Parallel Genetic Algorithm Interface II: A novel computational tool for accelerated simulation-based optimization

Patrik Furda
patrik.furda@stuba.sk

Slovak University of Technology in Bratislava

Miroslav Variny
Slovak University of Technology in Bratislava

Juraj Myšiak
Slovak University of Technology in Bratislava

Research Article

Keywords: accelerated computing, Aspen Plus, genetic algorithm, simulation-based optimization, software linking

Posted Date: February 7th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3921060/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Chemical Engineering Research and Design on May 1st, 2024. See the published version at https://doi.org/10.1016/j.cherd.2024.05.030.
Parallel Genetic Algorithm Interface II: A novel computational tool for accelerated simulation-based optimization

Patrik Furda*, Miroslav Variny, Juraj Myšiak

Department of Chemical and Biochemical Engineering, Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, Radlinského 9, 812 37 Bratislava, Slovakia
*Corresponding author. E-mail address: patrik.furda@stuba.sk

Abstract
The ever increasing power of computational tools encouraged by the general need for development of more sustainable technologies fuels the interest in modern optimization approaches. While simulation-based optimization has been receiving considerable attention in the past decades, it still struggles to overcome some challenges, namely excessive computation time. This study proposes a novel optimization interface, the Parallel Genetic Algorithm Interface II (PAGAN-II), which utilizes parallelization of flowsheet simulations to drastically reduce the optimization time without the need to use clustered CPUs and/or modified optimization algorithms. Results of a detailed performance study showed up to 2100% increase in computation rate when optimizing demanding process flowsheets; and approximately 300% increase when optimizing simple ones. Capabilities of the proposed interface were demonstrated by optimization of a 5 MTPA C3MR LNG technology processing 12 different feedstocks, where a 15–30% decrease in the specific energy consumption was achieved. At the same time, the algorithm increased the optimization speed 13-fold compared to the traditional approach. This translates into a reduction of optimization time from 69 days of non-stop computation to approximately 7 days.

Keywords: accelerated computing, Aspen Plus, genetic algorithm, simulation-based optimization, software linking

Nomenclature

Abbreviations
ANFIS artificial neuro-fuzzy inference system
ANN artificial neural network
C3MR propane-precooled mixed-refrigerant cycle
CPU central processing unit
GA genetic algorithm
GUI graphical user interface
LNG liquefied natural gas
MR mixed refrigerant
NG natural gas
NSGA-II non-dominated sorting genetic algorithm
PAGAN Parallel Genetic Algorithm Interface
RAM random access memory
VBA Visual Basic for Applications

Symbols
FUI fraction of unconverged individuals
g number of a generation
G number of individuals
h penalty factor
\( \dot{m} \) mass flow rate, kg s\(^{-1}\)
N number of parallel simulations
Introduction

Simulation-based optimization of industrial processes has experienced an interesting surge in the last two decades, encouraged by the ever increasing power of simulation software and development of modern optimization methods. While simulation software such as Aspen Plus, Aspen HYSYS, gPROMS, CHEMCAD, etc. usually provide the simulation environment, modern stochastic optimization methods such as genetic algorithm, particle swarm optimization, ant colony optimization, simulated annealing, etc. and their variations serve as the basis for optimization. By linking the simulation software to these optimization tools, a robust multi-parametric and single- or multi-objective optimization of complex nonlinear processes can be performed. However, this methodology comes with a series of shortcomings which are to be discussed and resolved in this article.

1.1 Genetic algorithms

Genetic algorithm (GA) and its multi-objective version, the nondominated sorting genetic algorithm (NSGA-II), are the most studied and generally most widely applied optimization algorithms in engineering practice. The GA is a global optimization method derived from evolution and natural selection [1]. It is characterized by binary representation of individual solutions, problem-independent crossover and mutation operators, and a proportional selection rule [2]. Due to its evolutionary nature, the GA can adapt to virtually any problem as it does not have many mathematical requirements and it can solve any objective function with any kind of constraints [3]. The GA is robust and does not break easily on changed input or in the presence of reasonable noise. It is good at taking large, and/or multi-modal, and/or n-dimensional spaces, navigating them, and searching for optimal combinations of values [4]. It has been proven by many studies that genetic algorithm is more efficient and more robust in localizing optimal solutions and reducing computational effort than other conventional methods [3]. Apart from the operational advantages, a great advantage of GA/NSGA-II algorithms is the fact that they have been already incorporated in the toolboxes of programming languages such as Matlab and are, therefore, ready-to-use algorithms for any more or less experienced user. Based on this information, the paper further focuses on the use of GA/NSGA-II.

1.2 Simulation-based optimization

Simulation-based optimization is of growing interest in industrial practice as explicitly shown by numerous published articles. However, capable optimization algorithms are not yet included in the mostly used commercial flowsheet simulators [5]. Among the wide range of commercial simulators, Aspen Plus is the...
leading simulation software. While it is more complex, slower, and less user-friendly when compared to, e.g., Aspen HYSYS or CHEMCAD, it is much more robust and it is capable of solving a vast range of simulation problems, from thermochemical [6, 7] to biochemical [8] applications. Hence, Aspen Plus is the simulation software of choice in our application as well.

However, several publications have reported problems regarding simulation-based optimization using GA/NSGA-II and Aspen Plus. Firstly, linking Aspen Plus to GA/NSGA-II coded in a programming language such as Matlab have been an issue for several authors. Alabdulkarem et al. [9] were among the first to try to connect these two interfaces but were unsuccessful and used Aspen HYSYS instead. Sharma et al. [10] developed a program which used MS Excel spreadsheet as an input form and connected Aspen Plus via ActiveX Automation Server to the Excel VBA-coded NSGA-II. This interface was later adapted by various authors [11-15]. Later on, some authors [16, 17] used a rather complicated approach where they used Matlab-coded NSGA-II, but the software link was done via Excel VBA. Nonetheless, this issue was partially resolved by Abril [18] who published a freeware with instructions on Aspen Plus and Matlab linking and by Furda et al. [19] who presented a detailed methodology for the software linking.

The second issue is that, because of the evolutionary nature of the algorithm, the respective simulation needs to be run several times during optimization. Moreover, simulators such as Aspen Plus or CHEMCAD do not use multi-core and/or multi-CPU computer systems efficiently [5] which results in an unreasonably high computation time. Yang et al. [20] stated that the interaction between Aspen Plus and MATLAB is time-consuming, and Aspen Plus runs the calculations slowly. It was reported by Bravo-Bravo et al. [21] that 95% of the total time of the optimization procedure is employed in performing the simulations. Pandit and Jana [22] optimized 5 objective functions using up to 100 individuals and 300 generations and repeating the process up to 12 times resulting in 240,000 simulation runs. The authors do not comment on the optimization time. Nonetheless, even with 1 second per simulation, 240,000 runs translate into almost 3 days of computation time. Lee et al. [23] used 200 individuals over 50 generations and their computation time reached 32 hours. Guzmán Martínez et al. [17] stated that the calculation time was 8 hours on average per optimization run, whereas they used only 40 individuals and 8 generations. Darkwah et al. [8] used 1000 individuals and stated that 8 generations took 52.1 hours. However, their system was complicated and required solving ordinary differential equations coded in Matlab.

