A novel high speed multi-objective evolutionary optimisation algorithm [★]

Viviane De Buck [*] Ihab Hashem [*] Jan Van Impe [*]

[*] *BioTeC+, Chemical and Biochemical Process Technology and Control Department of Chemical Engineering, KU Leuven, Belgium (e-mail: viviane.debuck@kuleuven.be, ihab.hashem@kuleuven.be, jan.vanimpe@kuleuven.be).*

Abstract: Multi-objective optimisation problems (MOOPs) consider multiple objectives simultaneously. Solving these problems does not render one unique solution but instead a set of equally optimal solutions, i.e., the Pareto front. The goal of solving a MOOP is to accurately and efficiently approximate the Pareto front. The use of evolutionary optimisation algorithms is widespread in this discipline. During each iteration, parent solutions are combined and mutated to create new offspring solutions. Both populations are subsequently combined and sorted. Only the N fittest solutions of the combined set are selected as the parent solutions for the subsequent iteration. The fitness of a solution is defined by its convergence to the Pareto front and its contribution to the overall solution diversity. Widely used evolutionary algorithms, like NSGA-II (Deb et al., 2002), use non-dominated sorting to assess the convergence of solutions and the concept of crowding distance to ensure a high solution diversity. Both concepts, however, require that all N solutions of the population are compared with all other $(N-1)$ solutions for both aspects, and this for all M objectives. This results in a computational complexity of $O(MN^2)$. In this contribution, a novel evolutionary algorithm is presented, boasting a significantly lower computational complexity of $O(N\log(N))$. This is achieved by subdividing the feasible space into angular sections. Solutions are scored based on their distance from the current Utopia point and the overall crowdedness of their respective section. Sorting the population based on the attributed scores allows the selection of the N fittest solutions, without having to mutually compare them.

Keywords: Evolutionary optimisation algorithms, Algorithm development,

1. INTRODUCTION

The (bio-)chemical industry is one of the most competitive industries in the world resulting in tight profit margins. Additionally, due to strict regulations, especially in the pharmaceutical and biotechnological sector, the possibilities to increase profits or adapt the process are limited. It is therefore of the essence that processes are executed as optimally as possible. As optimality is a broad spectrum and the already mentioned regulations often constrain the process possibilities in certain aspects, optimisation problems are often conceptually and mathematically challenging problems to design and solve. Multi-objective optimisation (MOO) gives the process operator the opportunity to optimise his/her process with respect to multiple, and often conflicting, objectives. Commonly, optimisation objectives are of economical (e.g., profit), societal (e.g., required full-time equivalents), or environmental (e.g., energy use or production of greenhouse gasses (GHGs)) nature (Ozcan-Deniz and Zhu, 2017). As multiple objectives are considered simultaneously, the resulting multi-objective optimisation problem (MOOP) will not provide a single unique optimal solution. Instead, an infinite set of equivalent trade-off, or non-dominated, solutions is obtained. When the decision maker (DM) would switch from one non-dominated solution to another, the cost of certain objectives will decrease (i.e., become more optimal), while the cost of others will increase. This phenomena is called a trade-off. While it mostly possible to translate all M considered objectives into a common unit, e.g., money, and combine into a single objective, MOO has some distinct advantages over single-objective optimisation (SOO) (Marler and Arora, 2004).

One of the most important advantages of MOO over SOO is that a MOO allows the user to examine how the different objectives correlate to each other. This is aspect is especially interesting when one of the objectives is less sensitive to changes in the optimisation variables than another objective. The resulting Pareto front in these scenarios will display so-called plateaus, which are Pareto front regions in which one objective cost will change drastically while the cost for another objective stays more static. Decision makers are not prone to selecting working points that are located on a plateau of the Pareto front because it is possible to improve at least one objective without significantly demoting the other (Mattson et al., 2004; Hashem et al., 2017). In the case of an SOO, it is impossible to do this assessment as only one solution is provided and the correlations of the objectives remain unknown. An additional advantage of MOO over SOO is that the different objectives remain intact and can be expressed on different scales and units. Especially when regulations, or other rapidly changing process aspects, or a high number of objectives are considered, it is advisable to not translate them into one objective. Also note that some qualitative objectives that might be considered, like

★ This work was supported by KU Leuven [PFV/10/002] Center-of-Excellence Optimization in Engineering (OPTEC). V. De Buck is supported by FWO-SB Grant 1SC0920N. I. Hashem is supported by FWO Grant 1S54217N.

environmental or societal impact, are not easily translated into a common, quantitative unit (Coello et al., 2019).

