Δr. The first parameter, Δt, quantifies what the user deems as a significant trade-off. By keeping points with differences between correposing objectives higher than Δt, the user controls the resolution of the Pareto front representation. The lower its value, the higher the resolution. Optionally, the user can supply a second parameter Δr, such that for any two points, the difference between any of the corresponding objectives should be higher than Δr. This parameter is used to control the density of points in the flat regions, as Δr is increased, the more distant the points in a flat segment from each other. Based on the PIT-region definition, two sets are obtained, the set of rejected points, which contains all solutions that are found to be in any PIT-region and the set of *smart Pareto points*, solutions with significant trade-offs between every pair of them ([22]). The advantage of a smart Pareto filter is that it produces a smaller set of practically interesting solutions to the decision maker. However, the drawback of this approach is that it is applied a posteriori to a large set of solutions which means that a lot of computational effort is needed to be invested first in obtaining insignificant solutions that are later removed ([16]).

2.4. Solution of dynamic optimization problems

The dynamic optimization problems in this paper are tackled by a *first discretizing and then optimizing* strategy. An *orthogonal collocation* ([5]) approach is followed, in which simulation and optimization problems are solved simultaneously. Both the control and the states of the problem are discretized. Consequently, the resulting nonlinear program is solved using an interior point optimization tool ([30]).

The employed software tool is Pomodoro ([4]) which is a software toolkit for solving dynamic optimization problems that has been developed at the BioTeC+ division of KU Leuven. It also contains a collection of algorithms for solving MOOCPs. In particular, several scalarization multi-objective optimization algorithms like WS, NBI and ENNC are available ([29]). Though, it should be noted that the calculation of derivatives involved within the solution process is performed by another specialized software kit, CasADi ([1]). CasADi is an open source software tool for automatic differentiation which can efficiently be exploited by dynamic optimization problems. It is written in self-contained C++ code with front ends to Python, Matlab and Octave, giving the users the flexibility to work in the programming language of their choice ([1]).

3. A novel algorithm for a fast representation of a Pareto front with adaptive resolution

3.1. Pareto optimality relevance conditions

First, before proceeding to explaining the algorithm's structure, the conditions under which a Pareto point can be considered significant with respect to another point are constructed. It is assumed that a DM is interested in a prespecified trade-off level t. In this section, the concept of quasi-dominance, first introduced by [15], is used to quantify the relevance of a given pair of Pareto points P_i and P_k in two dimensions. The relevance condition will then be expanded to m dimensions. First, the concept of quasi-dominance is introduced ([15]):

Definition 1. *A Pareto point P_i is said to quasi-dominate another Pareto point P_k if P_i significantly outperforms P_k in at least one objective, i.e. making an improvement higher than or equal t, while P_k does not significantly outperform P_i at any objective. For a minimization problem, this implies that there exists a set of objectives θ where $P_\mathrm{k}^{J_\mathrm{a}} - P_\mathrm{i}^{J_\mathrm{a}} \geq t$, $a \in \theta$, while for the remaining objectives $\mid P_\mathrm{i}^{J_\mathrm{b}} - P_\mathrm{k}^{J_\mathrm{b}} \mid < t$, $b \in \{1, 2, ..., m\} \setminus \theta$, where m is the number of objectives.*

This condition is formulated as follows:

$$P_\mathrm{k}^{J_\mathrm{a}} - P_\mathrm{i}^{J_\mathrm{a}} \geq t, a \in \theta \quad \textbf{and} \quad \mid P_\mathrm{k}^{J_\mathrm{b}} - P_\mathrm{i}^{J_\mathrm{b}} \mid < t, b \in \{1, 2, ..., m\} \setminus \theta \tag{9}$$

Using the quasi-dominance definition, the relevance of a given pair of Pareto points is formulated in bi-objectives and m-objectives problems.

3.1.1. Considering bi-objectives problems

In two dimensional objective space, two points P_i and P_k are said to not quasi-dominate each other if point P_i significantly outperforms P_k at one objective while P_k significantly outperforms P_i at the other objective. This is expressed by the following equation:

$$\min\{(\mid P_\mathrm{i}^{J_1} - P_\mathrm{k}^{J_1} \mid, \mid P_\mathrm{i}^{J_2} - P_\mathrm{k}^{J_2} \mid\} \geq t \tag{10}$$

Subsequently, the significance of two Pareto points relative to each other can be quantified by calculating $d_{\mathrm{i,k}}$, defined as following:

$$d_{\mathrm{i,k}} = \min\{(\mid P_\mathrm{i}^{J_1} - P_\mathrm{k}^{J_1} \mid, \mid P_\mathrm{i}^{J_2} - P_\mathrm{k}^{J_2} \mid\} \tag{11}$$

Now, simply comparing $d_{\mathrm{i,k}}$ to the prespecified trade-off level t quantifies the significance of the two points. Two cases can occur:

- If $d_{\mathrm{i,k}} \geq t$ the two points do not quasi-dominate each other, both points are significant

- If $d_{\mathrm{i,k}} < t$ one point quasi-dominates the other. The trade-off level is below what the DM is interested in.

The second case corresponds to the insignificance criterion used in the smart filter ([22]).

3.1.2. Considering high dimensional problems

Using the quasi-dominance definition, in m-dimensional space, two points P_{i} and P_{k} are said to not quasi-dominate each other if P_{i} significantly outperforms P_{k} in objectives subset θ, while P_{k} significantly outperforms P_{i} in objectives subset ω, where $\theta, \omega \subset 1, 2, ..., m$ and $\theta, \omega \neq \phi$. This condition can be formulated as follows:

$$P_{\mathrm{k}}^{J_{\mathrm{a}}} - P_{\mathrm{i}}^{J_{\mathrm{a}}} \geq t \quad \textbf{and} \quad P_{\mathrm{i}}^{J_{\mathrm{b}}} - P_{\mathrm{k}}^{J_{\mathrm{b}}} \geq t, \quad \forall a \in \theta, \quad \forall b \in \omega \tag{12}$$

Consequently, the two points satisfy the condition of not quasi-dominating one another when this condition is satisfied:

$$\min\{\max(R), \max(W)\} \geq t \tag{13}$$

Where R is the set of trade-offs for the θ objectives at which P_i outperforms P_k and W is the set of trade-offs for the ω objectives at which P_k outperforms P_i, defined as follows:

$$R : \{P_{\mathrm{k}}^{J_{\mathrm{a}}} - P_{\mathrm{i}}^{J_{\mathrm{a}}}, \quad \forall a \in \theta\} \tag{14}$$

$$W : \{P_{\mathrm{i}}^{J_{\mathrm{b}}} - P_{\mathrm{k}}^{J_{\mathrm{b}}}, \quad \forall b \in \omega\} \tag{15}$$

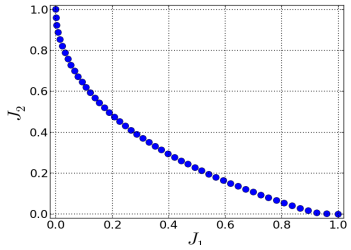
Therefore, to establish the relevance of two points P_{i} and P_{k} in m-dimensional space, $d_{\mathrm{i,k}}$ is calculated as follows:

$$d_{\mathrm{i,k}} = \min\{\max(R), \max(W)\} \tag{16}$$

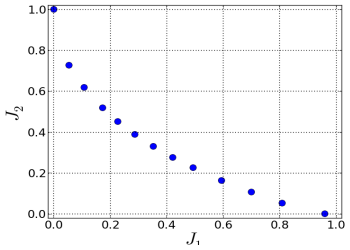
Similar to the bi-objectives case, comparing $d_{\mathrm{i,k}}$ with the prespecified trade-off level t determines if one point is significant relative to the other or not.