Various authors have battled this issue differently. Some authors [24-27] used only a very limited number of individuals and/or generations. However, it was proven by Furda et al. [28] that using such limited algebra can yield sub-optimal results, namely in multi-objective studies. Some authors proposed enhancing the algorithm to work more efficiently. De Buck [29] proposed a novel trade-off and termination criterion for NSGA-II and further enhanced the NSGA-II algorithm in the following publication [30]. Similar work was done by Rangaiah et al. [31]. Other authors [23, 27, 32] exploited the capabilities of server CPUs to speed up the calculations. To resolve the problems with computation time, Ali et al. [33] proposed using surrogate modeling. It was used by Sabbagh et al. [34], where they used three-variable and later four-variable [35] surrogate models; and by Enayatizadeh et al. [36] who used surrogate modeling and subsequent optimization using NSGA-II. However, such models use a limited number of parameters, and the process of surrogate model creation is complex and prone to errors. Lastly, several authors [6, 7, 21, 37-39] used artificial neural networks (ANN) to obtain a function representing the system’s behavior and used GA/NSGA-II to optimize the respective objective function. Okoji et al. [40] used artificial neuro-fuzzy inference system (ANFIS) to speed up the rate of obtaining the simulation results and subsequent optimization by the GA. Such an approach, however, requires building the ANN and training it.

A more sophisticated approach aimed at solving all beforementioned problems was proposed by Ernst et al. [5] who developed Advanced Process Optimizer (Adv:PO) which is a universal multi-objective optimization tool. This interface allows for linking simulation software with optimization algorithms and has already been proven useful, e.g., in the work of Johannsen et al. [41]. Although the authors state that this interface should be applicable with any optimization algorithm, the publication is focused solely on a specially tailored genetic algorithm. This algorithm encompasses an elite and diversity-preserving strategy [42] which promises more reliable results and faster convergence, and has an implemented parallelization concept for more efficient usage of the computation capacity. However, no information is given on how this
parallelization concept works. No comments are given either on the computation time and it remains unknown how to obtain this tool.

1.3 **PAGAN: Parallel Genetic Algorithm Interface**

In our previous work, we presented a novel approach to the issue: Parallel Genetic Algorithm Interface (PAGAN) [28]. The PAGAN algorithm, coded in Matlab programming language, uses ActiveX Automation Server which is a standard way for an external application to interact with Aspen Plus [43]. However, contrary to the traditional approach, the PAGAN algorithm uses:

1. vectorization of the fitness function, i.e., calling of the fitness function on the entire population at once [44],
2. asynchronous running of the Aspen plus simulations, i.e., exploitation of the fact that any number of Aspen Plus simulations can run simultaneously (provided that they have a different name) and the algorithm (Matlab) does not wait for a running simulation to finish and continues to the next step.

To accelerate the calculation, the population is divided into predefined number of groups, N. Next, N Aspen Plus simulations are initiated and each group of individuals is assigned to one simulation. Thus, a population of G individuals is simultaneously evaluated by N Aspen plus simulations as described in Fig. 1. This way, the number of calculation cycles per generation is reduced N-fold which drastically reduces the computation time.

The PAGAN algorithm was compiled into a user-friendly graphical user interface (GUI) and provided as a freeware [28]. Because of the simplistic nature of the interface, it can be used by virtually any user with minimal prior experience in optimization using GA/NSGA-II. It was successfully applied for optimization of a reactive distillation column design by Šulgan et al. [45].

![Fig. 1. Comparison of traditional approach and PAGAN algorithm [28].](image)

1.4 **Contribution of this work**

As was mentioned above, simulation-based optimization suffers from the need of a great number of repetitive simulations and the fact that software linking may prove challenging. To make the simulation-based optimization fast and available for anyone in the academia and/or industry, a novel optimization interface, the Parallel Genetic Algorithm Interface II (PAGAN-II), was developed. PAGAN-II is built on the foundations of the original PAGAN software but employs an improved optimization engine. This engine allows the algorithm to run the parallel simulations more effectively compared to PAGAN. As a result, PAGAN-II is expected to be faster and more stable compared to the previous version. Finally, thanks to its simple GUI, it is expected to be particularly useful for less experienced users.

2 **Methodology**

2.1 **PAGAN-II interface description**
PAGAN-II is a Matlab-coded algorithm and interface which uses the Aspen Plus ActiveX Automation Server to interact with Aspen Plus simulations and GA/NSGA-II algorithms for optimization. A simplified layout of the algorithm is displayed in Fig. 2. To accelerate the optimization, PAGAN-II creates and a predefined number of copies of the desired simulation (simulation engines). During the optimization, the entire population is sent to the optimization engine at once. The optimization engine assigns each simulation engine an individual, and the simulation engines start evaluating these individuals simultaneously. The optimization engine then continuously scans all simulation engines for their activity. Once a simulation engine becomes idle, i.e., the simulation run is completed, results of the simulation are collected, and another individual is assigned to the simulation engine. This way, a true parallelization is achieved without the need of using clustered computers or a server, and the optimization time is greatly reduced. Furthermore, it is generally known that continuous running of a software gradually increases the respective load on the RAM memory and overwhelms the processors’ cache memory. This phenomenon causes a decrease in computational speed. To battle this issue, PAGAN-II periodically reinitializes the simulation engines to diminish the effect of gradual optimization slow down.

The actual script of the algorithm of the optimization engine is provided in the Appendix. The PAGAN-II optimization interface is provided in the Supplementary material.

![Fig. 2. PAGAN-II optimization engine.](image_url)

### 2.2 Advantages of the PAGAN-II algorithm

Obviously, individual simulation runs can take different time, mainly because some simulation runs do not converge well or do not converge at all. Because the original PAGAN works in cycles, the slowest-working simulation engine determines the optimization speed. However, because PAGAN-II employs a series of independent simulation engines, the problem is eliminated.