MOOPs are generally mathematically challenging problems and are dealt with using algorithms especially designed for the purpose (Bhonsale et al., 2018). Two main algorithm categories can be distinguished: *(i)* Deterministic algorithms, and *(ii)* Stochastic algorithms (Logist et al., 2010). Deterministic algorithms transform the MOOP into a finite set of SOOPs using parameters. The generated SOOPs are subsequently iteratively solved using a SOO-algorithm. The unique solutions of the generated SOOPs are also solutions of the original MOOP. Deterministic algorithms are however prone to converge to local optima on the Pareto front, can have difficulties generating solutions located within non-convex Pareto front section, and can only generate *one* solution during each iteration. Stochastic algorithms on the other hand tackle the MOOP in its entirety. Unlike the deterministic algorithms, they are considered as *global* optimisation algorithm while they are less prone to converge to local optima and are able to generate solutions in the non-convex Pareto front sections. Additionally, they are capable of generating multiple Pareto-optimal solutions in a single run (Logist et al., 2010; Deb et al., 2002). The overall concepts of multi-objective optimisation, as well as the background of evolutionary algorithms, are discussed more into detail in Section 2

Evolutionary optimisation algorithms are a subcategory of the stochastic algorithms. Their overall functionality mimics the concept of *selection of the fittest* as seen in nature. A set of parent solutions is used to generate new offspring solutions with the use of crossovers, which are based on sexual reproduction, and mutations. Solutions are subsequently sorted and selected based on their fitness. The fitness of a solution is defined by its convergence to the Pareto front and its contribution to the solution diversity. One of the most commonly used evolutionary optimisation algorithms is NSGA-II, developed by Deb et al. (2002). For the fitness-assessment of the generated solutions, it uses fast non-dominated sorting and crowding distances. Both concepts, however, rely on the mutual comparison of each of the N generated solutions with all other $(N-1)$ solutions of the population, and this for all M objective functions. This results in a computational complexity of $O(MN^2)$ (Deb et al., 2002). Both non-dominated sorting and crowding distances are explained more into detail in Section 2.2.

A multitude of efforts have already been undertaken to reduce the computational complexity and/or time requirements of evolutionary algorithms. One approach to reduce the time requirements of evolutionary algorithms is to provide them with a problem-relevant stopping criterion, allowing to terminate the algorithm once solutions have converged. De Buck et al. (2019) presented, amongst others, a novel stopping criterion for an evolutionary algorithm based on NSGA-II. This resulted in a significant time gain compared to NSGA-II. However, the computational complexity of the proposed algorithm remains $O(MN^2)$. Reducing the computational complexity of evolutionary algorithms is mainly obtained by omitting the non-dominated sorting step. Usually, divide-and-conquer schemes are used for this purpose. The basic philosophy behind these schemes is that the computational complexity of the algorithm can be significantly decreased if mutual comparison can be replaced with comparing only a small subset of solutions with each other. Mishra et al. (2019), for instance, introduced a divide-and-conquer based non-dominated sorting scheme, resulting in an evolutionary algorithm with a best case computational complexity of $O(N \log N + MN)$.

This contribution presents a novel high-speed multi-objective optimisation algorithm with a computational complexity of $O(N \log N)$, which is the computational complexity of the Quick sort algorithm, as implemented in Matlab. The presented algorithm does not employ non-dominated sorting, nor does it use the concept of crowding distance to maintain a high solution diversity. Instead it uses an angular sorting scheme, which subdivides the feasible space into angular subsections. The fitness of solutions is subsequently assessed based on their distance to the Utopia point, i.e., the point containing all M individual minimisers of the considered objectives, and the crowdedness of their respective angular section. Previous efforts at subdividing, or decomposing, the feasible space of the MOOP in order to increase the performance of evolutionary algorithms (EA's) include, among others, the MOEA/D as proposed by Zhang and Li (2007).

The developed *Genetic Angular SorTing* $O(N \log N)$ *algorithm*, or the GASTON-algorithm for short, is discussed more into detail in Section 3. The NSGA-II, MOEA/D and GASTON-algorithm are applied to the CONSTR- and DO2DK-case studies (see Section 4). The obtained results are discussed in Section 5 and a conclusion is drafted in Section 6.