3.2. A divide and conquer algorithm formulation

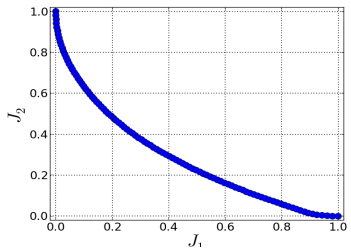
As mentioned previously, a smart Pareto filter aims at finding the interesting regions in the Pareto front through post analysis of the produced optimal solutions. Pareto points are compared and if a point is found to be in the vicinity of another point, it is removed. The result is a higher points density in interesting *knee* regions, in which points have significant trade-offs compared to each other and lower points density in flat regions. Thus, a Pareto front with adaptive resolution can be produced. However, this method suffers from two major drawbacks. The first one is the large overhead of insignificant points. This means that computational time is wasted to produce a large amount of points that eventually gets rejected by the filter. Another major disadvantage is the absence of a clear criterion to select the number of points to be produced by the MOO algorithm. A user interested in a specific significant trade-off level has no way to determine a priori the representation resolution to be produced by the MOO algorithm to be successfully filtered. Thus, more computational power could be wasted to filter overly dense representations of the Pareto front. As it can be observed in Figure 3, using excessive number of points provides similar results with the additional cost of removing a larger number of points that are insignificant to the decision maker. A possible way to overcome these drawbacks is by analyzing the solutions obtained during the solution process to direct the optimization algorithm towards more interesting regions. In this regard,



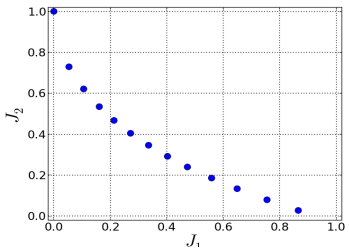
(a) A Pareto front representation generated using NBI, 50 points.



(b) The representation after being filtered using a smart filter with a a specification of 0.05.



(c) A Pareto front representation generated using NBI, 100 points.



(d) The representation after being filtered using a smart filter with a a specification of 0.05.

Figure 3: In the left, Pareto fronts generated by the normal boundary intersection method using 50 and 100 points. The corresponding graphs in the right are the smart filtered sets of 13 points under same specification, 0.05.

we follow a similar direction to the work of [6] in which they introduced an algorithm combining sandwiching and hyperboxing schemes to approximate the Pareto front using a predefined criterion. In this section, a novel algorithm is described to produce a Pareto front with adaptive resolution. Also, strategies to generalize the algorithm to higher dimensions are discussed. First, an efficient Divide and Conquer (D&C) algorithm to tackle multi-objective optimization problems should consist of the following two components:

- *A scheme for a recursive weight distribution:* this scheme should ensure that a uniform distribution of weights that covers the whole weights plane in the objective space is produced at every recursive call. This way, a uniform and complete exploration of the whole Pareto front is guaranteed.
- *A stopping criterion:* the algorithm needs to stop adding points when the new points become insignificant relative to previous results. Or, equivalently, when the amount of information yielded by the new points can be considered insignificant. Hence, segments with low trade-off level can be identified and excluded from the solution process.

3.3. A two dimensional divide and conquer algorithm

A two dimensional algorithm which exploits a divide and conquer approach is discussed first. Instead of obtaining the whole Pareto front and then filtering it, the aim is to reduce the amount of calculations by analyzing the Pareto points after each solution to the optimization problem using a divide and conquer (D&C) approach. A two dimensional D&C algorithm is coupled with a MOO algorithm, e.g., the NBI, to identify and explore the more interesting regions of the Pareto front. The algorithm's working principle is illustrated using Figure 3. It works as

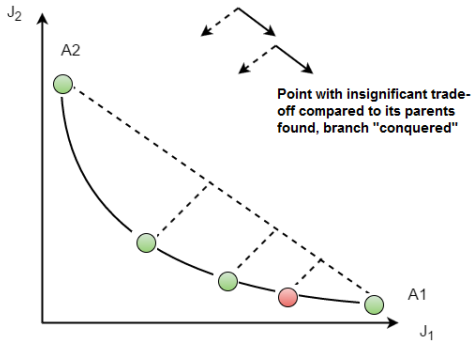
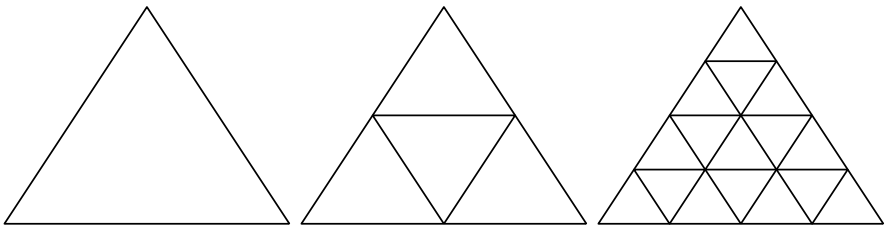


Figure 4: Illustration of the D&C algorithm. The CHIM is divided recursively. For each weight, an optimization sub-problem of maximizing the quasi-normal line is solved to obtain the corresponding Pareto point, the most right recursive branch is shown.

follows: first, the Pareto front is divided into two segments by finding a Pareto point using a set of weights that divided the CHIM line. Then, it continues by recursively dividing each segment to smaller ones. The algorithm stops when a midpoint is produced that is deemed irrelevant to its neighbors using the significance condition defined in Equation 10, based on a significant trade-off value t provided by the user. This way, the solution process is terminated at the segments with a low trade-off level, saving computational effort compared to the posterior use of a smart filter. Moreover, the algorithm does not work by specifying a priori an arbitrary number of points. Instead, the DM enters the filter specification, the t value, i.e., the minimum improvement in an objective to consider a new point relevant. Finally, it should be noted that the recursive algorithm comes at negligible computational cost compared to the process of finding the solutions. An overview of the algorithm structure is provided in Algorithm 1.

3.4. A three dimensional extension: triangular approach



(a) Recursive call: 0 (b) Recursive call: 1 (c) Recursive call: 2

Figure 5: A recursive triangular weight distribution scheme: a triangular weights plane connecting the anchors of the Pareto front is divided recursively to four symmetric equilateral triangles.

In three dimensions, connecting the normalized anchor points of the Pareto front produces a weights plane that resembles a two dimensional equilateral triangle. The recursive weight distribution for the 3D case needs to ensure a uniform distribution of weights that covers the weights triangle in every recursive call. The proposed scheme is inspired by a popular recursive fractal algorithm called the Sierpinski triangle. Fractals are patterns that repeat themselves at different scales. One of the most famous examples of fractals is the Sierpinski triangle which is an equilateral triangle that is subdivided recursively to smaller equilateral triangles ([3]). To obtain a uniform distribution of weights that can be used recursively, a modified version of the Sierpinski triangle is used. In this version, as shown in Figure 5,

Postprint version of paper published in Computers and Chemical Engineering, vol. 106, p. 544-558.
The content is identical to the published paper, but without the final typesetting by the publisher.

Algorithm 1 A 2D divide and conquer algorithm

Input: Significant trade-off level t.

Output: Pareto set with adaptive resolution S
Step 1: Initialization of solution set $S = \{\}$.
Step 2: Construction of anchors weight cell C_{in}:

$$C_{\text{in}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where each row in C_{in} represents a set of objective weights.
Step 3: Initialization of weight cell C_{w}: $C_{\text{w}} = C_{\text{in}}$.
Step 4: Using weights in C_{in}, solve two NBI sub-problem to find the anchor points P_{in1}, P_{in2} and add them to S. Points P_{i}, P_{j} are initialized such that $P_{\text{i}} = P_{\text{in1}}, P_{\text{j}} = P_{\text{in2}}$.
Step 5: Start the recursive function using weight cell C_{w} and Pareto points P_{i}, P_{j} as inputs. Solve the quasi-normal line maximization problem with the weight $(C_{\text{w}}[0] + C_{\text{w}}[1])/2$, the average of the first and the second row of C_{w}, to obtain the Pareto point P_{m}. Then the following condition is checked:
if P_{i}, P_{m} and P_{m}, P_{j} satisfy *significance condition* **do**