To put things into perspective, one can imagine the difference in performance between the standard approach, PAGAN and PAGAN-II with the following example (Fig. 3). Three groups of students are given six assignments each of which take different time to finish (for demonstration suppose the simulation times are 3, 3, 12, 3, 6, and 3 seconds, respectively). The first group consists of only one student, because only
one simulation engine is employed in the standard approach. The other two groups (PAGAN and PAGAN-
II) consist of three students (i.e., three simulation engines). The differences in the process of optimization
can be then described as follows:

- Using the standard approach, the algorithm solves each of the assignments one after another and the
total time is approximately equal to the sum of times needed to finish the individual assignments.
- PAGAN takes on the first three assignments and solves them simultaneously. However, the
algorithm needs to wait for all three assignments to be done before proceeding to the other group of
assignments.
- In PAGAN-II, the individual engines work independently and once an engine is not busy it
immediately takes on another assignment.
Fig. 3. Demonstration of optimization process for standard approach, PAGAN and PAGAN-II. Legend: black – start of a simulation, green – end of a simulation.
To assess the performance of the newly developed optimization interface, a series of optimization studies were performed. First, the interface was used for optimization of natural gas liquefaction process to address the possibilities brought by this invention, and second, a detailed performance study was realized to better understand the algorithm’s behavior.

3.1 C3MR LNG process

Since global energy requirements are expected to rise by 56% between 2010 and 2040 [46], and the development of renewable sources of energy is still under development, a search for a suitable transition fuel is underway. Natural gas (NG) is generally considered as the cleanest traditional fuel with an expected rise of 40% between 2017 and 2040 [35]. Due to its suitability for long-distance transport, the liquefied natural gas (LNG) has become an important trade commodity. However, the energy requirements of its production are substantial. Therefore, dozens of publications arose in the last decade aiming at the optimization of the LNG process [35, 47].

The propane-precooled mixed-refrigerant (C3MR) process is the most prevalent liquefaction technology used today and produces more LNG than any other process in the world [28]. It consists of two working cycles: the propane cycle, where the NG is cooled to ap. -35 °C and the mixed refrigerant (MR) is partially condensed by a sequential partial evaporation of compressed propane; and the mixed-refrigerant cycle, where the NG is subcooled and liquefied in a coil-wound heat exchanger due to the expansion of the MR. It is a relatively simple but highly non-linear process which makes it a great adept for studying simulation-based optimization. Moreover, the composition of NG varies greatly depending on its source. Nitrogen content in the NG can range from less than 1 mol. % to more than 20 mol. %. This highly affects the necessary pressure levels and composition of the MR, which needs to be considered.

In this work, a C3MR LNG unit producing 5 MTPA LNG while processing 12 different feedstocks (Table 1) was simulated in Aspen Plus V12 and optimized by the PAGAN-II algorithm. Process parameters and assumptions were adopted from Park et al. [48]. Peng-Robinson equation of state was selected for this study as it is the thermodynamic model of choice in virtually every LNG study [35, 47, 48]. The basic layout of the technology is displayed in Fig. 4.
Fig. 4. C3MR LNG process flow diagram.

Table 1
Different molar compositions of natural gas published in various studies.

<table>
<thead>
<tr>
<th>Label</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nitrogen content</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>0.0043</td>
<td>0.0070</td>
<td>0.0010</td>
<td>0.0049</td>
<td>0.0043</td>
<td>0.0090</td>
<td>0.0200</td>
<td>0.0100</td>
<td>0.0130</td>
<td>0.0456</td>
<td>0.0401</td>
<td>0.0497</td>
</tr>
<tr>
<td>Methane</td>
<td>0.9673</td>
<td>0.8200</td>
<td>0.8600</td>
<td>0.8723</td>
<td>0.9297</td>
<td>0.9400</td>
<td>0.9160</td>
<td>0.9000</td>
<td>0.8187</td>
<td>0.8945</td>
<td>0.8748</td>
<td>0.8689</td>
</tr>
<tr>
<td>Ethane</td>
<td>0.0155</td>
<td>0.1120</td>
<td>0.0750</td>
<td>0.0668</td>
<td>0.0418</td>
<td>0.0310</td>
<td>0.0450</td>
<td>0.0500</td>
<td>0.0611</td>
<td>0.0397</td>
<td>0.0550</td>
<td>0.0510</td>
</tr>
<tr>
<td>Propane</td>
<td>0.0055</td>
<td>0.0400</td>
<td>0.0350</td>
<td>0.0349</td>
<td>0.0123</td>
<td>0.0130</td>
<td>0.0110</td>
<td>0.0200</td>
<td>0.0650</td>
<td>0.0117</td>
<td>0.0212</td>
<td>0.0213</td>
</tr>
<tr>
<td>Isobutane</td>
<td>0.0037</td>
<td>0.0120</td>
<td>0.0100</td>
<td>0.0059</td>
<td>0.0054</td>
<td>0.0030</td>
<td>0.0050</td>
<td>0.0100</td>
<td>0.0085</td>
<td>0.0031</td>
<td>0.0089</td>
<td>0.0044</td>
</tr>
<tr>
<td>n-Butane</td>
<td>0.0009</td>
<td>0.0090</td>
<td>0.0100</td>
<td>0.0089</td>
<td>0.0054</td>
<td>0.0040</td>
<td>0.0030</td>
<td>0.0100</td>
<td>0.0196</td>
<td>0.0024</td>
<td>-</td>
<td>0.0045</td>
</tr>
<tr>
<td>Isopentane</td>
<td>0.0013</td>
<td>-</td>
<td>0.0030</td>
<td>0.0029</td>
<td>0.0004</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0042</td>
<td>0.0009</td>
<td>-</td>
<td>0.0001</td>
</tr>
<tr>
<td>n-Pentane and higher</td>
<td>0.0015</td>
<td>-</td>
<td>0.0060</td>
<td>0.0019</td>
<td>0.0007</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0099</td>
<td>0.0020</td>
<td>-</td>
<td>0.0001</td>
</tr>
<tr>
<td>Source</td>
<td>[49]</td>
<td>[50]</td>
<td>[9]</td>
<td>[51]</td>
<td>[52]</td>
<td>[53]</td>
<td>[54]</td>
<td>[55]</td>
<td>[56]</td>
<td>[57]</td>
<td>[58]</td>
<td>[48]</td>
</tr>
</tbody>
</table>
3.2 Optimization using genetic algorithm

As a demonstration, the C3MR LNG process encompassing all 12 feedstock alternatives was optimized by the proposed optimization interface using 200 individuals and 200 generations. The selected objective was to achieve minimal specific energy consumption, SEC, i.e., the objective function was stated as follows:

$$\min f(\mathbf{X}) = SEC$$

whereas

$$SEC = \frac{W}{m_{\text{LNG}}}$$

where $W$ is cumulative duty of the compressors, kW, and $m_{\text{LNG}}$ is mass flow rate of produced LNG, kg s$^{-1}$.