2. MULTI-OBJECTIVE OPTIMISATION

2.1 Mathematical definitions

Equation (1) presents the mathematical definition of a MOOP that will be used in this contribution (Das and Dennis, 1997; Logist et al., 2010):

$$\min_{\mathbf{x} \in C} \mathbf{F}(\mathbf{x}) = \{J_1(\mathbf{x}), \ldots, J_M(\mathbf{x})\} \tag{1}$$

The feasible space C is defined by (Das and Dennis, 1997; Logist et al., 2010):

$$C = \{\mathbf{x} : \mathbf{h}(\mathbf{x}) = 0,\ \mathbf{g}(\mathbf{x}) \leq 0,\ \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\} \tag{2}$$

with $\mathbf{h}(\mathbf{x}) = 0$ the (non-)linear equality constraints, $\mathbf{g}(\mathbf{x}) \leq 0$ the (non-)linear inequality constraints, and $\mathbf{a}$ and $\mathbf{b}$ the respective lower and upper boundaries of the variables $\mathbf{x} \in \mathbb{R}^n$.

2.2 Evolutionary algorithms

Evolutionary algorithms are powerful multi-objective optimisation algorithms that are widely used within the field. Their main advantages over other optimisation algorithms are their ability to generate multiple optimal solutions per iteration, their capability to generate optimal solutions in non-convex Pareto front regions, and the fact that they are less prone to converge to local optima. Additionally, while they do not require derivative information, they are extensively used for black-box multi-objective optimisation problems. Their main outline is based on Darwinian selection where only the N fittest solutions are selected from a combined solution set of parent solutions and offspring solutions. The initial phase of an evolutionary algorithm consists of generating a random set of parent solutions $\mathcal{P}_0$ within the feasible space which is defined by the (non-)linear constraints and the optimisation variables lower and upper boundaries (Deb et al., 2002).

Considering the t-th iteration, the parent set $\mathcal{P}_{t-1}$ ($|\mathcal{P}_{t-1}| = N$) is used to generate an offspring set $\mathcal{Q}_t$ ($|\mathcal{Q}_t| = N$) via crossovers and mutations. A crossover is the algorithmic equivalent of sexual reproduction and consists of a linear recombination of two parent solutions, rendering two new offspring solutions. A mutation on the other hand consists of randomly changing several optimisation variables of a parent solution, resulting in one new offspring solution. To increase the convergence speed of evolutionary algorithms, elitism is applied, which states that the best solutions of the previous iteration remain unchanged in the current one. This is obtained by merging the parent set $\mathcal{P}_{t-1}$ and the offspring set $\mathcal{Q}_t$ of the t-th iteration into the combined solution set $\mathcal{R}_t = \mathcal{P}_{t-1} \cup \mathcal{Q}_t$ ($|\mathcal{R}_t| = 2N$). Only the N fittest solutions of the combined solution set $\mathcal{R}_t$ are selected for the parent set $\mathcal{P}_t$ of the $(t+1)$-th iteration (Deb et al., 2002).

As already mentioned, the fitness of a solution is defined by its convergence to the Pareto front and its contribution to the solution diversity. NSGA-II, designed by Deb et al. (2002), employs a fast non-dominated sorting step for sorting the solutions of the combined solution population $\mathcal{R}_t$ based on their convergence to the Pareto front. The non-dominated sorting step appoints each solution to one and only one non-dominated front $\mathcal{F}_i$. All solutions that are part of the same non-dominated front $\mathcal{F}_i$ display the same degree of convergence to the Pareto front and cannot be considered better or worse than each other. These solutions do not dominate each other. However, they do dominate all solutions that are located in the higher non-dominated fronts $\mathcal{F}_j$ $(j > i)$, and are dominated by all solutions located in the lower non-dominated fronts $\mathcal{F}_k$ $(k < i)$. A solution $\mathbf{p}$ dominates $\mathbf{q}$ if and only if (Deb et al., 2002; Logist et al., 2010):

$$\mathbf{p} \prec \mathbf{q} \Leftrightarrow \mathbf{F}(\mathbf{p}) \leq \mathbf{F}(\mathbf{q}) \wedge \exists\, l : J_l(\mathbf{p}) < J_l(\mathbf{q}) \tag{3}$$

with $l \in \{1, \ldots, M\}$. Worst case scenario is that all N solutions are located in a different non-dominated front. In that scenario, for all N solutions, the objective cost of all $(N-1)$ solutions must be compared with the objective cost of the considered solution, and this for all M objectives, resulting in a computational complexity of $O(MN^2)$. The solutions located in the lowest non-dominated fronts are favoured for selection.

