Method Article

A parallelized hybrid genetic algorithm with differential evolution for heat exchanger network retrofit

Jan A. Stampfli\textsuperscript{a,b,*}, Donald G. Olsen\textsuperscript{a}, Beat Wellig\textsuperscript{a}, René Hofmann\textsuperscript{b}

\textsuperscript{a}Lucerne University of Applied Sciences and Arts, Competence Center Thermal Energy Systems and Process Engineering, Technikumstrasse 21, Horw 6048, Switzerland
\textsuperscript{b}TU-Wien, Institute of Energy Systems and Thermodynamics, Getreidemarkt 9/BA, Vienna 1060, Austria

\textbf{Abstract}

The challenge of heat exchanger network retrofit is often addressed using deterministic algorithms. However, the complexity of the retrofit problems, combined with multi-period operation, makes it very difficult to find any feasible solution. In contrast, stochastic algorithms are more likely to find feasible solutions in complex solution spaces. This work presents a customized evolutionary based optimization algorithm to address this challenge. The algorithm has two levels, whereby, a genetic algorithm optimizes the topology of the heat exchanger network on the top level. Based on the resulting topology, a differential evolution algorithm optimizes the heat loads of the heat exchangers in each operating period. The following bullet points highlight the customization of the algorithm:

- The advantage of using both algorithms: the genetic algorithm is used for the topology optimization (discrete variables) and the differential evolution for the heat load optimization (continuous variables).
- Penalizing and preserving strategies are used for constraint handling
- The evaluation of the genetic algorithm is parallelized, meaning the differential evolution algorithm is performed on each chromosome parallel on multiple cores.

\textcopyright{} 2022 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

\textbf{Article info}

Method name: Evolutionary based heat exchanger network retrofit for multi-period processes

Keywords: Heat exchanger network (HEN), Retrofit, Multi-period, Meta-heuristics, Genetic algorithm, Differential evolution, Parallel processing

Article history: Received 15 March 2022; Accepted 19 April 2022; Available online 26 April 2022

\textbf{Article history}

DOI of original article: 10.1016/j.compchemeng.2022.107771

* Corresponding author at: Lucerne University of Applied Sciences and Arts, Competence Center Thermal Energy Systems and Process Engineering, Technikumstrasse 21, 6048 Horw, Switzerland.

E-mail addresses: jan.stampfli@hslu.ch (J.A. Stampfli), donald.olsen@hslu.ch (D.G. Olsen), beat.wellig@hslu.ch (B. Wellig), rene.hofmann@tuwien.ac.at (R. Hofmann).

https://doi.org/10.1016/j.mex.2022.101711
2215-0161/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)
Specification table

| Subject area:                        | Chemical Engineering, Computer Science |
|-------------------------------------|----------------------------------------|
| More specific subject area:         | Process Integration, Heat exchanger network retrofit |
| Method name:                        | Evolutionary based heat exchanger network retrofit for multi-period processes |
| Name and reference of original method: | n/a                                    |
| Resource availability:              | doi:10.5281/zenodo.441140               |

Sets

- $C = \{1 \ldots NC\}$: Set of cold streams
- $CH = \{1 \ldots CH\}$: Set of heat load chromosomes
- $CT = \{1 \ldots CT\}$: Set of topology chromosomes
- $E = \{1 \ldots E\}$: Set of heat exchangers
- $H = \{1 \ldots H\}$: Set of hot streams
- $K = \{1 \ldots K\}$: Set of enthalpy stages
- $GH = \{1 \ldots GH\}$: Set of heat loads generations
- $GT = \{1 \ldots GT\}$: Set of topology generations
- $OP = \{1 \ldots OP\}$: Set of operating periods

Parameter

- $a$: Admixer existence (boolean)
- $b$: Bypass existence (boolean)
- $CR$: Crossover probability
- $EAM$: Exchanger address matrix / GA chromosome
- $ex$: Heat exchanger existence (boolean)
- $F_p$: Perturbation factor
- $f$: Fitness function
- $h$: Penalty function
- $MT$: Mutation probability
- $n$: number of infeasible solutions
- $P$: Population
- $U$: trial chromosome
- $V$: donor chromosome
- $X$: Heat load matrix / DE chromosome
- $\Delta$: Infeasibility penalty

Subscripts

- $best$: best found solution
- $init$: Initial solution
- $viol$: constraint violations