In the objective function (1), $\mathbf{X}$ denotes the matrix of decision variables. A total of 19 decision variables were optimized, based on [48, 59] and our previous work [28]. The decision variables as well as their ranges are listed in Table 2.

To achieve reasonable results, several constraints must be satisfied. Most of them can be presented as upper and lower bounds of the variables as presented in Table 2. Additionally, the temperature of the condensed propane (C3-00) cannot exceed 39°C. And, finally, an individual can be assumed relevant only if the simulation converged. Hence, these constraints affect the final value of the objective function in the form of a penalty function:

$$p(\mathbf{X}, r, h) = \begin{cases} f(\mathbf{X}) + r \left( \max_\mathbf{X} g(\mathbf{X}) \right)^2 & \text{if converged normally} \\ h & \text{otherwise} \end{cases}$$

where $p(\mathbf{X}, r, h)$ is a function minimized instead of $f(\mathbf{X})$, $h = 10^5$ and $r = 100$ are user-defined penalty factors, and:

$$g(\mathbf{X}) = 39 - T_{C3-00}$$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Base case</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-01 discharge pressure / kPa</td>
<td>240</td>
<td>$144 \leq P_{C3-28} \leq 336$</td>
</tr>
<tr>
<td>K-02 discharge pressure / kPa</td>
<td>391</td>
<td>$235 \leq P_{C3-30} \leq 547$</td>
</tr>
<tr>
<td>K-03 discharge pressure / kPa</td>
<td>685</td>
<td>$411 \leq P_{C3-32} \leq 959$</td>
</tr>
<tr>
<td>K-04 discharge pressure / kPa</td>
<td>1407</td>
<td>$844 \leq P_{C3-34} \leq 1,970$</td>
</tr>
<tr>
<td>K-05 discharge pressure / kPa</td>
<td>980</td>
<td>$588 \leq P_{MR-01} \leq 1,372$</td>
</tr>
<tr>
<td>K-06 discharge pressure / kPa</td>
<td>2,454</td>
<td>$1,472 \leq P_{MR-07} \leq 3,436$</td>
</tr>
<tr>
<td>K-07 discharge pressure / kPa</td>
<td>3,272</td>
<td>$1,963 \leq P_{MR-12} \leq 4,581$</td>
</tr>
<tr>
<td>K-08 discharge pressure / kPa</td>
<td>5,006</td>
<td>$3,004 \leq P_{MR-16} \leq 7,008$</td>
</tr>
<tr>
<td>TV-07 outlet pressure / kPa</td>
<td>130</td>
<td>$110 \leq P_{C3-14} \leq 182$</td>
</tr>
<tr>
<td>TV-09 outlet pressure / kPa</td>
<td>490</td>
<td>$294 \leq P_{MR-07} \leq 686$</td>
</tr>
<tr>
<td>E-08 outlet temperature (NG) / °C</td>
<td>-115.8</td>
<td>$-125.8 \leq T_{NG-04} \leq -105.8$</td>
</tr>
<tr>
<td>E-08 outlet temperature (MR, g) / °C</td>
<td>-114.5</td>
<td>$-124.5 \leq T_{MR-05} \leq -104.5$</td>
</tr>
<tr>
<td>E-08 outlet temperature (MR, l) / °C</td>
<td>-122.2</td>
<td>$-132.2 \leq T_{MR-02} \leq -112.2$</td>
</tr>
<tr>
<td>E-09 outlet temperature (NG) / °C</td>
<td>-139.7</td>
<td>$-149.7 \leq T_{NG-05} \leq -129.7$</td>
</tr>
<tr>
<td>E-09 outlet temperature (MR) / °C</td>
<td>-149.9</td>
<td>$-159.9 \leq T_{MR-06} \leq -139.9$</td>
</tr>
<tr>
<td>Composition of mixed refrigerant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nitrogen / mass fraction</td>
<td>0.0585</td>
<td>$0.01 \leq x_{N2} \leq 0.50$</td>
</tr>
<tr>
<td>Methane / mass fraction</td>
<td>0.4276</td>
<td>$0.01 \leq x_{CH4} \leq 0.50$</td>
</tr>
<tr>
<td>Ethane / mass fraction</td>
<td>0.3495</td>
<td>$0.01 \leq x_{C2H6} \leq 0.50$</td>
</tr>
<tr>
<td>Propane / mass fraction</td>
<td>0.1644</td>
<td>$0.01 \leq x_{C3H8} \leq 0.50$</td>
</tr>
</tbody>
</table>
3.3 Performance test

To assess the performance of PAGAN-II and to compare its capabilities with PAGAN and the traditional approach, four performance tests were realized. A 64-bit desktop computer with an AMD Ryzen 9 3900X 3.80 GHz 12-core processor and a 48 GB RAM was used for the analysis. The abovementioned C3MR LNG process was used as a model unit for optimization. The performance test was realized for various numbers of individuals and generations as listed in Table 3. For the sake of reproducibility, each optimization run was started with identical initial population.

Table 3
Parameters of the performance test.

<table>
<thead>
<tr>
<th>Test number</th>
<th>Algorithm</th>
<th>No. of parallel simulations</th>
<th>No. of individuals</th>
<th>No. of generations</th>
<th>Reinitialization*</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>Standard</td>
<td>1</td>
<td>50</td>
<td>20</td>
<td>no</td>
<td>Comparison</td>
</tr>
<tr>
<td></td>
<td>PAGAN</td>
<td>2, 3, 4, 6, 8, 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PAGAN-II</td>
<td>2, 3, 4, 6, 8, 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 2</td>
<td>Standard</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>no</td>
<td>Performance study</td>
</tr>
<tr>
<td></td>
<td>PAGAN-II</td>
<td>2, 3, 4, 6, 8, 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Standard</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>every 10 gen.</td>
<td>Performance study</td>
</tr>
<tr>
<td></td>
<td>PAGAN-II</td>
<td>2, 3, 4, 6, 8, 12</td>
<td></td>
<td></td>
<td>every 10 gen.</td>
<td></td>
</tr>
<tr>
<td>Test 3</td>
<td>Standard</td>
<td>1</td>
<td>100</td>
<td>50</td>
<td>every 10 gen.</td>
<td>Performance study</td>
</tr>
<tr>
<td></td>
<td>PAGAN-II</td>
<td>2, 4, 8, 12</td>
<td></td>
<td></td>
<td>every 10 gen.</td>
<td></td>
</tr>
<tr>
<td>Test 4</td>
<td>Standard</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>every 10 gen.</td>
<td>Performance study</td>
</tr>
<tr>
<td></td>
<td>PAGAN-II</td>
<td>2, 4, 8, 12</td>
<td></td>
<td></td>
<td>every 10 gen.</td>
<td></td>
</tr>
</tbody>
</table>

*Reinitialization of the simulation environment used to solve the issue described in 2.1 and 4.1.