1. Add P_{m} to S.
2. *Divide*: two sets of daughter weight cells C_{d1} and C_{d2} are constructed as follows:
$$C_{\text{d1}} = \begin{bmatrix} C_{\text{w}}[0] & \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} \end{bmatrix}^{\top}$$
$$C_{\text{d2}} = \begin{bmatrix} \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} & C_{\text{w}}[1] \end{bmatrix}^{\top}$$
3. Call the recursive function twice by updating inputs to step 5 such that: $C_{\text{w}} = C_{\text{d1}}$, $P_{\text{i}} = P_{\text{i}}$, $P_{\text{j}} = P_{\text{m}}$ and $C_{\text{w}} = C_{\text{d2}}$, $P_{\text{i}} = P_{\text{m}}$, $P_{\text{j}} = P_{\text{j}}$.

else *Conquer*: exit, stop exploring current segment.
Step 6: When all recursive calls are exited, produce S.

all triangles are recursively divided to four symmetrical equilateral triangles. This results in a uniform distribution of points at each and every recursive depth. The second necessity of an efficient divide and conquer algorithm is a stopping criterion. In the two dimensional case, the stopping criterion is straightforward: if the new point is found to be insignificant relative to its two neighbor points, the algorithm stops adding points in this region. On the other hand, if the point is found to be significant, it is added and the solution process continues. In three dimensions however, the only way to assess the significance of a new point is to compare it within some metric (e.g., Euclidean distance) of points surrounding it, which is clearly impractical. Therefore, a number of possible alternative stopping criteria are considered:

1. **Direct comparison criterion**
   This can be considered as a simple extension of the two dimensional case, all new points are compared to their parent points by using the significance condition defined in Equation 13. If a new point has been found to be insignificant compared to an existing point, the new point is not added to the final representation and the algorithm stops creating a more refined division.

2. **Centroid criterion**
   Before creating a division, the algorithm calculates the solution at the centroid of the existing cell. If this solution does not satisfy the significance condition with any of the parent points, the algorithm does not create new weights. Using a triangular weight approach, the Pareto point corresponding to the centroid of a triangle P_{centroid} can be calculated by solving a NBI sub-problem using the set of weights W_{centroid}. For the weight cell $C_{\text{w}\{\text{A,B,C}\}} = [W_{\text{A}}\ , W_{\text{B}}\ , W_{\text{C}}]^{\top}$, W_{centroid} is obtained from the following relation:

$$W_{\text{centroid}} = \frac{W_{\text{A}} + W_{\text{B}} + W_{\text{C}}}{3} \tag{17}$$

   After that, P_{centroid} is compared with the solutions in the Parent cell, $C_{\text{P}} = [P_{\text{A}}\ , P_{\text{B}}\ , P_{\text{C}}]^{\top}$, if the stopping condition is satisfied, the algorithm stops exploring C_{P}.

3. **Information criterion**
   For a set of new points, the variances of each objective are calculated. These variances reflect the trade-off level/information with respect to every objective for the new points. Therefore, using prespecified conditions by the user, the algorithm stops exploring the cell if it is deemed to have sufficient amount of information. The formulation of such an information criterion is discussed hereafter in detail.

Consider a solution cell consisting of N Pareto points, $C_{\text{P}} = [P_1\ , ..., P_{\text{i}}, ...\ , P_{\text{N}}]^{\top}$, each point consists of m components, $P_{\text{i}} = [P_{\text{i}}^{J_1}\ , P_{\text{i}}^{J_2}\ , P_{\text{i}}^{J_{\text{k}}} .... , P_{\text{i}}^{J_{\text{m}}}]^{\top}$, where m is the number of objectives. The set of values for a specific objective k across all cell points is defined as $S_{\text{J}_{\text{k}}} = \{P_{\text{i}}^{J_{\text{k}}}, i = 1, 2, ..., N\}$, $N \geq 2$. The information criterion IC can be defined as a vector whose components are the normalized individual variances $\sigma^2_{\text{S}_{\text{J}_1}}, \sigma^2_{\text{S}_{\text{J}_2}}, ..., \sigma^2_{\text{S}_{\text{J}_{\text{m}}}}$ for the sets $S_{\text{J}_1}, S_{\text{J}_2}, ..., S_{\text{J}_{\text{m}}}$:

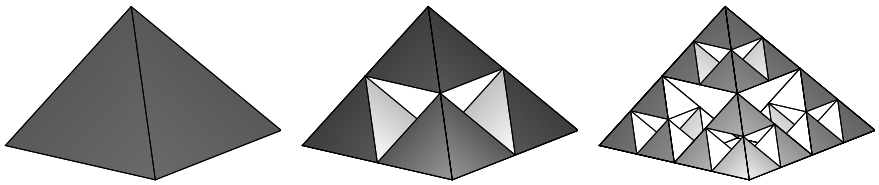
$$IC = [\sigma^2_{\text{S}_{\text{J}_1}}, \sigma^2_{\text{S}_{\text{J}_2}}, ..., \sigma^2_{\text{S}_{\text{J}_{\text{m}}}}]^{\top} \tag{18}$$

$$\sigma^2_{\text{S}_{\text{J}_{\text{k}}}} = \frac{1}{N}\sum_{i=1}^{N}\left(P_{\text{i}}^{J_{\text{k}}} - \frac{1}{N}\sum_{i=1}^{N} P_{\text{i}}^{J_{\text{k}}}\right)^2 \tag{19}$$

IC can be used as a criterion to terminate the recursive algorithm. When trade-off level with respect to all objectives of a certain cell, expressed by the calculated variances, meet the user's prespecified conditions, the cell keeps its last size and the algorithm stops exploring this region. One possible set of conditions to characterize trade-offs using this criterion is that the user enters a minimum trade-off variance σ^2_{low} and a maximum trade-off variance σ^2_{high}. For a given solution cell, if $\min IC < \sigma^2_{low}$ and $\max IC < \sigma^2_{high}$, the stopping criterion is satisfied. The two parameters σ^2_{low} and σ^2_{high} control the points density of the curve and the representation of the plateaus in 3D respectively. The information criterion is applied on a 3D case study in order to visually demonstrate the influence of varying the two parameters in Section 5. Finally, an overview of the triangular based algorithm is presented in Algorithm 2.

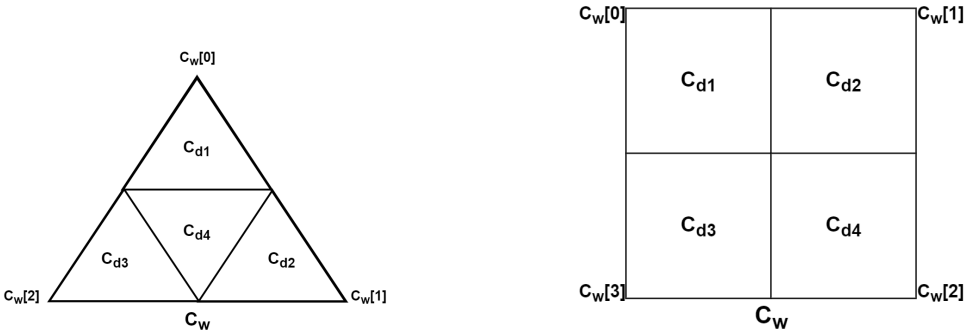
3.5. A three dimensional extension: square based approach

While a triangular weight distribution scheme could work for 3D problems, it cannot be generalized to higher dimensions. In m objectives problems, the weights hyperplane is a regular simplex. Recursively dividing a simplex it into smaller



(a) Recursive call: 0 (b) Recursive call: 1 (c) Recursive call: 2

Figure 6: Recursive division of a 3-simplex, a tetrahedron, using Sierpinski scheme. The space between the four daughter cells can not be filled by a tetrahedron and it will continue to grow with each additional recursive call.



(a) Triangular weight cell (b) Square based weight cell

Figure 7: Weight distribution in case of triangular versus square based weight cells.

simplexes is flawed theoretically. As shown in Figure 6, while a triangle can be filled by smaller symmetric triangles, a simplex cannot be filled by smaller symmetric simplexes ([27]). This means that the algorithm is "blind" to some spaces/gaps between the smaller simplexes. The weight distribution scheme is flawed, complete exploration of space is not possible and no uniform distribution of points on the Pareto front is attainable through this scheme. Therefore, the need arises for an alternative weight distribution scheme that can be generalized to higher dimensions.

