A Branching Strategy for Exploring the Objective Space in Bi-objective Optimization Problems ⋆

Ihab Hashem * **Viviane De Buck** * **Seppe Seghers** * **Jan Van Impe** *

* *KU Leuven, Chemical Engineering Department, BioTeC+ & OPTEC, Gebroeders De Smetstraat 1, 9000 Gent, Belgium (e-mail: jan.vanimpe@kuleuven.be)*

Abstract: The process of optimization of chemical/ biochemical processes can often involve multiple conflicting objectives. This gives rise to a class of problems called multi-objective optimization problems. Solving such problems results in an infinite set of points, the Pareto set, which includes all the solutions in which no objective can be improved without worsening at least one other objective. In this paper, we propose a new strategy that is inspired by branching phenomena in nature for exploring the objective space to obtain a representation of the Pareto set. The algorithm starts from a single point in the objective space, and systematically constructs branches towards the Pareto front by solving correspondingly-modified subproblems. This process continues till points that lie at the Pareto front are obtained. This way, it ensures that no region in the objective space gets explored more than a single time. Additionally, using a proximity parameter, the branches density can be controlled, consequently leading to controlling the resolution of the Pareto front. The proposed method has been applied to a numerical bi-objective optimization problem as well as the problem of the bi-objective control of a William-Otto reactor. Results show that the new algorithm has managed to obtain a Pareto front with adaptive resolution where the areas with high trade-offs are represented with higher points density.

Keywords: Multi-objective optimization, Optimal control, Recursive algorithms, Process industry

1. INTRODUCTION

An optimization problem that involves more than one objective is termed a Multi-Objective Optimization Problem (MOOP). In the engineering domain, these objectives are usually conflicting with each other. Hence, while solving a Single Objective Optimization Problem (SOOP) should ultimately result in a single optimal solution, solving a MOOP on the other hand will result in a solution set of mathematically equally optimal solutions, called the Pareto front, where no objective can be improved without worsening at least one other conflicting objective. A subcategory of MOOP that is commonly encountered in the context of the chemical industry is Multi-Objective Optimal Control Problems (MOOCPs). In such problems, the aim is to find the optimal time profile of a control variable in order to optimize a dynamic system with respect to multiple objectives. An example of a MOOCP would be to optimize the control variables of a bioreactor to maximize the production rate and minimize the cost simultaneously. The control variables in this case could be the reactor's feed rate, temperature and agitator speed.

Solving a MOOP corresponds to approximating its Pareto front. For that, two main classes of algorithms are implemented: vectorization and scalarization based methods. Vectorization algorithms are stochastic population-based algorithms that have been inspired by biological evolution and do not require the calculation of gradients (Deb (2001)). Alternatively, scalarization methods work by converting the MOOP into a set of SOOPs, such that solving each of the resulting problems yields a point on the Pareto front (Marler and Arora (2004)). Each SOOP is solved by supplying a solver with initial parameter values. Then, the solver deploys the appropriate optimization algorithm. In this paper, we propose an alternative deterministic algorithm that utilizes a branching strategy to obtain the Pareto front, starting from a single initial guess. The algorithm is then applied to two case studies, a numerical and an industrial one. The influence of the algorithm's key parameters on its performance is investigated. The rest of the paper is structured as follows: Section 2 introduces the formulation of a MOOCP as well as an overview of existing deterministic solution strategies. In Section 3, the case studies are introduced: a numerical bi-objective optimization problem and the optimal control of a chemical

⋆ This work was supported by the KU Leuven Research Council (OPTEC Center-of-Excellence Optimization in Engineering OPTEC and project C24/18/046), by the ERA-NET FACCESurPlus FLEXIBI Project, co-funded by VLAIO project HBC.2017.0176, by the Fund for Scientific Research-Flanders (projects G.0863.18 and G.0B41.21N), and by the European Union's Horizon 2020 Research and Innovation Programme (Marie Sklodowska-Curie grant agreement numbers 813329 and 956126). VDB is supported by FWO-SB grant 1SC0920N.

reactor. The algorithm is applied to both case studies, and the solution process and the results are discussed. Finally, Section 4 summarizes the conclusions of the paper.