Subsequently, in order to maintain a good solution diversity, the crowding distance of all solutions is calculated and this for each non-dominated front. The crowding distance of a solution is the average length of the edges in each objective direction of the cuboid that has the neighbouring solutions of the considered solution as vertices. When a solution has a high crowding distance, it implies that its neighbouring solutions are located at a high distance, and thus that the solution in question is located in an under-explored part of the Pareto front. The solutions with the highest crowding distances are favoured for selection. Again, the crowding distance concept of the NSGA-II algorithm requires all N solutions to be mutually compared, and this for all objectives. However, the objective costs of the solution are now only compared to those of the solutions that are located within the same non-dominated front, resulting in a lower amount of required comparison, and thus a lower overall complexity than the non-dominated sorting step. The GASTON-algorithm presented in this contribution boasts a non-dominance free fitness assessment, resulting in a significantly lower computational complexity.

3. GASTON ALGORITHM CONCEPT

The main idea of the algorithm is to design an evolutionary system where Darwinian selection acts to select for solutions that get as close as possible to the Pareto front, while at the same time selecting against crowdedness. Also, this has to be done while avoiding computationally expensive pairwise comparison of solutions. The algorithm starts by creating an initial population of solutions $\mathcal{P}_0$ of size N. Then, as typically done in genetic algorithms, crossover and mutation mechanisms are used to generate an offspring population $\mathcal{Q}_0$ of size N, with the probability of a certain offspring to be a result of a crossover or mutation event as μ_c and μ_m respectively, $\mu_c + \mu_m = 1$. A combined population of size $2N$ is hence generated, denoted by $\mathcal{R}_1$. This population gets processed via three steps: normalisation, angular sorting and fitness calculation. The normalisation step aims to overcome the differences in scale between different objectives values. This is done via finding the minimum and maximum value for each objective J_i for the set of solutions in the population, denoted as a_i^{min} and a_i^{max} respectively. Afterwards, the normalised value of each objective for each solution, $\bar{J}_i(\mathbf{x})$, is calculated as follows:

$$\bar{J}_i(\mathbf{x}) = \frac{J_i(\mathbf{x}) - a_i^{min}}{a_i^{max} - a_i^{min}} \tag{4}$$

The second step is the angular sorting of solutions, illustrated in Fig. 1. First, the distance between each solution and the Utopia point, d, is calculated where $d = \sqrt{\bar{J}_1(\mathbf{x})^2 + \bar{J}_2(\mathbf{x})^2}$ for a bi-objective problem. Then, the angle at which every solution lies with respect to the origin can be subsequently calculated as $\theta = \sin^{-1}\dfrac{\bar{J}_2(\mathbf{x})}{d}$. After each solution gets assigned a θ, a number of reference r lines, entered a priori by the user, is uniformly drawn in the space to span all solutions. Subsequently, each solution gets assigned to the reference line which has the closest angle to it, such that for each reference line, a set of solutions $\mathcal{S}_j$ gets assigned to it, with $j \in \{1, 2, ..., R\}$. The third step in the algorithm is calculating the fitness of the points belonging to each reference line in the solution space. For a reference line set, S_j of size R, the fitness of any solution k, Π_k, is calculated as follows:

$$\Pi_k = \frac{\dfrac{1}{d_k}}{\dfrac{1}{d_1} + \dfrac{1}{d_2} + ... + \dfrac{1}{d_R}} \tag{5}$$

Using this fitness equation ensures the realisation of two critical conditions. First, solutions closest to the origin will get assigned highest fitness. Secondly, the sum of fitness of solutions belonging to any reference line will be equal to 1. Hence, solutions belonging to less crowded reference lines will enjoy a competitive advantage over solutions in crowded sets. And selection will tend to spread solutions across reference lines. Finally, to ensure that solutions quickly spread to cover the whole Pareto front, the two solutions with lowest and highest θ will get always selected, whatever their position with respect to other points on their reference line. Hence, even if the initial population lay at close angles with respect to each other in the solution space, subsequent generations will angularly spread quickly. Finally, the population is sorted according to fitness and the best N individuals are selected to generate a new parent population. The flowchart of the algorithm is shown in Fig. 2.