A square based D&C algorithm is proposed for three dimensional problems. The algorithm starts by creating a square weight cell that encloses the triangular CHIM, a possible configuration is shown in Figure 8. The initial cell then gets divided recursively into four daughter cells. For every weight cell, if a weight lies in the CHIM, it is used to obtain a Pareto point by solving an NBI sub-problem. A stopping criterion is used to evaluate the trade-off level of the Pareto points corresponding to the weight cell. The process continues recursively till one of the following conditions occur:

- A weight cell is created that lies completely outside the triangular CHIM.
- The stopping criterion is activated indicating that the trade-off level at the Pareto segment associated with a certain weight cell has reached the desired resolution.

An overview of the square based algorithm is presented in Algorithm 3.

The main advantage of a square based algorithm is that it can be easily generalized to m dimensions by using a hypercube based approach. In m dimensions, the CHIM is a simplex and the initial weight cell in the algorithm is a hypercube

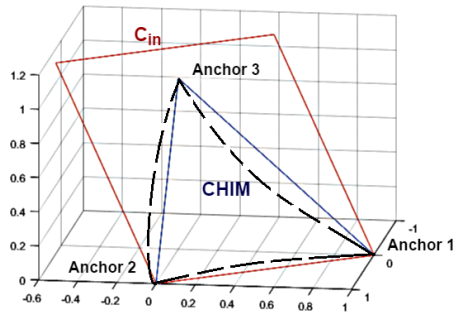


Figure 8: An illustration of the initial square weight cell which encloses the CHIM of the Pareto front. Subsequently, the square weight cell is divided recursively; when a weight is found to lie on the CHIM, a NBI sub-problem is solved using this weight to find the corresponding Pareto point. The recursive process continues till a stopping criterion is met.

of 2^{m-1} vertices enclosing the CHIM. The algorithm proceeds similar to the 3D case by dividing the weight hypercube recursively, solving the NBI sub-problems and testing the obtained Pareto points using the prespecified stopping criterion. A general disadvantage of the hypercube based approach is the occurrence of *rough boundaries*, i.e., the representation of the Pareto boundaries could have an irregular point density. A discussion of this drawback is conducted in Section 5.2. An overview of the general algorithm is presented in Algorithm 4.

4. Case studies

The algorithm is evaluated by applying it to a number of scalar problems as well as a dynamic optimization problem. This section introduces the formulation of the case studies tackled within this paper.

4.1. Scalar benchmark problems

Three scalar MOOPs characterized with large variations in the slope of their Pareto fronts are discussed.

4.1.1. A numerical bi-objective problem

The first benchmark problem that is used in this paper is the bi-objective problem utilized by [22] to illustrate the performance of the smart filter. The Pareto front of this problem is characterized by a sharp knee and long plateaus. The problem is formulated as follows ([22]):

$$\min_x \{x_1, x_2\} \tag{21}$$

subject to:

$$\left(\frac{x_1 - 10}{10}\right)^8 + \left(\frac{x_2 - 5}{5}\right)^8 - 1 \leq 0 \tag{22}$$

$$-10 \leq x_1 \leq 10 \tag{23}$$

$$-10 \leq x_2 \leq 10 \tag{24}$$

4.1.2. DO2DK problem

This problem is formulated by [7] based on the DLTZ functions previously introduced in [12] and [14]. A notable characteristic of this example is the ability

to control the shape of the resulting Pareto front by manipulating the problem's parameters. The DO2DK problem is formulated as follows ([7]):

$$\min_x \{J_1, J_2\} \tag{25}$$

where:

$$J_1(x) = g(x)r(x_1)(\sin(\pi * x_1/2^{s+1} + (1 + \frac{2^s - 1}{2^{s+2}} * \pi) + 1) \tag{26}$$

$$J_2(x) = g(x)r(x_1)(\cos(\pi * x_1/2 + \pi) + 1) \tag{27}$$

subject to:

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_{\mathrm{i}} \tag{28}$$

$$r(x_1) = 5 + 10(x_1 - 0.5)^2 + \frac{1}{k}\cos(2k\pi x_1)2^{s/2} \tag{29}$$

$$0 \leq x_{\mathrm{i}} \leq 1, \quad i = 1, 2, ..., n \tag{30}$$

Where n is the number of variables, s is a parameter that controls the skewness of the front and k is a parameter that controls the number of knees in the Pareto front. The ability to generate a Pareto front with multiple knees can be used to test the capability of a MOO algorithm to locate these segments ([7]).

4.1.3. A numerical 3 objectives problem

The last scalar benchmark problem is a three objectives problem that has been used in [11]. It is described by the following equations ([11]):

$$\min_x \{J_1, J_2, J_3\} \tag{31}$$

subject to:

$$J_{\mathrm{i}} = x_{\mathrm{i}}, \quad i = 1, 2, 3 \tag{32}$$

$$x_1 \geq x_2^{-1} + x_3^{-1} \tag{33}$$

$$x_2 \geq x_1^{-1} + x_3^{-1} \tag{34}$$

$$x_3 \geq x_1^{-1} + x_2^{-1} \tag{35}$$

$$0.2 \leq x_{\mathrm{i}} \leq 10, \quad i = 1, 2, 3 \tag{36}$$

4.2. Dynamic case study: William-Otto reactor

The fourth case study is the multi-objective optimization of the William-Otto fed-batch reactor ([31]). The following reactions take place within the reactor:

$$A + B \rightarrow C \tag{37}$$

$$C + B \rightarrow P + E \tag{38}$$

$$P + C \rightarrow G \tag{39}$$

where A is a reactant that is initially present in the reactor, B is fed continuously to the reactor during operation, P and E are the products of the process and C is an intermediate that can react with P to form side product G.

Postprint version of paper published in Computers and Chemical Engineering, vol. 106, p. 544-558.
The content is identical to the published paper, but without the final typesetting by the publisher.

These reactions are highly exothermic, therefore, generated heat is removed via a cooling jacket. The formulation of the MOOCP discussed in this Section is based on the work of [19] and [17].

The objectives of the process is to maximize the production of P and E by controlling the feeding rate of B and the cooling jacket temperature. They are formulated as follows ([19]):

$$\min_u \{J_1, J_2\} \tag{40}$$

$$J_1 = -x_{\mathrm{P}}(t_{\mathrm{f}})V(t_{\mathrm{f}}) \tag{41}$$
$$J_2 = -x_{\mathrm{E}}(t_{\mathrm{f}})V(t_{\mathrm{f}}) \tag{42}$$

The dynamics of the reactor are described by the following set of equations, ([19]):

$$\frac{dx_{\mathrm{A}}}{dt} = \frac{x_{\mathrm{A}}u_1}{1000V_{\mathrm{R}}} - k_1\eta_1 x_{\mathrm{A}}x_{\mathrm{B}} \tag{43}$$
$$\frac{dx_{\mathrm{B}}}{dt} = \frac{(1-x_{\mathrm{B}})u_1}{1000V_{\mathrm{R}}} + k_1\eta_1 x_{\mathrm{A}}x_{\mathrm{B}} - k_2\eta_2 x_{\mathrm{B}}x_{\mathrm{C}} \tag{44}$$
$$\frac{dx_{\mathrm{C}}}{dt} = \frac{-x_{\mathrm{C}}u_1}{1000V_{\mathrm{R}}} + k_7\eta_1 x_{\mathrm{A}}x_{\mathrm{B}} - k_3\eta_2 x_{\mathrm{B}}x_{\mathrm{C}} - k_6\eta_3 x_{\mathrm{C}}x_{\mathrm{P}} \tag{45}$$
$$\frac{dx_{\mathrm{P}}}{dt} = \frac{-x_{\mathrm{P}}u_1}{1000V_{\mathrm{R}}} + k_2\eta_2 x_{\mathrm{B}}x_{\mathrm{C}} - k_4\eta_3 x_{\mathrm{C}}x_{\mathrm{P}} \tag{46}$$
$$\frac{dx_{\mathrm{E}}}{dt} = \frac{-x_{\mathrm{E}}u_1}{1000V_{\mathrm{R}}} + k_3\eta_2 x_{\mathrm{B}}x_{\mathrm{C}} \tag{47}$$
$$\frac{dx_{\mathrm{G}}}{dt} = \frac{(T_{\mathrm{F}}-T)u_1}{1000V_{\mathrm{R}}} + k_5\eta_3 x_{\mathrm{C}}x_{\mathrm{P}} \tag{48}$$
$$\frac{dT}{dt} = \frac{x_{\mathrm{A}}u_1}{1000V_{\mathrm{R}}} + k_8\eta_1 x_{\mathrm{A}}x_{\mathrm{B}} + k_9\eta_2 x_{\mathrm{B}}x_{\mathrm{C}} + k_{10}\eta_3 x_{\mathrm{C}}x_{\mathrm{P}} - l_1(T-1000u_2) \tag{49}$$
$$\frac{dV_{\mathrm{R}}}{dt} = \frac{u_1}{1000} \tag{50}$$