2. MATHEMATICAL FORMULATION AND METHODS

A MOOCP can be formulated as follows (Logist et al. (2010)):

$$\min_{u(\epsilon),x(\epsilon),p,\epsilon_\mathrm{f}} \{J_1, J_2, ..., J_\mathrm{m}\} \tag{1}$$

which is subject to:

$$\frac{dx}{d\epsilon} = F(x(\epsilon), u(\epsilon), p, \epsilon) \tag{2}$$
$$0 = b_\mathrm{i}(x(0), p) \tag{3}$$
$$0 = b_\mathrm{t}(x(\epsilon_\mathrm{f}), p) \tag{4}$$
$$0 \geq c_p(x(\epsilon), u(\epsilon), p, \epsilon) \tag{5}$$
$$0 > c_t(x(\epsilon_\mathrm{f}), u(\epsilon_\mathrm{f}), p, \epsilon_\mathrm{f}) \tag{6}$$

with vector u representing the control variables of the process and x as the state variables describing the process. ϵ represents the independent variable of the process, usually space or time, where $\epsilon \in [0, \epsilon_\mathrm{f}]$. And p is a vector which represents the constant parameters of the system. The objectives of the problem are represented by $J_1, J_2, ..., J_\mathrm{m}$, where m is the total number of objectives. The constraints of the system are described via b_i and b_t, which represent the initial and terminal boundary conditions respectively. The path and terminal inequality constraints on the other hand are represented via c_p and c_t. An individual objective function can be formulated as follows:

$$J_\mathrm{i} = M(x(\epsilon_\mathrm{f}), p, \epsilon_\mathrm{f}) + \int_{\epsilon_0}^{\epsilon_\mathrm{f}} L(x(\epsilon), u(\epsilon), p, \epsilon) d\epsilon \tag{7}$$

with M and L representing the Mayer and Lagrange terms respectively. The Mayer term indicates the terminal cost of the objective function. It is used to model objectives such as the conversion rate of the process. The Lagrange term is path dependent, for example, the total heat removal during the process. Finally, the vector y is defined as $y = [x(\cdot)^\top, u(\cdot)^\top, p^\top, \epsilon_\mathrm{f}]^\top$ and represents all the optimization variables of the process. The objectives of the problem are grouped as $J(y) = [J_1(y), J_2(y), ..., J_\mathrm{m}(y)]^\top$. A Pareto optimal set consists of all solutions where there is no other feasible solution that could improve a subset of the objectives without worsening at least one other objective. Therefore, a vector y^* is said to be a Pareto optimal solution if there exists no $y \in S$ such that $J_\mathrm{i}(y) \leq J_\mathrm{i}(y^*)$ for $\mathrm{i} = 1, 2, ..., \mathrm{n}$ and $J_\mathrm{i}(y) < J_\mathrm{i}(y^*)$ for at least one J_i.

According to Marler and Arora (2004), there are two classes of algorithms to obtain a representation of the Pareto front: vectorization algortihms and scalarization algorithms. The most popular examples of vectorization algorithms are genetic algorithms like NSG A-II. Their drawbacks can be summarized in their inability to handle constraints, having a computationally expensive nature and their unreliability in high dimensional spaces (Logist et al. (2013)). Scalarization algorithms work by parameterizing the problem into a set of subproblems using a weights vector. Each subproblem is then solved to obtain a corresponding point on the Pareto front. And then, a filtering algorithm can be used a posteriori to keep the points that are relevant to the Decision Maker (DM) such that the more interesting "knee regions" of the Pareto front, where high tradeoffs exist between solutions, are represented by a higher density of points than the less interesting plateau segments (Mattson et al. (2004)).