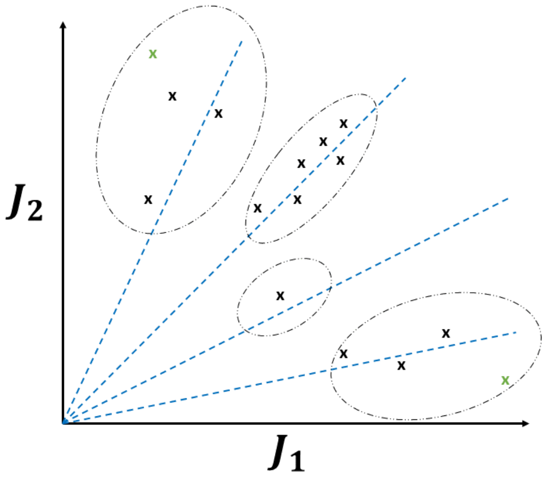

Fig. 1. Illustration of the angular sorting used in the algorithm. Each reference line is assigned a group of points, where the sum of the fitnesses of the points belonging to the same group is equal to 1. Also, the two points with lowest and highest angle get assigned maximum fitness.

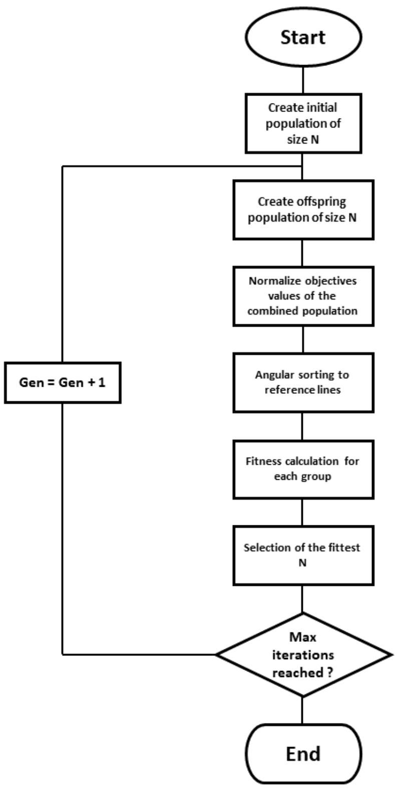
Start
Create initial population of size N
Create offspring population of size N
Normalize objectives values of the combined population
Angular sorting to reference lines
Fitness calculation for each group
Selection of the fittest N
Max iterations reached ?
End
Gen = Gen + 1

Fig. 2. Flowchart of the GASTON algorithm.

4. CASE STUDY

The multi-objective optimisation case study considered in this paper to benchmark the algorithm is the CONSTR-problem, first presented by Deb et al. (2002). The mathematical formulation of this problem is as follows:

$$\min_{\mathbf{z}}(J_1, J_2) \tag{6}$$

with the two objectives J_1 and J_2 defined as follows:

$$J_1(\mathbf{z}) = z_1 \tag{7}$$
$$J_2(\mathbf{z}) = (1 + z_2)/z_1 \tag{8}$$

under the following constraints:

$$g_1(\mathbf{z}) = z_2 + 9z_1 \geq 6 \tag{9}$$
$$g_2(\mathbf{z}) = -z_2 + 9z_1 \geq 1 \tag{10}$$

where

$$z_1 \in [0.1, 1.0] \tag{11}$$
$$z_2 \in [0, 5] \tag{12}$$

The CONSTR-problem is a bi-objective optimisation problem is typically used to test the performance of multi-objective optimisation algorithms. NSGA-II algorithm has been applied to it before and successfully provided a complete representation of the Pareto front in $O(N^2)$ time. Hence, this case study is ideal for an initial comparison of the algorithm provided in this work and an established algorithm in literature. Additionally, the Pareto front of this problem is characterised by a a steep region, ending up with a sharp knee and a flat segment. This abrupt changes in the geometry of the Pareto front could pose a difficulty for optimisation algorithms to capture them. Also, the flat segment is expected to pose a potential challenge for the GASTON algorithm since points lying in this segment will have similar angles to each other. For these reasons, the CONSTR-Problem is a rich case study to test the novel algorithm.