where x_{i}, $i \in \{A, B, C, P, E, G\}$, is the dimensionless concentration of the reagent, T is the reactor temperature, V_{R} is the reactor volume, u_1 is the feeding rate of B, u_2 is the temperature of the cooling jacket, k_{j}, $j \in \{1, 2, ..., 10\}$ is the pre-exponential reaction constant and η_1, η_2, η_3 are the Arrhenius terms for the three reactions respectively which depend on the temperature.

Finally, the system is subject to the following operational constraints, [19]:

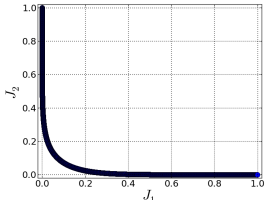
$$60 \leq T(t) \leq 90 \tag{51}$$
$$0 \leq u_1(t) \leq 5.784 \tag{52}$$
$$0.02 \leq u_2(t) \leq 0.1 \tag{53}$$
$$V_{\mathrm{R}}(t_{\mathrm{f}}) \leq 5 \tag{54}$$

5. Simulation results

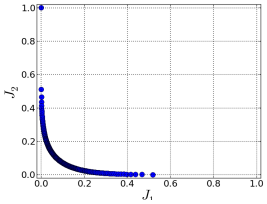
In this section the numerical results illustrating the presented algorithm are presented. First, the D&C algorithm is compared to the smart filter approach

in bi-objectives problems. Subsequently its performance in three dimensions is discussed.

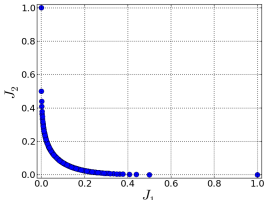
5.1. Bi-objectives problems



(a) Pareto front representation generated using NBI, 2000 points.



(b) A smart Pareto set of 254 points produced by filtering a dense representation using a smart filter with a normalized specification of 0.0005.



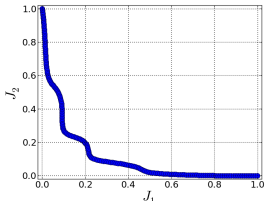
(c) A smart Pareto set of 232 points produced directly using a D&C algorithm of normalized specification of 0.0005.

Figure 9: Comparing normalized Pareto front representations produced by the smart filter and the D&C algorithm for the numerical bi-objective problem.

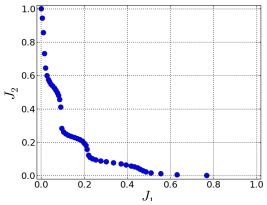
First, the D&C algorithm is compared to the a posteriori use of the smart filter using the numerical bi-objective problem that has been originally used to illustrate the concept of operation of the smart filter algorithm in [22]. The Pareto front of this problem is characterized by a sharp, high trade-off region, knee and two long, low trade-off regions, plateaus. The aim is to obtain a representation with "adaptive" resolution in which the number of points is proportional to the trade-off level/information content of the Pareto segment.

As shown in Figures 9(a) and 9(b), a smart filter with a specification of 0.0005 is used to filter a 2000 points dense normalized Pareto front that has been obtained using the NBI method. This is the least dense representation for the filter to be fully functional. For every two neighboring smart points in the filtered set, at least one point in the original set has been removed between them. In Figure 9(c), a similar representation is obtained using the D&C algorithm with the same specification. However, as illustrated in Figure 12(a), the number of overhead points, points that are removed from the filtered representation, is much lower in the D&C algorithm compared to the smart filter. In case of filtering a 2000 points dense Pareto front, the smart filter removed 1746 points to produce a representation of 254 points. This means that only 12.7% of the calculated solutions are deemed significant for the DM. This percentage also decreases in case of using higher guesses for the number of points to be produced in the initial dense representation. On the other hand, using the D&C algorithm, the percentage of the significant points is always 50% of the calculated solutions. Due to the lower overhead, the D&C algorithms has consistently higher speed than the smart filter strategy, around 431% higher speed in case of 2000 points dense initial representation. The reduced overhead is explained by the recursive nature of the D&C algorithm where the search for additional points is halted once a Pareto segment is found to be low informative.

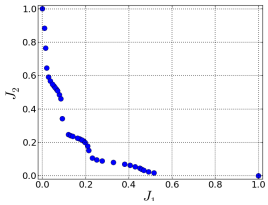
The D&C algorithm is subsequently tested on the DO2DK problem while fixing the number of knees parameter to be $k = 4$. As shown in Figure 10, the algorithm has been able to successfully represent all the four non-convex knees of the Pareto front with an overhead of 36 points compared to at least an overhead of 151 points when using smart filtering of a densely represented normalized Pareto front, corresponding to a 271% improvement in computational effort. A more detailed comparison of the insignificant solutions overhead using both approaches is



(a) Pareto front representation generated using NBI, 198 points.



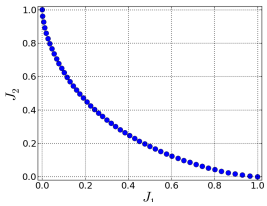
(b) A smart Pareto set of 47 points produced by filtering a dense representation using a smart filter with a normalized specification of 0.005.



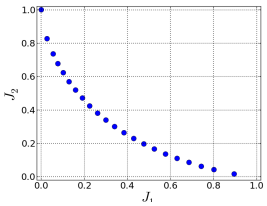
(c) A smart Pareto set of 37 points produced directly using a D&C algorithm of normalized specification of 0.005.

Figure 10: Comparing normalized Pareto front representations produced by the smart filter and the D&C algorithm for DO2DK problem.

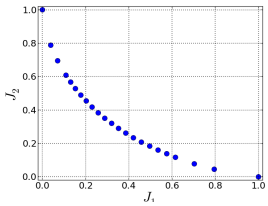
provided in Figure 11(b). This example shows that the D&C algorithm can operate successfully on Pareto fronts with non-convex regions provided that no non-global Pareto optimal points exist on the frontier.



(a) Pareto front representation generated using NBI, 51 points.



(b) A smart Pareto set of 22 points produced by filtering a dense representation using a smart filter with a normalized specification of 0.02.

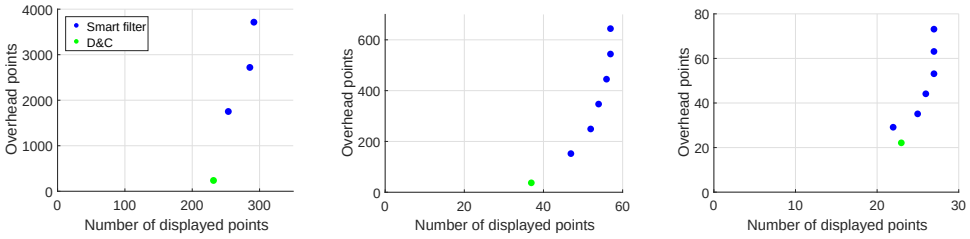


(c) A smart Pareto set of 23 points produced directly using a D&C algorithm of normalized specification of 0.02.