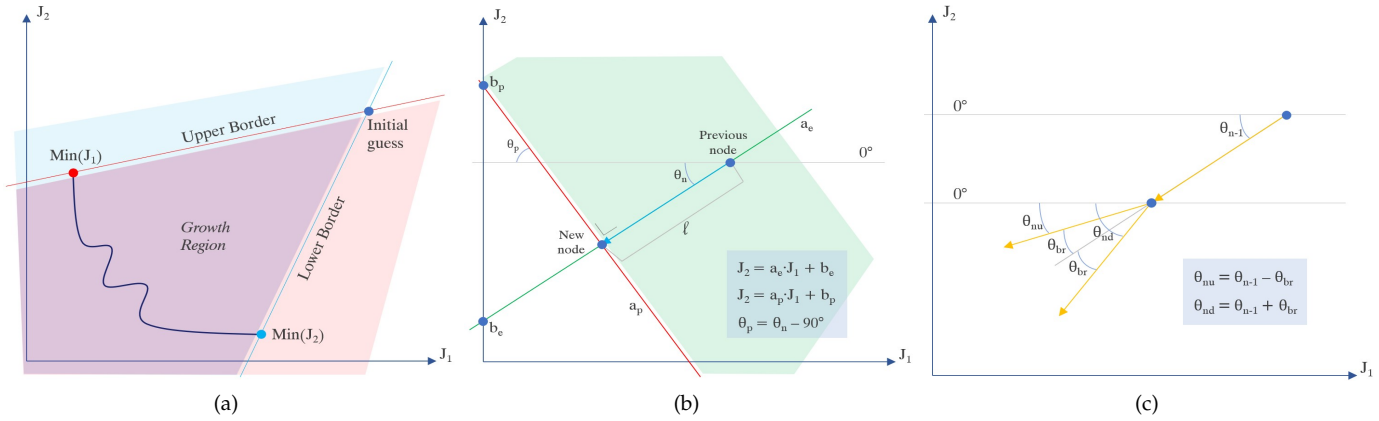
2.1 Branching strategy to explore the objective space

The new algorithm attempts to find points on the Pareto front starting from an Initial Guess (IG), the point in the objective space which corresponds to the initial guess of the parameters values supplied by the user. First, the anchor points corresponding to solving the two SOOPs $\min J_1$ and $\min J_2$ are obtained. The two anchor points, combined with the IG, can be used to determine the boundaries of the growth region, the subset of the objective space where the solution process will take place, shown in Figure 1(a). The algorithm constructs branches in the objective space, starting from IG, by solving a SOOP sub-problem at predefined directions. Hereby, a branch in the objective space corresponds to solving a SOOP subproblem starting from the current point/ node to find a new node that lies at a distance equal to the Branch Length (BL), as shown in Figure 1(b).

To set the angle and the length of the branch, two additional constraints are adjoined to the original problem. An equality constraint is added to specify the branch's direction, described in Figure 1(b) using the slope a_e and the intercept b_e. a_e is determined by the angle θ_n, and b_e is calculated from the previous node and a_e. Secondly, an inequality constraint limits the length of the branch by constructing a second line that is perpendicular to the equality constraint, at a distance BL from the previous node with slope a_p and intercept b_p. Each branch ends at a node in the objective space closer to the Pareto front than its parent. This node will be the initial point for the next subproblem. The iterations are performed through a recursive pattern. In this algorithm, two new branches grow from each node. The directions of these two new branches are based on the direction of the parent branch θ_{n-1} and on a certain user-defined branching angle θ_{br}. The two new branch directions are then defined as: $\theta_{nu} = \theta_{n-1} - \theta_{br}$ and $\theta_{nd} = \theta_{n-1} + \theta_{br}$. The angles corresponding to each branch are relative to the horizontal line, as shown in Figure 1(c). The recursive function is exited whenever: (i) a node lies outside the growth region, (ii) a node lies at a distance to another existing node that is lower than a prespecified proximity parameter, or (iii) when a point is found at the Pareto front. In the latter case, the obtained point is added to the solution set, which gets displayed when all recursive calls are exited. Hence, this strategy can increase the speed of the solution process as the objective space gets explored only once. An overview of the algorithm's structure is presented in Algorithm 1.

2.2 Software

MATLAB R2020b is used as the optimization platform and is run on a 64-bit Windows 10 system with an Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz processor and 16 GB of RAM installed. The built-in optimization toolbox



(a) (b) (c)

Fig. 1. An illustration of the branching strategy procedure. (a) Starting from the inital guess, the MOOP is solved for the anchor points. And the borders of the growth region are set. (b) Starting from an existing node, an optimization problem is solved. The solution process stops once an intermediate solution lies at a distance exceeding the prefixed branch length. (c) For any existing node, two branches are created at angles θ_{nu} and θ_{nd}.