The Pareto front of the DO2DK-problem has an overall similar shape to that of the CONSTR-problem, as it also has a flat segment. Additionally, it has both convex and concave Pareto fronts areas, which could pose additional difficulties regarding the convergence of the solutions. The DO2DK-problem is mathematically defined as follows (Branke et al., 2004):

$$\min_{\mathbf{z}}(J_1, J_2) \tag{13}$$

with

$$J_1(\mathbf{z}) = g(\mathbf{z})r(z_1)\sin\left(\pi\frac{z_1}{2^{s+1}} + \left(1 + \frac{2^s - 1}{2^{s+2}}\pi\right) + 1\right) \tag{14}$$

$$J_2(\mathbf{z}) = g(\mathbf{z})r(z_1)\left(\cos\left(\pi\frac{z_1}{2} + \pi\right) + 1\right) \tag{15}$$

and

$$g(\mathbf{z}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} z_i \tag{16}$$

$$r(z_1) = 5 + 10(z_1 - 0.5)^2 + \frac{1}{k}\cos(2k\pi z_1)2^{s/2} \tag{17}$$

$$z_i \in [0, 1],\ i = 1, 2, \ldots, n \tag{18}$$

The skewness of the Pareto front can be adapted using the s-variable, whereas the number of so-called Pareto front knees can be altered using the k-variable. n represents the number of decision variables. The DO2DK-parameters are set as follows: $s = 1.00$, $n = 300$, $k = 4$.

5. RESULTS AND DISCUSSION

The NSGA-II algorithm and GASTON-algorithm are both applied to the CONSTR and DO2DK-case studies, as presented in Section 4. The general algorithm parameters, like the population size, the crossover and mutation probabilities, and the maximum number of iterations, are the same for both algorithms. For the GASTON-algorithm, in order to assess the influence of the number of reference lines on the quality of the solutions, for different algorithm set-ups are tested. The numerical values of the algorithm parameters are presented in Table 1. Note that the MOEA/D-algorithm does not employ a mutation step for solution reproduction.

Table 1. Values of the parameters of the NSGA-II and GASTON-algorithm.

NSGA-II, MOEA/D & GASTON	
Population size	250
Crossover probability	0.9
Maximum iterations	500
NSGA-II & GASTON	
Mutation probability	0.1
GASTON	
Reference lines	18, 36, 72, 144

The Pareto fronts of the CONSTR-case study generated by the GASTON-algorithm are presented in Fig. 3 (using 18, 36, and 72 reference lines), and Fig. 4 (using 144 reference lines). Matlab R2018b is used as the optimisation platform and is run on a 64-bit Windows 10 system with an Intel Core i5-8500 CPU @ 3.00 GHz processor and 16 GB of RAM installed.

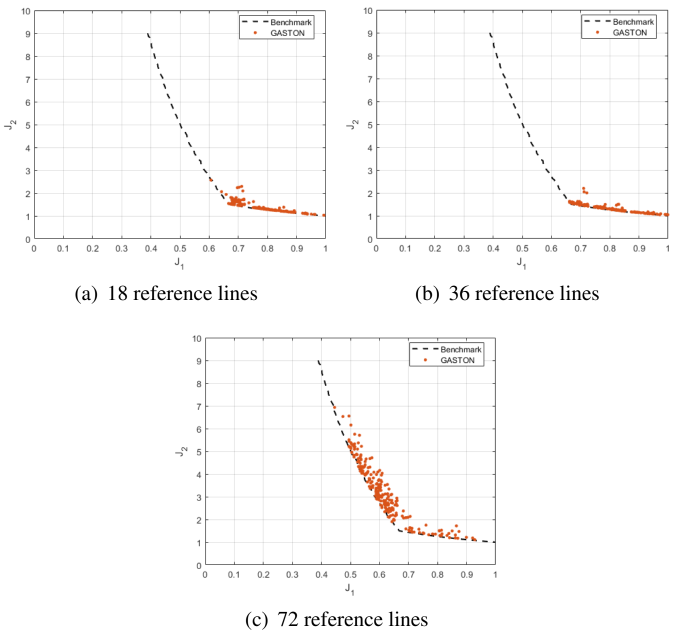

Fig. 3. Pareto fronts of the CONSTR-case study generated by the GASTON-algorithm.