4 Results and discussion

4.1 Performance assessment

Performance of the proposed PAGAN-II algorithm was tested against the performance of PAGAN and the standard approach in Test 1. To achieve reliable results and diminish the effect of random errors, the optimization runs for each number of parallel simulations were repeated several times as follows: 4 repetitions for standard approach, 3 repetitions for PAGAN, and 6 repetitions for PAGAN-II. Therefore, a total of 54 test runs were carried out. Average measured optimization times are displayed in Fig. 5. In the figure, one “parallel” simulation represents the standard approach. The relative computation rate in Fig. 5 represents the ratio of the optimization time using the respective number of parallel simulations to the standard approach. From the figure it is evident that both algorithms drastically reduce the necessary optimization time, whereas PAGAN-II clearly outperforms the original PAGAN algorithm; up to 6.5-fold decrease in computation time was achieved. It needs to be stated that, here and hereon, these results are highly dependent on the actual optimization problem and the used hardware.
Fig. 5. Average optimization times and average relative computation rates for Test 1.

For a deeper insight into the optimization process, the optimization times for each generation were also measured during Test 1 and visualized in Fig. 6 and Fig. 7.

Fig. 6. Measured optimization times per generation for Test 1 using PAGAN.

Fig. 7. Measured optimization times per generation for Test 1 using PAGAN-II.

It can be observed in the figures that PAGAN-II is not only faster, but the fluctuations in optimization times per generation are also flattened due to the parallel nature of the algorithm which diminishes the effect of non-converged individuals. However, it can be seen that the optimization time gradually increases with each new generation. This phenomenon can be attributed to the fact that continuous running of any software gradually consumes more RAM and overwhelms the processor cache memory which results in increased runtime. For a more representative illustration, measured optimization times of generations without non-converged individuals are plotted in Fig. 8. The slopes of trendlines in the figure show that increasing the
number of parallel simulations diminishes the effect of gradual computation slow-down and also the fact
that PAGAN-II is more effective in doing so compared to PAGAN.

![Fig. 8. Measured optimization times per generation for Test 1 comprising solely fully-converged
populations. Legend: triangles – standard approach, circles – PAGAN, squares – PAGAN-II.]

To battle the gradual computation slow-down a reinitialization concept was employed. The idea is simple –
the simulation engine is restarted every pre-set number of generations. This concept was tested in Test 2
where the number of generations was increased to 50. Results of Test 2 are visualized in Fig. 9. In the figure
there can be observed that the reinitialization of simulation environment greatly decreases the optimization
time whereas it is restored approximately to the times reached at the beginning of the optimization. Total
optimization times with different numbers of parallel simulations and differences caused by employing the
reinitialization concept are displayed in Fig. 10. In this case one needs to define the base for assessing the
relative computation rate because the reinitialization concept also accelerates the optimization using
standard approach by a factor of ap. 1.5. Hence, three sets of different relative rates can be obtained. The
most important, however, is the relative rate which puts in comparison the optimization time with and
without using the reinitialization concept (orange line).
**Fig. 9.** Measured optimization times per generation for Test 2. **Legend:** squares – optimization with reinitialization, circles – optimization without reinitialization; full symbols – fully converged population, empty symbols – population containing non-converged individuals.

**Fig. 10.** Optimization times and relative computation rates for Test 2.
In Tests 3 and 4, the number of individuals and generations were set to values (Table 3) which are closer to commonly used values in optimization studies. Due to time reasons, the number of tests was reduced so that only optimizations using 1, 2, 4, 8, and 12 parallel simulations were assessed. Fig. 11 and Fig. 12 depict the measured optimization times per generation. Total optimization times and relative computation rates for Tests 3 and 4 are shown in Fig. 13 and Fig. 14.

**Fig. 11.** Measured optimization times per generation for Test 3. **Legend:** full symbols – fully converged population, empty symbols – population containing non-converged individuals.

**Fig. 12.** Measured optimization times per generation for Test 4. **Legend:** full symbols – fully converged population, empty symbols – population containing non-converged individuals.
4.2 Algorithm’s behavior

Fig. 10, Fig. 13 and Fig. 14 document a decrease in relative computation rate between 8 and 12 parallel simulations. To better understand the PAGAN-II algorithm and to explain this phenomenon, a mathematical model mimicking the algorithm’s behavior was created and thoroughly studied. The mathematical model is provided in the **Supplementary material**. Based on the study of the measured algorithm performance, the following assumptions were applied:

- simulation time for each individual is linearly dependent on the elapsed optimization time (to account for the gradual slow-down),
- a quadratic relationship exists between the intercept of the simulation time and the number of parallel simulations (i.e., increasing the number of parallel simulations lengthens the base simulation time),
- read-and-write time of the algorithm is linearly dependent on the elapsed optimization time (i.e., Matlab performance is also affected over time),
- reinitialization of the simulation environment resets the simulation time for individuals but does not affect the read-and-write time of the algorithm (i.e., Aspen Plus simulations are reinitialized but the Matlab algorithm is not),
- converged and unconverged individuals have different simulation times,
- the number of unconverged individuals in a model generation is equal to the number of unconverged individuals in the respective measured generation,
- exact individuals considered unconverged are randomly chosen.

These assumptions result in the following set of equations:

\[ \tau^C_i = \tau^C_b + X_1 \tau_e \]  \hspace{1cm} (5)
\[ \tau^U_i = \tau^U_b + X_2 \tau_e \]  \hspace{1cm} (6)
\[ \tau_{RW} = X_3 + X_4 \tau_e \]  \hspace{1cm} (7)
\[ \tau^C_b = X_5 N^2 - X_6 N + X_7 \]  \hspace{1cm} (8)
\[ \tau^U_b = X_8 N^2 - X_9 N + X_{10} \]  \hspace{1cm} (9)
where $\tau_i^C, \tau_i^{UI}$ are the simulation times of the $i$-th converged and unconverged individuals, s, respectively,
$\tau_b^C, \tau_b^{UI}$ are the base times of converged and unconverged simulations, s, respectively, $\tau_i^{RW}$ is the read-and-write time of the $i$-th individual, s, $N$ is the number of parallel simulations, $\tau_e$ is the elapsed optimization time, s, and $X_1 - X_{10}$ are parameters of the mathematical model. Parameters $X_1 - X_{10}$ were fitted to the measured data using the least-squares method.