Algorithm 1 A branching algorithm

Input: The Initial Guess, IG, and Minimal Proximity, MP.
Output: Pareto set, S, with a resolution which depends on MP.
Step 1: The solution set is initializaed, $S = \{\}$.
Step 2: The two SOOPs, $\min J_1$ and $\min J_2$, are solved, to determine the anchor points and the borders of the growth space.
Step 3: Starting from IG, an initial branch is constructed with an angle of $\theta_0 = (\theta_{J_1} + \theta_{J_2})/2$, with θ_{J_1} and θ_{J_2} are the angles of the lower border and the upper border respectively to the horizontal line.
Step 4: The solution process ends once an intermediate solution lies at a distance from the IG that is higher than BL. That solution is labeled N_1.
Step 5: The start of the recursive process, with the node N_1 as an input.
For the current node N_i, check the three stoppage conditions:
IF N_i does lie outside the growth region,
OR IF, N_i is at a distance less than MP from another node
EXIT
IF N_i lies at the Pareto front:
Add N_i to S and EXIT
ElSE: create two branches at angles, θ_{nd} and θ_{nu}, to obtain the next two nodes N_{i+1} and N_{i+2}.
Step 6: When all recursive calls are exited, produce solution set S.

of MATLAB is also used, specifically the `fmincon` function and the `ode15s` function. `fmincon` is a function that returns parameters that minimize an objective (SOOP) while satisfying the problem's constraints, here the selected algorithm is Sequential Quadratic Programming (SQP). `ode15s` is used to solve systems of stiff differential equations via numerical differentiation.

3. RESULTS AND DISCUSSION

The algorithm is illustrated through two case studies (De Buck et al. (2021)). Each case study involves finding the Pareto front for a bi-objective optimization problem.

3.1 Case Study 1: Numerical Bi-objective Problem

Formulation The first case study is relatively straightforward, as the two objective functions are equal to the decision variables and the inequality constraint limits the feasible space to a rectangle with rounded corners. It was introduced by C. Mattson (2004), and can be described mathematically as:

$$\begin{aligned} &\min_x (J_1, J_2) \\ \text{with} \quad & \\ &J_1 = x_1 \\ &J_2 = x_2 \\ \text{and} \quad & \\ &\left(\frac{x_1 - 10}{10}\right)^8 + \left(\frac{x_2 - 5}{5}\right)^8 - 1 \leq 0 \\ &x_1 \in [-10, 10] \\ &x_2 \in [-10, 10] \end{aligned} \tag{8}$$

The high-degree polynomial inequality constraint establishes a high-trade-off region of the Pareto front near the origin of the objective space. The Pareto front is long and flat for higher values of the objectives (De Buck et al., 2021).

Results The branching algorithm successfully sampled the Pareto front; results from two runs are shown in Figures 2 and 3, where two different branching strategies are compared. In the first run, a relatively short branch length of $BL = 0.2$ is used. This results in dense branching and more thorough exploration of the objective space. However, more branches end up at the non Pareto boundaries of the growth region, leading to a high number of nodes and longer calculation times. For the second run, a different strategy is tested where $BL = 0.5$ and the Branch Depth Multiplier (BDM) is set to be $BDM = 0.7$. The BDM is an auxiliary parameter which is equal to the ratio between two the lengths of two consecutive branches. It allows the branching scheme to begin with longer branches to quickly get closer to the Pareto front, before using shorter branches to explore the Pareto front

thoroughly. This has resulted in a decreased runtime and number of generated nodes, while getting a Pareto front with a resolution similar to the first run. Both runs were able to generate a higher density of points at the knee region of the problem, which is deemed to be more interesting to the decision maker as it is characterized by higher level of trade-offs than the long plateaus. The Minimal Proximity (MP) parameter prevents the growth from a node that lies in the neighborhood of an existing node. And thus, it saves further time as there is no need to continue the solution processes that leads to Pareto points that are close to an already existing one. This in turn eliminates the need for a posteriori filtering. An overview of the algorithm's parameter values and the performance parameter values are provided in Tables 1 and 2. It can be seen from Table 2 that using the second branching strategy resulted in a faster runtime and a lower number of nodes, despite aiming for a higher Pareto front resolution.