Several observations can be drafted based on a visual analysis of the generated Pareto fronts: *(i)* Increasing the number of reference lines increases the diversity of the generated solutions, *(ii)* Not all generated solutions are Pareto-optimal. The first observation is most likely correlated to the dynamic nature of the scoring mechanism and the shape of the Pareto front. While the anchor points of the population can change with each iteration, so will the position of the Utopia point. Solutions that are located closer to the Utopia inevitably will get a higher score to some extent, leading to an increased selections of these solutions. In order to circumvent this *huddling* phenomenon, the highest scoring solutions of the first and last reference lines were always selected. Still, however, during the optimisation process, fluctuating huddling and scattering phenomena were

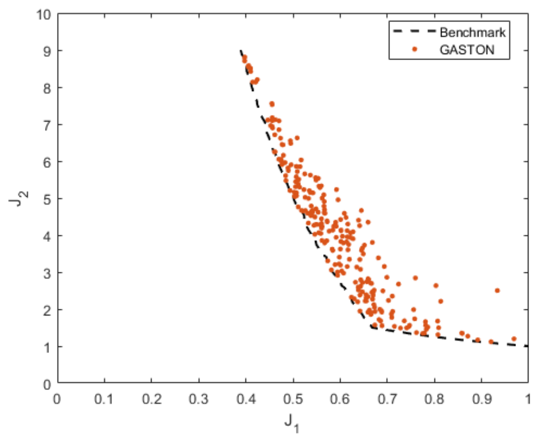

Fig. 4. Pareto front of the CONSTR-case study generated by the GASTON-algorithm (144 reference lines).

observed. To bypass this, a first future improvement of the GASTON-algorithm could be to store during the optimisation process the best anchor points that are generated thus far and use these during the normalisation and clustering process. This would only allow the actual non-normalised scope of reference lines to increase, forcing the solutions to discover the entire Pareto front and not to huddle together. The fluctuating huddling phenomenon was less pronounced when the number of reference lines used increased. This is presumably related to the fact that with a decreasing number of reference lines, but a fixed population size, all angular sections become equally crowded, and thus decreasing the effect of crowdedness during the selection procedure.

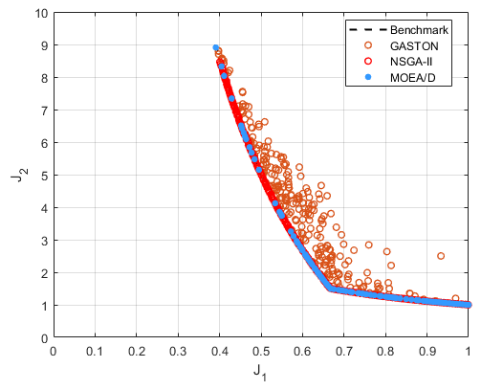

Fig. 5. Pareto front of the CONSTR-case study generated by the NSGA-II, MOEA/D, and GASTON-algorithm.

When comparing the Pareto front of the CONSTR-case study generated by the GASTON-algorithm (see Fig. 4) and the NSGA-II and MOEA/D algorithms (see Fig. 5), it is notable that not all solutions generated by the GASTON-algorithm are non-dominated, whereas all 250 solutions generated by the NSGA-II and MOEA/D algorithms have converged to the Pareto front after 500 iterations. Nonetheless, the overall shape of the Pareto front can be distinguished based on the solutions generated by the GASTON-algorithm. If desired, the algorithm can still easily be equipped with a non-dominated sorting step during the last iteration in order to select and display only the non-dominated solutions. Regardless, the GASTON-algorithm is perfectly suitable to provide the DM with a accurate approximation of the Pareto front in an extremely limited amount of time. The NSGA-II algorithm required 1379.6 s to generate the Pareto front, the MOEA/D algorithm required 934.0 s, whereas the GASTON-algorithm only required 24.7 s.

The Pareto front of the DO2DK-problem generated by all three algorithms, is represented in Fig. 6.

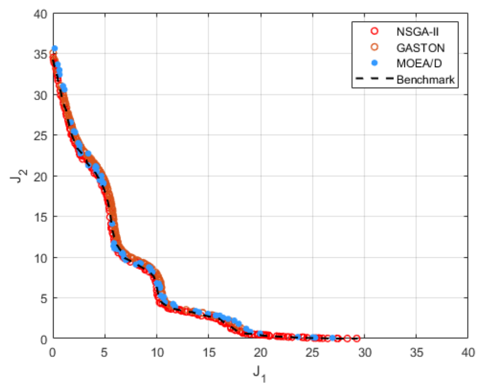

Fig. 6. Pareto front of the DO2DK-case study generated by the NSGA-II, MOEA/D, and GASTON-algorithm.