Resulting parameters of the model are summarized in Table 4. The model was verified against the measured times per generation as well as the total optimization time. Results of the verification for Test 3 are shown in Fig. 15 and summarized in Table 5 and in Fig. 16. It needs to be reminded here that, even though the mathematical model is generally applicable to any problem, the actual values of parameters $X_1 - X_{10}$ depend on the actual optimization problem and the used hardware.

Table 4
Parameters of the mathematical model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 / s$</td>
<td>$7.328 \times 10^{-4}$</td>
</tr>
<tr>
<td>$X_2 / s$</td>
<td>$2.571 \times 10^{-4}$</td>
</tr>
<tr>
<td>$X_3 / s$</td>
<td>$8.885 \times 10^{-1}$</td>
</tr>
<tr>
<td>$X_4 / s$</td>
<td>$6.983 \times 10^{-7}$</td>
</tr>
<tr>
<td>$X_5 / s$</td>
<td>$4.967 \times 10^{-2}$</td>
</tr>
<tr>
<td>$X_6 / s$</td>
<td>$4.798 \times 10^{-2}$</td>
</tr>
<tr>
<td>$X_7 / s$</td>
<td>$4.357$</td>
</tr>
<tr>
<td>$X_8 / s$</td>
<td>$3.964 \times 10^{-2}$</td>
</tr>
<tr>
<td>$X_9 / s$</td>
<td>$1.124 \times 10^{-1}$</td>
</tr>
<tr>
<td>$X_{10} / s$</td>
<td>$5.428$</td>
</tr>
</tbody>
</table>
Standard approach: 2 parallel simulations:

4 parallel simulations: 12 parallel simulations:

Fig. 15. Measured and model-obtained optimization times per generation for Test 3.

Table 5

<table>
<thead>
<tr>
<th>Time / min</th>
<th>Number of parallel simulations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 2</td>
<td></td>
</tr>
<tr>
<td>Measurement</td>
<td>255</td>
</tr>
<tr>
<td>Model</td>
<td>258</td>
</tr>
<tr>
<td>Error</td>
<td>+0.96%</td>
</tr>
<tr>
<td>Test 3</td>
<td></td>
</tr>
<tr>
<td>Measurement</td>
<td>675</td>
</tr>
<tr>
<td>Model</td>
<td>663</td>
</tr>
<tr>
<td>Error</td>
<td>-1.91%</td>
</tr>
<tr>
<td>Test 4</td>
<td></td>
</tr>
<tr>
<td>Measurement</td>
<td>1213</td>
</tr>
<tr>
<td>Model</td>
<td>1207</td>
</tr>
<tr>
<td>Error</td>
<td>-0.52%</td>
</tr>
</tbody>
</table>

Verification of the mathematical model.
Results of the verification show that the mathematical model can mimic the algorithm’s behavior satisfactorily as demonstrated in Fig. 15. The model’s results deviate by no more than 2% from the measured data for the majority of cases and up to 5% in some specific cases where the deviation can be attributed to the uncertainty of measurement rather than the model’s precision.

The verified model was subsequently used for a deeper study of the algorithm’s behavior. Activity of the individual simulation instances are displayed in Table 6. “White spaces” in the plots enclosed in the table represent the time when the respective simulation cores are idle, i.e., they are not running a simulation, and they are waiting for another individual to be sent to the simulator. This “waiting time” is the time which is necessary for the algorithm to read the results of the simulation, to write them into an internal variable, and to send new parameters to the simulator, i.e., the read-and-write time. It is evident that the algorithm uses 8 parallel simulations more effectively when compared to 12. Numerically speaking, when using 8 parallel simulations, the simulation engine is active for 80-85% of the total optimization time, whereas some simulations are active for only 77% of the total time when using 12 parallel simulations. Therefore, the algorithm spends a considerable amount of time just “waiting” when using 12 parallel simulations which results in a longer optimization time.

**Fig. 16.** Comparison of measured and model-provided total optimization time (log scale). **Legend:** full bars – measured data; lined bars – model data.
### Table 6
Activity of individual simulation instances over the optimization time and their overall utilization.

<table>
<thead>
<tr>
<th>No.</th>
<th>Activity over time</th>
<th>Util.</th>
<th>Activity over time</th>
<th>Util.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>85%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>85%</td>
</tr>
<tr>
<td>2</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>85%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>85%</td>
</tr>
<tr>
<td>3</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>84%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>85%</td>
</tr>
<tr>
<td>4</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>84%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>85%</td>
</tr>
<tr>
<td>5</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>78%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>80%</td>
</tr>
<tr>
<td>6</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>78%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>80%</td>
</tr>
<tr>
<td>7</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>78%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>80%</td>
</tr>
<tr>
<td>8</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>77%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>80%</td>
</tr>
<tr>
<td>9</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>78%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>77%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>77%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td>78%</td>
<td><img src="Image" alt="Activity Bar" /></td>
<td></td>
</tr>
</tbody>
</table>

### 4.3 Performance case studies
The performance test unveiled a significant impact of the read-and-write time on the overall performance of the algorithm. However, it was assumed that this impact is dependent on the base simulation time. To test
this assumption, three case studies with various simulation times were conducted: using the conditions of
Test 3, using the conditions of Test 4, and employing 200 individuals and 200 generations. In the studies, it
was assumed that the read-and-write time and its intercept as well as the impact of the number of parallel
simulations remain unchanged.

Because the distribution of unconverged individuals in the case-study optimization was unknown, a function
for the fraction of unconverged individuals (FUI), was constructed as follows:

\[
FUI = \begin{cases} 
0 & ; g = 1 \\
2.19 \times 10^{-5} g^4 - 7.616 \times 10^{-4} g^3 + 6.636 \times 10^{-3} g^2 + 1.985 \times 10^{-3} g + 5.128 \times 10^{-3} & ; 1 < g < 15 \\
3.99 \times 10^{-4} g^2 - 2.217 \times 10^{-2} g + 0.308 & ; 15 \leq g < 30 \\
0 & ; g \geq 30 
\end{cases}
\]  

(10)

where \( g \) stands for the number of a generation. The function was fitted to the average fractions of
unconverged individuals in Tests 1 to 4 (Fig. 17.)