Abbreviations

- $DE$: Differential evolution
- $EAM$: Exchanger address matrix
- $GA$: Genetic algorithm
- $HEN$: Heat exchanger network
- $HENS$: Heat exchanger network synthesis
- $HEX$: Heat exchanger
- $HoF$: Hall of Fame
- $TAC$: Total annual cost

Method details

Heat exchanger network synthesis (HENS) is an important tool to design energy efficient production plants in process industry. Furman and Sahinidis [1] showed, that HENS is $\mathcal{NP}$-hard in the strong sense. The heat exchanger network retrofit formulation, is an extension of HENS with increased complexity. Complexity is further increased by the additional dimension of operating periods, the possibility to integrate bypasses or admixers, and practical constraints. It is unlikely that deterministic algorithms can provide a feasible solution for such problems on a large-scale. Stochastic algorithms are able to handle such problems. Therefore, a hybrid two-level evolutionary algorithm based on genetic algorithm (GA) and differential evolution (DE) is developed. The concept of GA was first introduced by Holland [2] and further extended by Goldberg [3]. DE was introduced by Storn [4],
Storn and Price [5]. In the developed algorithm, a GA handles the topology optimization in the top level and a DE in the sub level is used to optimize the operation parameter.

**Topology optimization using genetic algorithm**

In order to define the topology of a HEN, integer values are used to address the position of each heat exchanger (HEX). Genetic algorithms, which are based on the evolution process in nature, are suitable to handle such discrete variables. Thereby, individuals within a population are compared and only the fittest of them survive. During the process, new individuals are generated through the mating of the fittest individuals. Coping errors (mutations) may occur during mating. An overview of the algorithm is shown in Fig. 1. The following sections describe the evolutionary operators in detail.

**Genetic algorithm - initialization**

In a first step, a population $P_t = \{\text{EAM}_0 \ldots \text{EAM}_{\text{NCT}}\}$ with NCT random individual topologies, hereafter called chromosomes, is initialized. Each of these chromosomes represents a HEN topology and is described using an exchanger address matrix (EAM). An EAM of an example topology is shown in Fig. 2. A chromosome contains the following genes (configurations of a HEX): (e) HEX number, (i) hot stream number, (j) cold stream number, (k) enthalpy stage number, (b_h) bypass on hot stream, (a_h) admixer on hot stream, (b_c) bypass on cold stream, (a_c) admixer on cold stream, and (ex) existence.
of the HEX. During the initialization, for each HEX it is first randomly decided if the HEX exists. For all existing HEXs, a random hot stream, cold stream, and enthalpy stage is defined. To ensure only feasible solution, the preserving constraint handling technique is used (e.g., for the stream number a random value between zero and the number of hot streams NH is selected). Bypass and admixer are not initialized, thus their existence depends on the heat loads. Hence, the need of a mixer and its configuration is determined in the DE.

**Genetic algorithm - evaluation**

After the initialization, each chromosome needs to be evaluated. In order to define the required areas and thus, the resulting cost, the heat loads need to be defined first. The optimal heat loads are found by the DE. To reduce computation time, the penalizing constraint handling technique is used. Thereby, only feasible topologies are optimized using DE. Infeasible solutions are evaluated by a penalty function which is less computationally expensive. The fitness $f^{gt}_{DE,ct}$ of the topology chromosome $ct$ in the topology generation $gt$ is given by

$$f^{gt}_{DE,ct} (\text{EAM}^{gt}_{ct}) = \begin{cases} f^{gt}_{DE,ct} (\text{EAM}^{gt}_{ct}, X^{gt}_{ct, best}) & \text{if } n^{gt}_{GA,viol} = 0 \\ 1 & \text{otherwise} \end{cases}$$

whereby, $f^{gt}_{DE, best}$ is the fitness of the DE in function of the EAM and the heat loads of the best DE solution $X^{gt}_{ct, best}$. The distance of each violated constraint $c$ is given by $n^{gt}_{GA,viol,c}$. For infeasible topologies, the fitness is defined by reciprocal value of the penalty function, given by

$$h^{gt}_{GA} (n^{gt}_{GA,viol}) = \Delta + \sum_{c \in C_{GA}} \left(0 - n^{gt}_{GA,viol,c} \right)^2$$

with $\Delta >> TAC_{init}$

whereby $C_{GA}$ is the set of all GA constraints. A constant value $\Delta$, which is larger than the initial total annual cost of the existing process, is added to the sum of the squared distances to the feasible region.