Contrarily to the CONSTR-case study, the solutions generated by the GASTON-algorithm have fully converged to the DO2DK Pareto front, just like the solutions generated by the NSGA-II and MOEA/D-algorithms. The GASTON-algorithm does not fully explore the flat region of the Pareto front. Note, however, that this flat region of the Pareto front represents a so-called low trade-off area. Solutions that are located in a low trade-off are less likely to be eventually picked by the decision maker. The GASTON-algorithm required 35.6 s to generate its solutions, whereas the NSGA-II algorithm required 1400.9 s and the MOEA/D algorithm required 940.0 s.

6. CONCLUSION

The existence of multiple conflicting objectives is a common feature in optimal control problems. The theoretical solution of such problems is a set of solutions in which no objective can be improved with out worsening at least another objective, called the Pareto front. Genetic algorithms aim to capitalise on Darwinian selection in order to reach a solution set that approximates the Pareto front. A natural design concept for such algorithms is to check for dominance; stochastically produce points in the solution space, assign highest fitness to the solutions that are least dominated by other points in the solution space. However, such strategy requires piece-wise comparison between all points in the population and thus is computationally expensive. In this paper, a novel algorithm is introduced which depends on designing a fitness function such that points are rewarded the closer they get to the Pareto front and punished for being close to each other in the solution space. This way, the candidate points population naturally evolves to provide a representation of the Pareto front with out the need for expensive dominance check operations. The algorithm is successfully applied to benchmark case studies and showed a significant reduction in computational time compared to established algorithms in literature. Two noted drawbacks are that the final representation produced by GASTON will still contain non-dominated points and that the algorithm is not always capable of fully exploring the Pareto front. However, this can be easily circumvented by coupling the algorithm with a dominance filter and by updating the dynamic normalisation step using the MOOP's anchor points. In future work, the aim is to generalise GASTON procedure for solving high dimensional problems, as well as tuning the selection procedure to produce a more smooth final result.

ACKNOWLEDGEMENTS

This work was supported by KU Leuven [PFV/10/002] Center-of-Excellence Optimization in Engineering (OPTEC). V. De Buck is supported by FWO-SB Grant 1SC0920N. I. Hashem is supported by FWO Grant 1S54217N.

REFERENCES

Bhonsale, S., Telen, D., Vercammen, D., Vallerio, M., Hufkens, J., Nimmegeers, P., Logist, F., and Van Impe, J. (2018). Pomodoro: A novel toolkit for dynamic (multiobjective) optimization, and model based control and estimation. *IFAC-PapersOnLine*, 51(2), 719–724.

Branke, J., Deb, K., Dierolf, H., and Osswald, M. (2004). Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature* - PPSN-VIII, 722–731.

Coello, C.A.C., Brambila, S.G., Gamboa, J.F., Tapia, M.G.C., and Gómez, R.H. (2019). Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, 1–16.

Das, I. and Dennis, J. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multi-criteria optimization problems. *Structural Optimization*, 14.

De Buck, V., Nimmegeers, P., Hashem, I., Muñoz López, C., and Van Impe, J. (2019). Improving evolutionary algorithms for multi-objective optimisation. Unpublished.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2), 182–197.

Hashem, I., Telen, D., Nimmegeers, P., Logist, F., and Van Impe, J. (2017). A novel algorithm for fast representation of a pareto front with adaptive resolution: Application to multi-objective optimization of a chemical reactor. *Computers & Chemical Engineering*, 106, 544–558.

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.

Marler, R.T. and Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6), 369–395.

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

Mishra, S., Saha, S., Mondal, S., and Coello, C.A.C. (2019). A divide-and-conquer based efficient non-dominated sorting approach. *Swarm and Evolutionary Computation*, 44, 748 – 773.

Ozcan-Deniz, G. and Zhu, Y. (2017). Multi-objective optimization of greenhouse gas emissions in highway construction projects. *Sustainable cities and society*, 28, 162–171.

Zhang, Q. and Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6), 712–731.