The case study was done with the following simulation times: 1, 3, 5, 10, 15, and 30 seconds. First, the
effect of different simulation times was studied for the case of Test 3, i.e., 100 individuals and 50
generations. Results of the case study for Test 3 are displayed in Fig. 18 and Fig. 19. Fig. 18 shows that the
base simulation time has a significant effect on the total optimization time achieved by the algorithm. This
effect is better observable from the values of relative computation rates (Fig. 19) where it can be seen that,
for fast simulations, i.e., with a simulation time of less than 5 seconds, the relative computation rate reaches
a maximum. Therefore, in such cases, using additional parallel simulations is not beneficial. Nonetheless,
the PAGAN-II algorithm still achieves over 300% increase in the optimization rate. However, it can be
observed that using additional parallel simulations is highly beneficial in the case of demanding simulations,
i.e., with a simulation time over 10 seconds, where almost 1300% increase in the computation speed can be
achieved.
Similar results can be observed for the case of Test 4, i.e., optimization using 100 individuals over 100 generations (Fig. 20). However, the situation slightly changes when more individuals and generations are added. In Fig. 21, it can be observed that when using 200 individuals and 200 generations (which is a common practice in optimization using the genetic algorithm), the impact of read-and-write time is lower and using a higher number of parallel simulations becomes beneficial for simulations with a simulation time over approx. 4 seconds. Furthermore, significantly higher relative computation rates (up to 2100%) can be achieved in such optimizations.

4.4 Optimization of the C3MR LNG process

The proposed algorithm was utilized in optimization of a 5 MTPA C3MR LNG process with 12 different feedstocks. 200 individuals and 200 generations were used in the optimization. Prior to the optimization of
all 12 alternatives, the model’s prediction was used to choose the optimal number of parallel simulations. In
Fig. 22., it can be observed that an optimization of 12 process alternatives with 200 individuals and
generations would take approx. 69 days (or more than 2 months) of non-stop computation if the standard
approach was used. This time was drastically reduced to less than one week (approx. 7 days) with the use
of PAGAN-II with 12 parallel simulations. It should be noted that the model prediction of the optimization
time using the standard approach assumes reinitialization of the simulation environment and, hence, the
“true” standard approach would require significantly longer time than 2 months.

Fig. 22. Model prediction of the necessary time for optimization of the C3MR LNG process with 12
various feedstocks and the associated relative computation rate.

Finally, results of the optimization are summarized in Table 7. It can be observed that a considerable
decrease in specific energy consumption has been achieved with all types of feedstocks. Depending on the
actual feedstock, the SEC was decreased in the range of 15 to 30%. Regarding the process parameters,
pressure ratios in the mixed-refrigerant compressor, outlet temperatures of the main coil-wound heat
exchanger, and the content of nitrogen in the mixed refrigerant have proven to be the most important.
### Table 7
Optimized parameters of the C3MR process.

| Parameter                        | Base (L) | A | B | C | D | E | F | G | H | I | J | K | L |
|----------------------------------|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nitrogen content                 | High     | Low| Low| Low| Low| Low|Low| Medium| Medium| Medium| High| High| High| High |
| K-01 discharge pressure / kPa    | 240      | 288| 287| 287| 287| 287| 296| 287    | 287   | 287   | 287 | 289 | 287 | 287 |
| K-02 discharge pressure / kPa    | 391      | 517| 518| 518| 523| 521| 519| 520    | 519   | 520   | 519 | 520 | 521 | 521 |
| K-03 discharge pressure / kPa    | 685      | 920| 922| 921| 924| 922| 922| 922    | 921   | 922   | 921 | 921 | 925 | 922 |
| K-04 discharge pressure / kPa    | 1,407    | 1,424| 1,426| 1,430| 1,426| 1,419| 1,417| 1,332 | 1,426 | 1,426 | 1,423 | 1,415 | 1,416 |
| K-05 discharge pressure / kPa    | 980      | 1,243| 1,241| 1,240| 1,240| 1,244| 1,244| 1,247 | 1,241 | 1,241 | 1,241 | 1,245 | 1,248 | 1,247 |
| K-06 discharge pressure / kPa    | 2,454    | 2,101| 2,104| 2,102| 2,105| 2,107| 2,103| 2,105 | 2,104 | 2,104 | 1,945 | 2,105 | 2,108 |
| K-08 discharge pressure / kPa    | 5,006    | 4,324| 4,324| 4,320| 4,323| 4,328| 4,324| 4,329 | 4,322 | 4,325 | 4,325 | 4,331 | 4,329 |
| TV-07 outlet pressure / kPa      | 130      | 139| 140| 140| 140| 140| 140| 140    | 140   | 140   | 140 | 140 | 140 | 140 |
| TV-09 outlet pressure / kPa      | 490      | 396| 404| 395| 402| 399| 369| 396    | 399   | 398   | 398 | 398 | 398 | 400 |
| E-08 outlet temp. (NG) / °C      | -115.8   | -116.3| -119.2| -119.3| -117.3| -116.3| -118.2| -118.2 | -117.2 | -117.6 | -116.3 | -116.0 |
| E-08 outlet temp. (MR, g) / °C   | -114.5   | -113.7| -113.4| -113.9| -112.4| -113.6| -119.0| -111.2 | -113.2 | -113.2 | -113.8 | -113.1 |
| E-08 outlet temp. (MR, l) / °C   | -122.2   | -112.6| -113.4| -113.9| -112.9| -120.9| -112.4| -113.9 | -113.9 | -113.9 | -113.5 | -120.9 | -121.9 |
| E-09 outlet temp. (NG) / °C      | -139.7   | -130.6| -130.8| -132.9| -131.4| -131.4| -131.3| -130.4 | -130.4 | -130.3 | -130.9 | -130.4 |
| E-09 outlet temp. (MR) / °C      | -149.9   | -159.3| -159.3| -159.3| -157.3| -158.2| -154.9| -156.3 | -159.3 | -159.3 | -156.2 | -157.7 |
| Composition of MR                |          |     |     |     |     |     |     |        |        |        |        |        |        |
| Nitrogen / mass fraction         | 0.0585   | 0.0352| 0.0316| 0.0360| 0.0360| 0.0315| 0.0274| 0.0240 | 0.0277 | 0.0360 | 0.0298 | 0.0315 | 0.0315 |
| Methane / mass fraction          | 0.4276   | 0.3638| 0.3709| 0.3727| 0.3727| 0.3758| 0.3713| 0.3800 | 0.3759 | 0.3727 | 0.3640 | 0.3759 | 0.3757 |
| Ethane / mass fraction           | 0.3495   | 0.4407| 0.3969| 0.3922| 0.3922| 0.3975| 0.4020| 0.4084 | 0.3964 | 0.3922 | 0.4172 | 0.3976 | 0.3976 |
| Propane / mass fraction          | 0.1644   | 0.1604| 0.2006| 0.1991| 0.1991| 0.1952| 0.1992| 0.1876 | 0.2000 | 0.1991 | 0.1890 | 0.1950 | 0.1952 |
| Base case SEC / kJ kg⁻¹          | 1.317    | 1.154| 1.063| 1.021| 1.073| 1.098| 1.101| 1.049 | 1.087 | 1.322 | 1.091 | 1.102 |
| Optimized SEC / kJ kg⁻¹          | 964.3    | 812.6| 829.7| 854.6| 915.3| 938.4| 908.8| 881.9 | 774.7 | 956.0 | 922.1 | 938.5 |
| Decrease / %                     | 26.77    | 29.60| 28.44| 16.33| 14.73| 14.53| 17.44| 15.97 | 28.70 | 27.63 | 15.45 | 14.81 |
4.5 Study limitations and further potential