Parameter	Run 1	Run 2
Branch Length (BL)	0.2	0.5
Branch Depth Multiplier (BDM)	1	0.7
Branch Angle (BA)	14	14
Minimal Proximity (MP)	0.02	0.01
Initial Guess (IG)	(1.28,1.28)	(1.28,1.28)

Table 1. Algorithm's parameters values for the bi-objective numerical case study.

Parameter	Run 1	Run2
Recursive runtime (s)	14.9	9.8
Number of nodes	221	178
Number of Pareto points	43	52

Table 2. Performance parameters values for the bi-objective numerical case study.

3.2 Case Study 2: Williams-Otto Reactor

Formulation The branching algorithm is tested for an industrial problem using a mathematical model for the Williams-Otto fed-batch reactor (Williams and Otto (1960)). In this reactor, the following reactions take place:

$$\begin{aligned} A + B &\longrightarrow C \\ C + B &\longrightarrow P + E \\ P + C &\longrightarrow G \end{aligned} \tag{9}$$

Reactant A is present in the reactor and reactant B is fed continuously. The products P and E are formed during the exothermic reactions, as well as G, the side-product. A cooling jacket is required for the removal of heat generated through the exothermic reactions. This cooling jacket is controlled by the temperature of the cooling water (Hannemann and Marquardt (2010)). The two considered objectives are the maximal conversion of the main products P and E. During the process, there are some variables that need to be observed, being the path constraints on the inlet flow rate of reactant B, denoted as $F_{B,in}$, the reactor temperature T, the reactor volume V , and the scaled cooling water temperature T_W. The control variables of this process that can be manipulated are $F_{B,in}(t)$ and $T_W(t)$. The dynamic model consists of nine differential equations (Hannemann and Marquardt (2010), Williams and Otto (1960)).

$$\frac{dx_A}{dt} = \frac{x_A x_1}{1000V} - k_1\eta_1 x_A x_B \tag{10}$$

$$\frac{dx_B}{dt} = \frac{(1-x_B)x_1}{1000V} - k_1\eta_1 x_A x_B - k_2\eta_2 x_B x_C \tag{11}$$

$$\frac{dx_C}{dt} = \frac{x_C x_1}{1000V} + k_7\eta_1 x_A x_B - k_3\eta_2 x_B x_C - k_6\eta_3 x_P x_C \tag{12}$$

$$\frac{dx_P}{dt} = -\frac{x_P x_1}{1000V} + k_2\eta_2 x_B x_C - k_4\eta_3 x_P x_C \tag{13}$$

$$\frac{dx_E}{dt} = \frac{x_E x_1}{1000V} + k_3\eta_2 x_B x_C \tag{14}$$

$$\frac{dx_G}{dt} = \frac{x_G x_1}{1000V} + k_5\eta_3 x_P x_C \tag{15}$$

$$\frac{dT}{dt} = \frac{(T_F - T)x_1}{1000V} + k_8\eta_1 x_A x_B + k_9\eta_2 x_B x_C + k_1\eta_3 x_P x_C - h(T - 1000x_2) \tag{16}$$

$$\frac{dV}{dt} = \frac{x_1}{1000} \tag{17}$$

Not to confuse x_A and x_1, the former is a dimensionless weight fraction and the latter a decision variable. The initial conditions are: $x_0 = \{1, 0, 0, 0, 0, 0, 65, 2\}$. k_j, with $j \in 1, 2, ..., 10$ are the pre-exponential reaction constants, whose values along with the rest of the problem's parameters can be found in Williams and Otto (1960) and De Buck et al. (2021). Each reaction has its own temperature dependency given by Arrhenius terms η_m:

$$\eta_1 = \exp(\frac{-6666}{T + 273}) \tag{18}$$

$$\eta_2 = \exp(\frac{-8333}{T + 273}) \tag{19}$$

$$\eta_3 = \exp(\frac{-11111}{T + 273}) \tag{20}$$

This problem is controlled by two process variables, the feeding rate x_1 of the reactant B and the scaled temperature x_2 of the refrigerant in the cooling jacket. Dimensional constraints and safety precautions put constraints on the system:

$$60 \leq T(t) \leq 90 \tag{21}$$

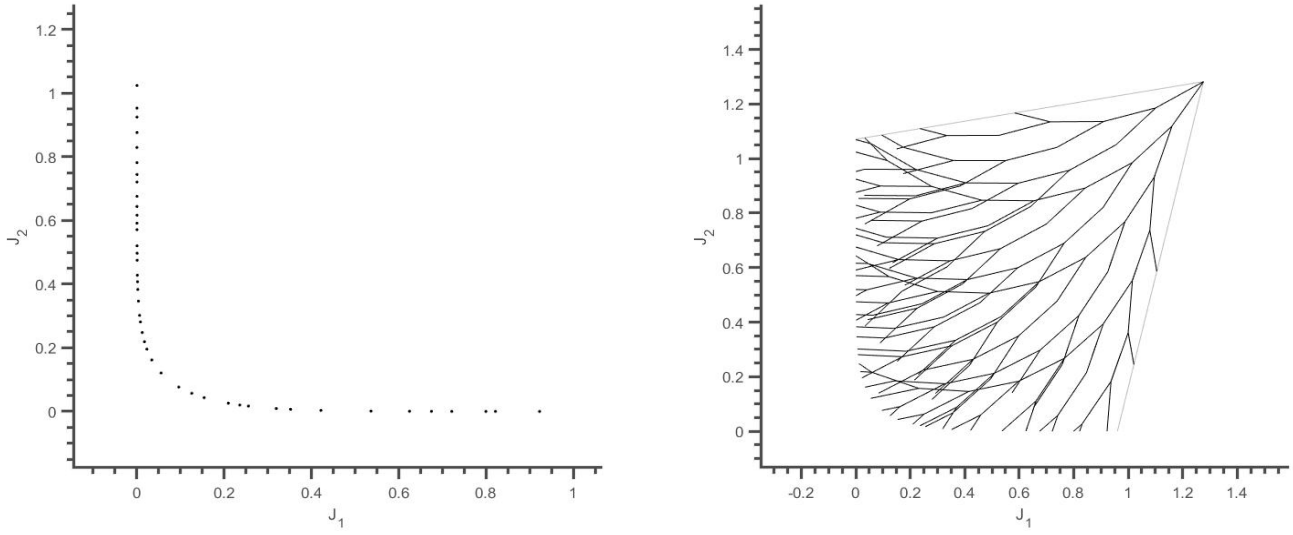
$$0 \leq x_1 \leq 5.7 \tag{22}$$

$$0.02 \leq x_2 \leq 0.1 \tag{23}$$

$$V(t_f) \leq 5 \tag{24}$$

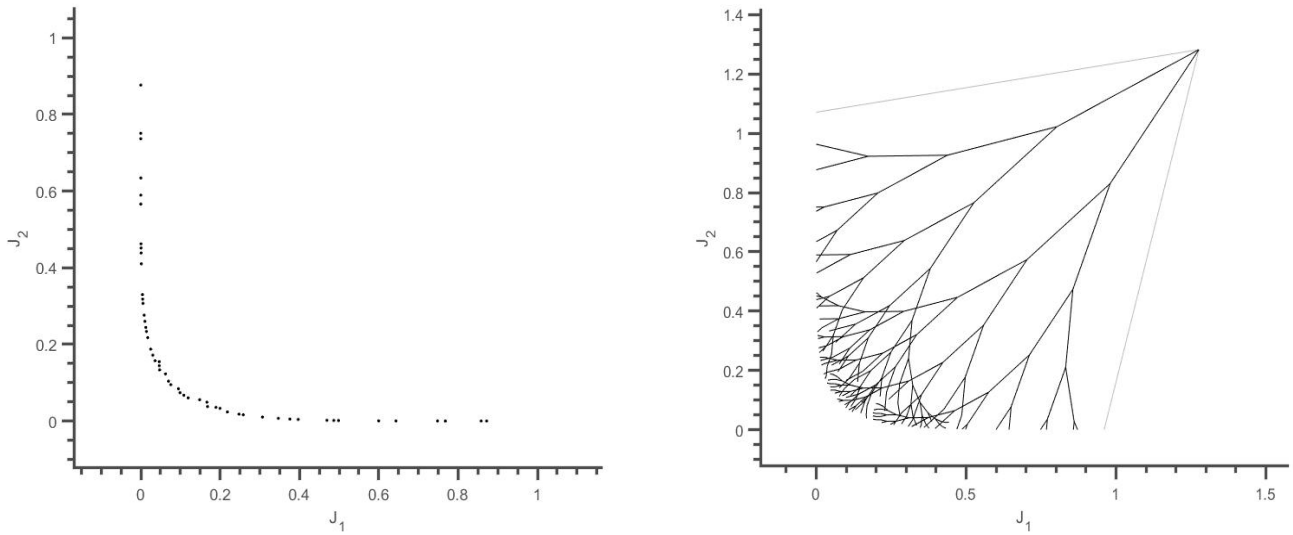
The derivatives are calculated using a MATLAB integrator, namely `ode15s` (De Buck et al. (2021)). The formulation of the objectives is:

$$\begin{aligned} &\min_x (J_1, J_2) \\ \text{with} \quad & \\ &J_1 = -x_P(t_f)V(t_f) \\ &J_2 = -x_E(t_f)V(t_f) \end{aligned} \tag{25}$$



(a) (b)

Fig. 2. Visualizations of the first run for the bi-objective numerical case study. While the algorithm was able to obtain a representation of the Pareto front, it is noted that a number of branches ends up growing laterally towards the borders of the growth space. (a) The obtained Pareto set. (b) A representation of the iterative solution process in the objective space, starting from the initial guess.



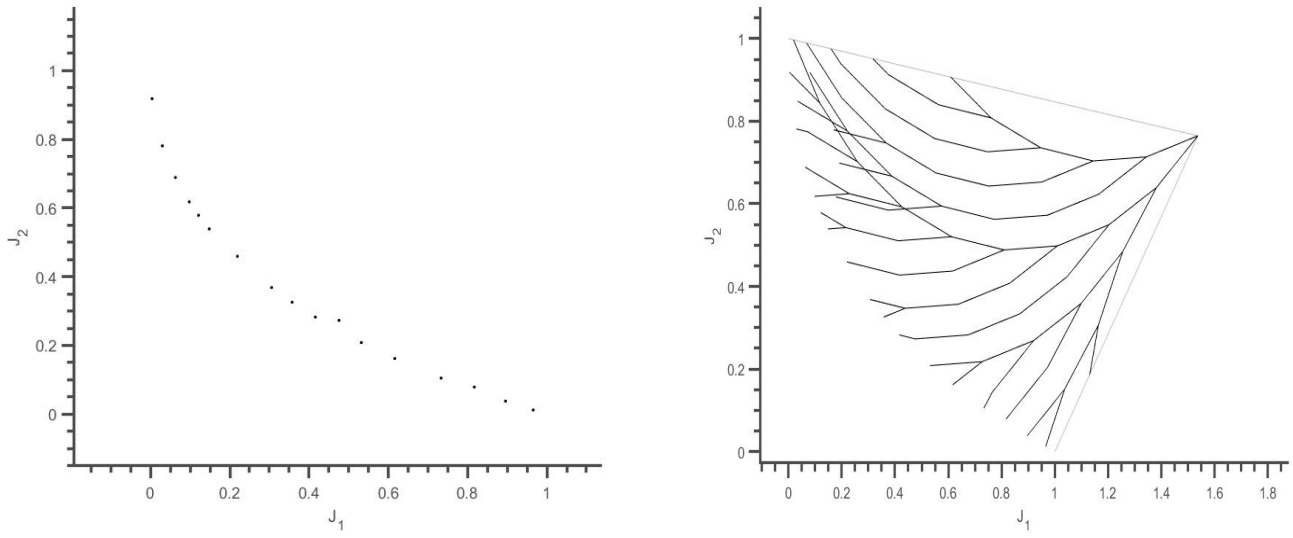
(a) (b)

Fig. 3. Visualizations of the second run for the bi-objective numerical case study. The usage of initially large branches reduces solution processes that end up at the boundaries of the growth space, whilst maintaining an accurate representation of the Pareto front. (a) The obtained Pareto set. (b) A representation of the iterative solution process in the objective space, starting from the initial guess.