Figure 11: Comparing normalized Pareto front representations produced by the smart filter and the D&C algorithm for William - Otto problem.

The last example is the dynamic multi-objective optimization of the William-Otto reactor where the objectives are selected to be maximizing P and E. The Pareto front of this problem is characterized by high curvature in the middle of the curve. In Figure 11, the smart filter and the D&C algorithm are used to obtain normalized Pareto fronts with adaptive resolutions using a specification of 0.02. For representations with similar quality, the D&C algorithm produced less overhead points than the a posteriori use of a smart filter, a comparison of the overhead points produced using each technique is provided in Figure 12(c).

Figure 13 shows the D&C algorithm's speed compared to the smart filter strategy. The gain in speed is a direct result of the less overhead of insignificant points and it depends on two factors. The first is the number of points in the initial representation. As the number of points in the initial representation increase, the gain in speed using the D&C algorithm compared to filtering increases. This is a result of the more computational effort that is invested in obtaining solutions that will get subsequently filtered. The second factor is the Pareto front shape. The gain in speed is greater in Pareto fronts where large variations in slope exist. The segments with low slope are characterized by high density of insignificant solutions that are produced in the initial representation. On the other hand, the D&C algorithm produces optimized number of insignificant solutions as it stops exploring low trade-off

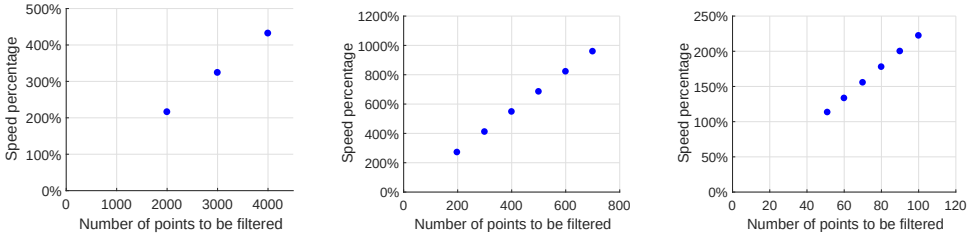


(a) Numerical bi-objective problem (b) DOD2K problem (c) William-Otto problem

Figure 12: Number of overhead points versus representation points when using a smart filter compared to the D&C algorithm for different examples.

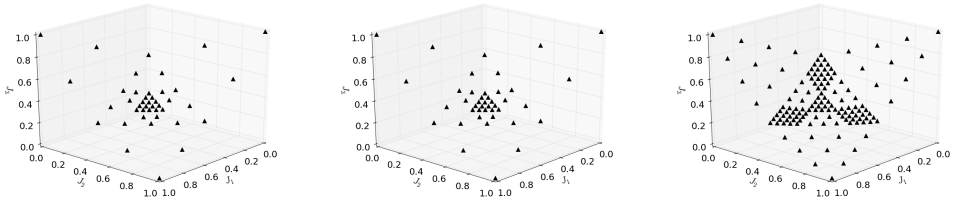
segments once identified.

In summation, for a DM interested in practical significant trade-offs between P and E production, instead of guessing a dense resolution for an initial representation to be filtered, he/she can enter directly the t value of interest to the D&C algorithm. Subsequently, this leads to reducing the computational time invested in non significant solutions, in addition to the more simple intuitive solution procedure compared to the smart filter strategy.



(a) Numerical bi-objective problem (b) DOD2K problem (c) William-Otto problem

Figure 13: D&C algorithm's speed as a percentage of the a posteriori analysis of a dense Pareto front by a smart filter for different dense raw presentations.



(a) Direct comparison criterion with specification t = 0.05. (b) Centroid criterion with specification t = 0.01. (c) Information criterion with specifications $\min IC < 0.005$, $\max IC < 0.2$.

Figure 14: Using D&C algorithm for the numerical three objectives problem: triangular approach.

5.2. Three objectives problem

An extension of the D&C algorithm in three dimensions has been tested on the numerical [illegible] while

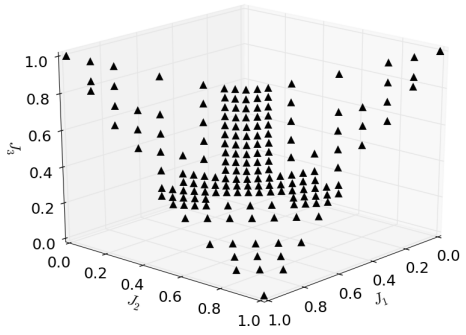
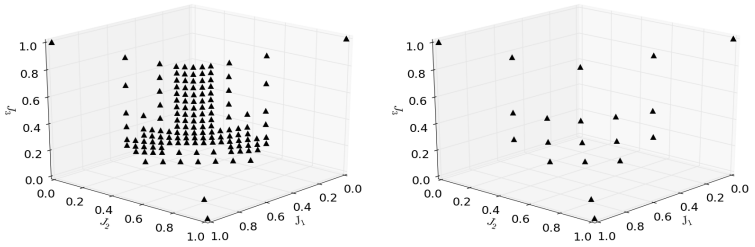


Figure 15: Using D&C algorithm for the numerical three objectives problem: square based approach

applying a triangular scheme. As shown in Figure 14, similar representations are produced where the points density is highest in the curved region at the center of the Pareto front. However, this is achieved at different specifications as the three criteria have different concepts of operation. In the direct comparison criterion, the three points corresponding to a triangular weight cell are compared together to test for significance. On the other hand, when using the centroid criterion, an additional point is created corresponding to the center of the weight cell for the significance test. This means that a lower value for t is used to produce representations with similar density as the direct comparison criterion. Also, the centroid points are not produced at the final representation. Finally, the information citerion uses two specifications $\min IC$ and $\max IC$ which can be used to fine-tune the density of the final representation according to the user's preferences.

The square based approach is tested on the same example using the information criterion. The terminating conditions have been chosen to be $\min IC < 0.0005$ and $\max IC < 0.02$. In Figure 15, the points density is significantly higher at the curved region of the Pareto front than its flat areas. Comparing the representation produced by the square based approach in Figure 15 with its counterpart in Figure 14(c) yields two main observations. First, representations produced by the square based approach has higher points density, 179 points compared to 132 points when using the triangular approach. The second observation is the irregularities in the points density near the boundaries of the Pareto front when using the square based method. This is a result of the simple, center of mass, cutting strategy used in this approach. A square that has one vertex lying in the CHIM will keep getting divided till it gets another vertex in the CHIM to undergo the stopping criterion, resulting in two points that are close to each other in the final representation at the boundaries of the Pareto front. However, while the triangular scheme is more natural in the three objectives problems, only the square based scheme can be generalized to higher dimensions. In Figure 16, the influence of $\min IC$ and $\max IC$ is interpreted graphically. Since plateaus in 3D are segments characterized by absence of trade-offs with respect to one objective, $\min IC$ in such segments is always close to zero. And using a high $\max IC$ value, 0.2 in Figure 16(a), leads to low points density in the plateaus corresponding to high trade-offs between the points with respect to one of the other two objectives. On the other hand, in curved regions, trade-offs exist with respect to all objectives. Therefore, increasing $\min IC$ in Figure 16(b) leads a to lower points density in the curved segment at the center of the Pareto front.

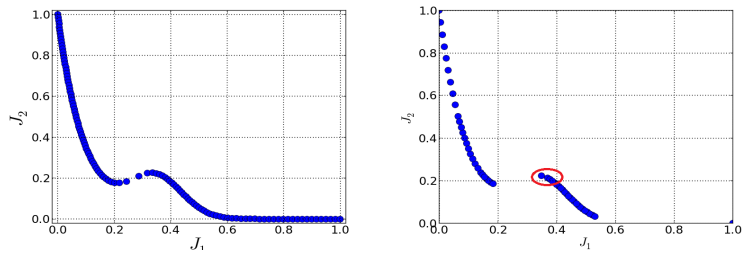
Finally, the use of the D&C algorithm is limited to the cases when solving a NBI sub-problem always yields a globally optimal Pareto points. In extreme cases, where



(a) min $IC < 0.0005$, max $IC < 0.2$. (b) min $IC < 0.005$, max $IC < 0.2$.