Time-related challenges of simulation-based optimization are well documented in literature as well as in the presented study. Various authors battled these challenges differently, as summarized in Table 8. While most studies utilize artificial neural networks or surrogate modeling, some authors proposed their modifications of the original algorithm to improve its efficiency. However, it is virtually impossible to quantify the impact of their modifications and/or approaches on the overall optimization time and to compare those results with the results of this study. Up to date, there is, to our best knowledge, no comprehensive interface which would utilize the parallelization concept apart from the Adv:PO interface proposed by Ernst et al. [5] and used by Johannsen et al. [41]. However, even this interface relies mostly on a modified algorithm and the mentioned parallelization concept is not discussed in detail. Therefore, we consider PAGAN-II as a novel and unique approach in the field of simulation-based optimization.

Results of the performance assessment and the subsequent case studies proved the potential of PAGAN-II as a generally applicable optimization tool. Even though the exact results of the performance test are heavily dependent on the used hardware, it can be declared that the algorithm’s internal behavior is not. Therefore, any future user would find out that:

- PAGAN-II is especially useful when dealing with demanding simulation problems.
- Effectivity of PAGAN-II decreases when dealing with “fast” simulations, although significant increase in relative computation rate can be achieved, nonetheless.

It should be noted that the presented results were achieved by a desktop computer and that the current version of PAGAN-II utilizes the original GA/NSGA-II algorithms. However, since the interface is highly modifiable, more efficient versions of the algorithm and/or other optimization algorithms can be easily implemented which might lead to another acceleration of the optimization rate. Furthermore, an option to utilize clustered CPU systems (servers) exist which could potentially speed up the computation. Hence, the proposed interface has a considerable potential in both academic and industrial applications.

Table 8

<table>
<thead>
<tr>
<th>Author</th>
<th>Modified GA/NSGA-II</th>
<th>Surrogate modeling</th>
<th>Artificial neural networks</th>
<th>Parallelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bravo-Bravo et al. [21]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>Gómez-Castro et al. [38]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Ernst et al. [5]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>Rangaiah et al. [31]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Al-Zareer et al. [37]</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Ali et al. [33]</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>De Buck et al. [29]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Qiu et al. [39]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>Farsi et al. [6]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Razi et al. [7]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>Sabbagh et al. [34]</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>De Buck et al. [30]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Johannsen et al. [41]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>Sabbagh et al. [35]</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Enayatizadeh et al. [36]</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Okoji et al. [40]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>This study</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
</tbody>
</table>

5 Conclusions

Latest developments in the fields of process simulation and modern stochastic optimization fuel the need for a robust tool for time-effective simulation-based optimization. While the procedure of linking process simulators to modern optimization algorithms is well known, the optimization suffers from extensive computation time. This paper resolves this issue by proposing a novel optimization tool – the Parallel
The PAGAN-II algorithm performed well in optimization of a 5 MTPA C3MR LNG process where a 15 to 30% decrease in specific energy consumption was achieved. At the same time, the optimization speed was increased 13-fold compared to traditional approach.

Results of the case studies indicated that the performance of the PAGAN-II algorithm is highly dependent on the nature of the optimized simulation. While optimization of highly demanding processes with longer simulation times could be accelerated up to 2100%, the benefits of using PAGAN-II are lower in cases of simpler and faster simulations. Nonetheless, an approx. 300% increase in computation rate could be achieved with such simulations. Therefore, the algorithm can be declared to be generally applicable and considerably useful in any simulation-based optimization.

CRediT authorship contribution statement


Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This study was supported by the Slovak Research and Development Agency under contract nos. APVV-18-0134 and APVV-19-0170, and by the Slovak Scientific Agency, grant no. VEGA 1/0511/21.

Appendix

```matlab
function Obj = OptEngine(app, Aspen, X)
% This is the PAGAN optimization engine. The engine continuously scans the
% simulation cores for their activity and assigns available cores a task.
% If an individual has already been simulated by the respective core, the
% core is instructed to acquire simulation results. Subsequently, or if
% there is not an individual previously simulated by the respective core,
% the core is instructed to find another individual that has not yet been
% simulated and execute simulation. This procedure repeats until all
% individuals are simulated and all simulation results are acquired.
% Each individual is assigned two parameters:
% The first parameter represents the overall status of and individual
% 1 - evaluation of the individual is not finished
% 0 - evaluation of the individual is finished
% The second parameter represents the activity of the cores
% 0 - the individual has not yet been simulated
% n - the individual has been simulated by the n-th core but the results
% have not yet been acquired
% -1 - evaluation of the individual is finished
% Status = [ones(size(X, 1) 1) zeros(size(X, 1), 1)]; % Initially, the first and the second
% parameter of each individual are set to
% 1 and 0, respectively.
```

```text
function Obj = OptEngine(app, Aspen, X)
% This is the PAGAN optimization engine. The engine continuously scans the
% simulation cores for their activity and assigns available cores a task.
% If an individual has already been simulated by the respective core, the
% core is instructed to acquire simulation results. Subsequently, or if
% there is not an individual previously simulated by the respective core,
% the core is instructed to find another individual that has not yet been
% simulated and execute simulation. This procedure repeats until all
% individuals are simulated and all simulation results are acquired.
% Each individual is assigned two parameters:
% The first parameter represents the overall status of an individual
% 1 - evaluation of the individual is not finished
% 0 - evaluation of the individual is finished
% The second parameter represents the activity of the cores
% 0 - the individual has not yet been simulated
% n - the individual has been simulated by the n-th core but the results
% have not yet been acquired
% -1 - evaluation of the individual is finished
% Status = [ones(size(X, 1), 1) zeros(size(X, 1), 1)]; % Initially, the first and the second
% parameter of each individual are set to
% 1 and 0, respectively.
```
while sum(Status(:, 1)) > 0
    for j = 1 : app.Cores.Value
        if Aspen(j).Engine.IsRunning == 1
            continue
        else
            done = find(Status(:, 2) == j);
            if ~isempty(done)
                Obj(done, :) = AspenOutput(Aspen, j);
                Status(done, 2) = -1;
                Status(done, 1) = 0;
            end
done = find(Status(:, 2) == j);
    end
end

References


Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Model.rar
- PAGANII.rar