**Genetic algorithm - selection**

To choose the parent chromosomes for mating, tournament selection is performed. This means that the fittest out of a given number of randomly selected chromosomes is chosen.

**Genetic algorithm - crossover**

With a crossover probability of $CR_{GA}$, the parents mate. Thereby, a one-point crossover is performed. Both parent chromosomes are cut below a random selected HEX number. The HEXs above the cut are swapped between the two chromosomes. The two resulting children chromosomes replace the parent chromosomes and are evaluated.

**Genetic algorithm - mutation**

In nature, during the mating process in the crossover, coping errors (mutations) may occur. In the algorithm, with a mutation probability of $MIT_{GA}$, for each allele (a single scalar value within a gene), a mutation occurs. For the mutation in the existence of a HEX gene (ex), a random bit flip is performed.
In the genes (i), (j), and (k), the value is changed within the upper and lower boundary (e.g., for (i) the number of existent hot streams) of the gene using an uniform distribution. It is ensured by using the preserving strategy, none of these values exceed their boundaries.

**Genetic algorithm - next generation**

After the application of the evolutionary operators (selection, crossover, and mutation), the new generated topologies are evaluated and the population is updated by replacing the parent chromosome with the new children chromosomes for the next generation. In order to keep track of the best solutions, a Hall of Fame (HoF) list is created. This list contains the current best solutions. If in the evaluation step a fitter solution is found, the HoF list is updated. With this additional feature, the flexibility of the algorithm in use is increased as the engineer is now able to choose between the most promising solutions. The algorithm is terminated as soon as the maximal number of topology generations NGT is reached.

**Heat load optimization using differential evolution**

In contrast to the topology optimization, the heat loads are continuous variables with upper and lower bounds. Differential evolution algorithms are best to deal with the continuous variables. Compared to deterministic algorithms such as gradient descent, DE does not require the model to be differentiable. DE algorithms use the same concepts of evolution as GAs. However, the order of the evolutionary operators is reversed. There are different options for the DE configuration. In this case, the standard configuration DE/rand/1/bin is used. This means that the individuals for mutation are selected randomly, only one difference for perturbation (F: perturbation factor) is considered, and a binomial crossover is performed. In Fig. 3, an overview of the algorithm is shown. The following sections describe the algorithm in detail.

**Differential evolution - initialization**

For each feasible topology from the GA, a population \( R_h = \{X_0 \ldots X_{NCH}\} \) with NCH random heat load chromosomes is initialized. Each of these chromosomes consists of all the heat loads of each heat exchanger in every OP, resulting in a two-dimensional array. Chromosome \( ch \) is given by

\[
X_{ch} = (Q_{ch}^{op})
\]

whereby the heat load is an array of the size \( NE \times NOP \). The initialization of the heat loads of existing HEXs \( Q_{ch,e}^{op} \) is constraint by a minimal user defined value and a maximal value given by the enthalpy differences of the connected streams (case study dependent value).

**Differential evolution - evaluation**

For the evaluation of the population of heat loads, the fitness of each chromosome needs to be determined. Infeasible solutions are evaluated by a penalty function. In each generation \( gh \) the fitness \( f_{DE.ch}^{gh} \) is given by

\[
f_{DE.ch}^{gh} = \begin{cases} 
1 & \text{if } n_{DE.viol}^{gh} = 0 \\
\frac{TAC(EAM_{ct}^{gh},X_{ch}^{gh})}{\sum_{e=1}^{NE} Q_{ch,e}^{op}} & \text{otherwise.}
\end{cases}
\]

whereby, the \( TAC \) is the total annual cost of the current topology and heat loads. TAC consists the yearly operating cost and the investment cost for the retrofit. The detailed model for calculating the TAC is formulated in the corresponding research paper [6]. In order to minimize the TAC, the reciprocal value represents the fitness of a chromosome. Each distance of a heat load constraint violation is given by \( n_{DE.viol,c}^{gh} \). For infeasible operation parameter, the fitness is defined by reciprocal
Fig. 3. Flowchart of Differential Evolution for operation parameter optimization.

The value of the penalty function, given by

$$h_{DE}(n_{DE,\text{viol}}) = \Delta + \sum_{c \in \mathcal{C}_{DE}} \left(0 - n_{\text{viol},c}\right)^2 \text{ with } \Delta >> TAC_{\text{init}}$$  \hspace{1cm} (6)

whereby \(\mathcal{C}_{DE}\) is the set of all DE constraints. A constant value \(\Delta\) which is larger than the initial TAC is added to the sum of the squared distances to the feasible region.