Figure 16: The effects of varying min IC and max IC on the representations produced by the D&C algorithm for the numerical three objectives problem. max IC controls the points density in the plateaus while min IC controls the points density in the curved regions of the front.

MOO algorithms can yield locally optimal Pareto points, the performance of the D&C algorithm will deteriorate considerably. In Figure 17, the D&C algorithm has been applied to such case from [23]. Points obtained in the midsection of the front are non globally optimal Pareto points. The recursive approach of the algorithm leads to a high points density in curved segments, convex or non-convex, however, it does not differentiate between segments of global and local Pareto optimality.



(a) NBI (b) D&C

Figure 17: Using the D&C algorithm for extreme Pareto fronts where non globally optimal Pareto points exist

6. Conclusion

In multi-objective optimization, a DM is provided with a set of points to make a decision by considering the trade-offs between different solutions. In order to facilitate the decision making process, a smart filter is used to reduce the solution set by keeping only the solutions with significant trade-offs among them. However, this strategy suffers from the necessity of producing a large number of insignificant solutions in the initial presentation for the filter to work on. This can be computationally very expensive when handling complex optimization problems such as multi-objective optimal control of chemical processes.

In this paper, a divide and conquer approach is implemented in order to obtain a Pareto front with adaptive resolution. By analyzing the Pareto points obtained during the solution process, the algorithm can identify and stop exploring segments with low trade-off level. The technique has been tested and compared to the smart filter strategy on three scalar problems and a benchmark dynamic optimization problem. The D&C approach is based on the idea that Pareto points obtained during the solution process have information about the trade-offs in the segment where they are located. It has been found that using the D&C algorithm has two

main advantages. First, lower overhead of insignificant solutions which translates to an improvement in speed over the smart filter strategy. A second, non-quantifiable, advantage is the more intuitive, trade-off oriented, solution procedure in which the DM only enters the trade-off level he/she is interested in.

Finally, two disadvantages of the D&C approach have been observed. First, when dealing with three or more objectives using a square based approach, inadequate representation of the Pareto boundaries could occur. Additionally, the algorithm inherits the inability of differentiating between local and global Pareto optimality from the NBI technique. In future work, we aim to address these drawbacks by using an adaptive weight cutting strategy to construct weights near the Pareto boundaries. The D&C scheme could also be integrated with concepts from [29] to explore the extreme Pareto boundaries that lie outside the CHIM. Also, one possible strategy to address the second disadvantage is to use techniques from [20] which allow to directly filter out non-Pareto optimal points when using NBI or ENNC.

ACKNOWLEDGMENTS

IH is supported by FWO-SB grant 1S54217N. The research was supported by: PFV/10/002 (OPTEC), FWO-G.0930.13 and BelSPO: IAP VII/19 (DYSCO). The authors would like to thank Mattia Vallerio and Lorenzo Cabianca for productive discussions.

References

[1] Andersson, J., Akesson, J., Diehl, M., 2012. CasADi - a symbolic package for automatic differentiation and optimal control. In: Proceedings of the 6th International Conference on Automatic Differentiation.

[2] Antipova, E., Pozo, C., Gosalbez, G., Boer, D., Cabeza, L., Jimenez, L., 2015. On the use of filters to facilitate the post-optimal analysis of the Pareto solutions in multi-objective optimization. Computers and Chemical Engineering 74, 48 – 58.

[3] Barnsley, M., Hutchinson, J., Stenflo, Ö., 2003. V-variable fractals and superfractals. arXiv preprint math/0312314, 1–17.

[4] Bhonsale, S., Vallerio, M., Telen, D., Vercammen, D., Logist, F., Van Impe, J., 2016. Solace: An open source package for nonlinear model predictive control and state estimation for (bio)chemical processes. Proceedings of European Symposium on Computer Aided Process Engineering (ESCAPE), 1971-1976.

[5] Biegler, L., 2010. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. MOS-SIAM.

[6] Bortz, M., Burger, J., Asprion, N., Blagov, S., Böttcher, R., Nowak, U., Scheithauer, A., Welke, R., Küfer, K.-H., Hasse, H., 2014. Multi-criteria optimization in chemical process design and decision support by navigation on pareto sets. Computers and Chemical Engineering 60, 354 – 363.

[7] Branke, J., Deb, K., Dierolf, H., Osswald, M., 2004. Finding knees in multi-objective optimization. In: Parallel Problem Solving from Nature-PPSN VIII. Springer Berlin Heidelberg, pp. 722–731.

[8] Das, I., Dennis, J., 1997. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Structural Optimization 14, 63–69.

[9] Das, I., Dennis, J., 1998. Normal-Boundary Intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM Journal on Optimization 8, 631–657.

[10] de Hijas-Liste, G., Klipp, E., Balsa-Canto, E., Banga, J., 2014. Global dynamic optimization approach to predict activation in metabolic pathways. BMC Systems Biology 8 (1).

[11] de Motta, R., Afonso, S., Lyra, P., 2012. A modified nbi and nc method for the solution of n-multiobjective optimization problems. Structural and Multidisciplinary Optimization, 1–21.

[12] Deb, K., 1999. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary computation 7 (3), 205–230.

[13] Deb, K., 2001. Multi-Objective optimization using evolutionary algorithms. John Wiley, Chichester, England, UK.

[14] Deb, K., Thiele, L., Laumanns, M., Zitzler, E., 2002. Scalable multi-objective optimization test problems. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02. Vol. 1. IEEE, pp. 825–830.
[15] Dias, L., Clímaco, J., 2000. Additive aggregation with variable interdependent parameters: the vip analysis software. Journal of the Operational Research Society, 1070–1082.
[16] Hancock, B., Nysetvold, T., Mattson, C., 2015. L-dominance: An approximate-domination mechanism for adaptive resolution of pareto frontiers. Structural and Multidisciplinary Optimization 52 (2), 269–279.
[17] Hannemann, R., Marquardt, W., 2010. Continuous and discrete composite adjoints for the hessian of the lagrangian in shooting algorithms for dynamic optimization. SIAM J. Scientific Computing 31 (6), 4675–4695.
[18] Logist, F., Houska, B., Diehl, M., Van Impe, J., 2010. Fast pareto set generation for nonlinear optimal control problems with multiple objectives. Structural and Multidisciplinary Optimization 42, 591–603.
[19] Logist, F., Vallerio, M., Houska, B., Diehl, M., Van Impe, J., 2012. Multi-objective optimal control of chemical processes using ACADO toolkit. Computers and Chemical Engineering 37, 191–199.
[20] Logist, F., Van Impe, J., 2012. Novel insights for multi-objective optimisation in engineering using normal boundary intersection and (enhanced) normalised normal constraint. Structural and Multidisciplinary Optimization 45 (3), 417–431.
[21] Marler, R., Arora, J., 2004. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26, 369–395.
[22] Mattson, C., Mullur, A., Messac, A., 2004. Smart pareto filter: Obtaining a minimal representation of multiobjective design space. Engineering Optimization 36 (6), 721–740.
[23] Messac, A., Ismail-Yahaya, A., Mattson, C., 2003. The normalized normal constraint method for generating the Pareto frontier. Structural and Multidisciplinary Optimization 25, 86–98.
[24] Miettinen, K., 1999. Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston.
[25] Nimmegeers, P., Telen, D., Logist, F., Impe, J. V., 2016. Dynamic optimization of biological networks under parametric uncertainty. BMC Systems Biology 10 (1), 1–20.
[26] Sanchis, J., Martinez, M., Blasco, X., Salcedo, J., 2008. A new perspective on multiobjective optimization by enhanced normalized normal constraint method. Structural and Multidisciplinary Optimization 36, 537–546.
[27] Senechal, M., 1995. Quasicrystals and geometry. Cambridge University Press, Cambridge New York.
[28] Telen, D., Logist, F., Vanderlinden, E., Van Impe, J., 2012. Optimal experiment design for dynamic bioprocesses: a multi-objective approach. Chemical Engineering Science 78, 82–97.
[29] Vallerio, M., Vercammen, D., Van Impe, J., Logist, F., 2015. Interactive NBI and ENNC methods for the progressive exploration of the criteria space in multi-objective optimization and optimal control. Computers and Chemical Engineering 82, 186 – 201.
[30] Wächter, A., Biegler, L., 2006. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming 106 (1), 25–57.
[31] Williams, T., Otto, R., 1960. A generalized chemical processing model for the investigation of computer control. Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 79 (5), 458–473.