Results The algorithm's configurations when applied to the Williams-Otto reactor problem are over-viewed in table 3, while the quantitative results of its performance are provided in table 4. As seen in Figure 4, the algorithm managed to obtain a representation of the Pareto front, albeit requiring a relatively long runtime compared to the previous case study, owing to the higher complexity of this problem. A main difference that can be observed compared to the last case study, is the point density of the obtained Pareto front, owing to choosing a higher value for MP, the branching structure is more sparse and the Pareto front is less dense. The usage of MP allows the user of the algorithm to adjust the resolution of the Pareto front based on their needs.

4. CONCLUSION

In this paper, a novel branching strategy is proposed for solving bi-objective optimization problems. The new algorithm explores the objective space using a branching



(a) (b)

Fig. 4. Visualization of the Pareto front as well as the solution process of the branching algorithm when applied to the Williams-Otto reactor case study. (a) The obtained Pareto set. (b) A representation of the iterative solution process in the objective space, starting from the initial guess.

Parameter	Value
Branch Length (BL)	0.2
Branch Depth Multiplier (BDM)	1
Branch Angle (BA)	12
Minimal Proximity (MP)	0.04
Initial Guess (IG)	(1.54, 0.76)

Table 3. Algorithm's parameters values for the William-Otto case study.

Parameter	Value
Recursive runtime (s)	2 370.3
Recursive runtime (min)	39.5
Number of nodes	78
Number of Pareto points	17

Table 4. Performance parameters values for the William-Otto case study.

strategy by solving a series of subproblems for a number of iterations, corresponding to a predefined distance in the objective space, while recursively constructing further branches to obtain a representation of the Pareto front. This has two advantages over current methods. First, it allows a systematic exploration for the objective space, which leads to reducing the number of iterations. Second, a proximity criterion ensures that branches that get closer to existing nodes are terminated, which allows controlling the resolution of the Pareto front *during* the iterative solution process, instead of filtering the obtained solutions afterwards as it is currently traditionally done, thus reducing the computational effort. The algorithm has been applied and illustrated on two case studies and managed to successfully obtain the Pareto front. Future work is needed to (i) test the algorithm on further case studies and (ii) to further reduce the number of parameters that the algorithm uses.

REFERENCES

C. Mattson, A. Mullur, A.M. (2004). Smart pareto filter: obtaining a minimal representation of multiobjective design space. *Eng. Optim.*, 36(6), 721–740.

De Buck, V., Nimmegeers, P., Hashem, I., Muñoz López, C.A., and Van Impe, J. (2021). Exploiting trade-off criteria to improve the efficiency of genetic multi-objective optimisation algorithms. *Frontiers in Chemical Engineering*, 3, 3. doi:10.3389/fceng.2021.582123.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley, Chichester, London, UK.

Hannemann, R. and Marquardt, W. (2010). Continuous and discrete composite adjoints for the hessian of the lagrangian in shooting algorithms for dynamic optimization. *SIAM journal on scientific computing*, 31(6), 4675–4695.

Logist, F., Houska, B., Diehl, M., and Van Impe, J. (2010). Fast pareto set generation for nonlinear optimal control problems with multiple objectives. *Structural and Multidisciplinary Optimization*, 42, 591–603.

Logist, F., Telen, D., Houska, B., Diehl, M., and Van Impe, J. (2013). Multi-objective optimal control of dynamic bioprocesses using acado toolkit. *Bioprocess and Biosystems Engineering*, 36, 151–164.

Marler, R. and Arora, J. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26, 369–395.

Mattson, C., Mullur, A., and Messac, A. (2004). Smart pareto filter: Obtaining a minimal representation of multiobjective design space. *Engineering Optimization*, 36(6), 721–740.

Williams, T.J. and Otto, R.E. (1960). A generalized chemical processing model for the investigation of computer control. *Transactions of the American Institute of Electrical Engineers. Part 1. Communication and electronics*, 79(5), 458–473.