**Differential evolution - mutation**

The first evolutionary operation in a DE is mutation. Thereby, a three non-equal chromosomes \(X_{r_1}^{gh}, X_{r_2}^{gh}, X_{r_3}^{gh}\) \((r_1 \neq r_2 \neq r_3)\) of the current generation \(gh\) are selected randomly. A new donor chromosome is generated by

$$v_{ch}^{gh} = X_{r_1}^{gh} + F_p \left( X_{r_2}^{gh} - X_{r_3}^{gh} \right)$$  \hspace{1cm} (7)

whereby one difference for perturbation is used and weighted with the perturbation factor \(F_p \in [0, 2]\). After the mutation, for all infeasible heat loads a new random value within the boundaries is assigned.
Differential evolution - crossover

In the crossover operator, a trial chromosome $U_{ch}^{gh}$ is generated by

$$u_{ch,e}^{gh, oc} = \begin{cases} u_{ch,e}^{gh, oc}, & r^{gh} < CR_{DE} \lor p = r \\ x_{ch,e}^{gh, oc}, & \text{otherwise} \end{cases}$$

(8)

whereby, $r^{gh} \sim U(0, 1)$ has a uniform distribution. With a probability of $CR_{DE}$ a crossover is performed. To ensure at least one crossover per operating parameter, a random index within the chromosome is chosen for which crossover is always performed.

Differential evolution - selection

In the selection operator, the new created trial chromosome is evaluated. To determine the new target chromosome for the next generation $gh + 1$ with

$$x_{ch}^{gh+1} = \begin{cases} u_{ch}^{gh}, & \text{if } f(u_{ch}^{gh}) > f(x_{ch}^{gh}) \\ x_{ch}^{gh}, & \text{otherwise}, \end{cases}$$

(9)

a simple greedy selection is performed.

Differential evolution - next generation

For each generation, the evolutionary operators (mutation, crossover, and selection) are executed till one of the termination criteria is fulfilled. The first termination criterion is satisfied when the maximal number of heat load generations $NGH$ is reached. The second termination criterion is reached when the number of consecutive generations without improvement of the fitness exceeds its limit.

Implementation and parallelization

The algorithm is implemented in Python 3.8.2 using the library DEAP - Distributed Evolutionary Algorithms in Python [7]. Evolutionary algorithms are predestined for parallel computing as they work with populations of chromosomes which can be evaluated separately by distributing them among multiple processors. Therefore, all feasible GA chromosomes are distributed to multiple processors on which the heat loads are optimized using DE. The source code of the algorithm is published under an open-source license [8].

Conflict of Interest

The Authors confirm that there are no conflicts of interest.

Acknowledgments

This research project is financially supported by the Swiss Innovation Agency Innosuisse and is part of the Swiss Competence Center for Energy Research – Efficiency of Industrial Processes SCCER EIP. Further financial support is provided by the Lucerne University of Applied Sciences and Arts, Switzerland. This research is part of a collaboration between the Lucerne University of Applied Sciences and Arts and TU-Wien.

References

[1] K.C. Furman, N.V. Sahinidis, Computational complexity of heat exchanger network synthesis, Comput. Chem. Eng. 25 (9-10) (2001) 1371–1390, doi:10.1016/S0098-1354(01)00681-0.
[2] J.H. Holland, Adaptation in Natural and Artificial Systems, University Michigan press, Ann Arbor, MI, 1975.
[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publish Company, Inc., 1989.
[4] R. Storn, On the usage of differential evolution for function optimization, in: Biennial Conference of the North American Fuzzy Information Processing Society - NAFIPS, 1996, pp. 519–523, doi:10.1109/NAFIPS.1996.534789.
[5] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, doi:10.1023/A:1008202821328.

[6] J.A. Stampfli, B.H. Ong, D.G. Olsen, B. Wellig, R. Hofmann, Applied heat exchanger network retrofit for multi-period processes in industry: A hybrid evolutionary algorithm, Computers and Chemical Engineering 161 (2022) 107771, doi:10.1016/j.compchemeng.2022.107771.

[7] F.-A. Fortin, F.-M. De Rainville, U. Marc-André Gardner, M. Parizeau, C. Gagné, et al., DEAP: evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (2012) 2171–2175.

[8] Stampfli, J. A. (2021). J-A-St/moc_retrofit_ga_de: v1.0 (Version v1.0). 10.5281/zenodo.4441140