Postprint version of paper published in Computers and Chemical Engineering, vol. 106, p. 544-558.
The content is identical to the published paper, but without the final typesetting by the publisher.

Algorithm 2 A 3D divide and conquer algorithm: triangular approach

Input: Stopping criterion specifications σ^2_{low}, σ^2_{high}

Output: Pareto set with adaptive resolution S

Step 1: Initialization of solution set $S = \{\}$.

Step 2: Construction of a weight cell C_{in}:

$$C_{in} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 3: Initialization of weight cell C_{w}: $C_{\text{w}} = C_{\text{in}}$.

Step 4: Start the recursive function using weight cell C_{w} as an input:

for i = 1:3 **do**

Solve a NBI sub-problem using $C_{\text{w}}[i]$ to obtain corresponding Pareto point P_{i}

end for

Step 5: Using obtained Pareto points, construct a Pareto cell C_{P}: $C_{\text{P}} = \begin{bmatrix} P_1 & P_2 & P_3 \end{bmatrix}$. Then the following condition is checked:

if *Stopping criterion* is not activated **do**

1. Add C_{P} points to S
2. *Divide*:construct four daughter cells C_{d1}, C_{d2}, C_{d3}, C_{d4} (see Figure 7(a)) such that:

$$\begin{aligned} C_{\text{d1}} &= \begin{bmatrix} C_{\text{w}}[0] & \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} & \frac{C_{\text{w}}[0]+C_{\text{w}}[2]}{2} \end{bmatrix}^\top \\ C_{\text{d2}} &= \begin{bmatrix} \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} & C_{\text{w}}[1] & \frac{C_{\text{w}}[1]+C_{\text{w}}[2]}{2} \end{bmatrix}^\top \\ C_{\text{d3}} &= \begin{bmatrix} \frac{C_{\text{w}}[0]+C_{\text{w}}[2]}{2} & \frac{C_{\text{w}}[1]+C_{\text{w}}[2]}{2} & C_{\text{w}}[2] \end{bmatrix}^\top \\ C_{\text{d4}} &= \begin{bmatrix} \frac{C_{\text{w}}[1]+C_{\text{w}}[2]}{2} & \frac{C_{\text{w}}[0]+C_{\text{w}}[2]}{2} & \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} \end{bmatrix}^\top \end{aligned} \tag{20}$$

3. Call the recursive function four times by updating the input to step 4 such that: $C_{\text{w}} = C_{\text{d}}, \quad C_{\text{d}} \in \{C_{\text{d1}}, C_{\text{d2}}, C_{\text{d3}}, C_{\text{d4}}\}$.

else *Conquer*: exit, stop exploring current segment.

Step 6: When all recursive calls are exited, produce S.

Postprint version of paper published in Computers and Chemical Engineering, vol. 106, p. 544-558.
The content is identical to the published paper, but without the final typesetting by the publisher.

Algorithm 3 A 3D divide and conquer algorithm: square based approach

Input: Stopping criterion specifications σ^2_{low}, σ^2_{high}

Output: Pareto set with adaptive resolution S

Step 1: Initialization of solution set $S = \{\}$.

Step 2: Construction of a weight cell $C_{\text{in}} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix}^\top$ such that C_{in} is a square cell enclosing the *CHIM*.

Step 3: Initialization of weight cell C_{w}: $C_{\text{w}} = C_{\text{in}}$.

Step 4: Start the recursive function using weight cell C_{w} as an input:

for i = 1:4 **do**

if $C_{\text{w}}[i]$ in *CHIM* **do**

Solve a NBI sub-problem using $C_{\text{w}}[i]$ to obtain corresponding Pareto point P_{i}

end for

Step 5: Using obtained Pareto points, construct a Pareto cell C_{P}: $C_{\text{P}} = \begin{bmatrix} P_1 & \dots & P_{\text{n}} \end{bmatrix}$, $n \leq 4$. Then the following condition is checked:

if *Stopping criterion* is not activated **do**

1. Add C_{P} points to S
2. *Divide*: construct four daughter cells C_{d1}, C_{d2}, C_{d3}, C_{d4} such that:

$$C_{\text{d1}} = \begin{bmatrix} C_{\text{w}}[0] & \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} & \frac{\sum C_{\text{w}}}{4} & \frac{C_{\text{w}}[0]+C_{\text{w}}[3]}{2} \end{bmatrix}^\top$$
$$C_{\text{d2}} = \begin{bmatrix} \frac{C_{\text{w}}[0]+C_{\text{w}}[1]}{2} & C_{\text{w}}[1] & \frac{C_{\text{w}}[1]+C_{\text{w}}[2]}{2} & \frac{\sum C_{\text{w}}}{4} \end{bmatrix}^\top$$
$$C_{\text{d3}} = \begin{bmatrix} \frac{\sum C_{\text{w}}}{4} & \frac{C_{\text{w}}[1]+C_{\text{w}}[2]}{2} & C_{\text{w}}[2] & \frac{C_{\text{w}}[2]+C_{\text{w}}[3]}{2} \end{bmatrix}^\top$$
$$C_{\text{d4}} = \begin{bmatrix} \frac{C_{\text{w}}[0]+C_{\text{w}}[3]}{2} & \frac{\sum C_{\text{w}}}{4} & \frac{C_{\text{w}}[2]+C_{\text{w}}[3]}{2} & C_{\text{w}}[3] \end{bmatrix}^\top$$

3. Call the recursive function four times by updating the input to step 4 such that: $C_{\text{w}} = C_{\text{d}}, \quad C_{\text{d}} \in \{C_{\text{d1}}, C_{\text{d2}}, C_{\text{d3}}, C_{\text{d4}}\}$.

else *Conquer*: exit, stop exploring current segment.

Step 6: When all recursive calls are exited, produce S.

Algorithm 4 A general hypercube based 3D divide and conquer algorithm

Input: Stopping criterion specifications σ^2_{low}, σ^2_{high}

Output: Pareto set with adaptive resolution S

Step 1: Initialization of solution set $S = \{\}$.

Step 2: Construction of a weight cell $C_{\text{in}} = \begin{bmatrix} w1 & w2 & ... & w2^{m-1} \end{bmatrix}^\top$ such that C_{in} is a hypercube cell of 2^{m-1} vertices enclosing the simplex shaped *CHIM*, m is the number of the objectives of the problem.

Step 3: Initialization of weight cell C_{w}: $C_{\text{w}} = C_{\text{in}}$.

Step 4: Start the recursive function using weight cell C_{w} as an input:

for i = 1:2^{m-1} **do**

if $C_{\text{w}}[i]$ in *CHIM* **do**

Solve a NBI sub-problem using $C_{\text{w}}[i]$ to obtain corresponding Pareto point P_{i}

end for

Step 5: Using obtained Pareto points, construct a Pareto cell C_{P}: $C_{\text{P}} = \begin{bmatrix} P_1 & ... & P_{\text{n}} \end{bmatrix}$, $n \leq 2^{m-1}$. Then the following condition is checked:

if *Stopping criterion* is not activated **do**

1. Add C_{P} points to S
2. *Divide*: construct daughter cells C_{d1}, C_{d2}, ... , $C_{\text{d2}^{\text{m}-1}}$.
3. Call the recursive function by updating the input to step 4 such that: $C_{\text{w}} = C_{\text{d}}, \quad C_{\text{d}} \in \{C_{\text{d1}}, C_{\text{d2}}, ..., C_{\text{d2}^{\text{m}-1}}\}$.

else *Conquer*: exit, stop exploring current segment.

Step 6: When all recursive calls are exited, produce S.