To appear in *Engineering Optimization*
Vol. 00, No. 00, Month 20XX, 1–17

Assessment of a Differential Evolution for Multi-objective Optimization in Natural Convection problem solved by a Local Meshless Method

Matjaž Depolli[a*] and Gregor Kosec[a]

[a] *Jožef Stefan institute, Jamova cesta 39, SI-1000 Ljubljana, Slovenia*;

(*11 March 2016*)

The performance of a Differential Evolution for Multi-objective Optimization (DEMO) in a non-linear coupled transport problem, solved by a Meshless Local Strong Form Method (MLSM), is assessed from different points of view. First, the behaviour of the optimization algorithm is tested for different scenarios, ranging from optimization of trivial diffusive transport to more complex non-linear natural convection problems. And second, a hybrid parallel implementation of both, the optimization and simulation codes, is introduced to optimize execution time, since such simulation based optimization might require vast amount of computational power. The goal of optimization is to partially cover the differentially heated cavity with non-permeable obstacles to maximally obstruct the flow with a minimal possible coverage. Different scenarios are taken into account to analyse the optimization performance. The results are presented in terms of temperature contours, velocity fields, Pareto fronts, optimizer convergence analyses, analyses of optimizer stability, and finally, parallel execution performance.

Keywords: diffusion; Navier Stokes; local meshless method; CFD; evolutionary strategy; parallel

1. Introduction

The most straightforward approach towards technological development is experimental fabrication and testing of the products. Such approach is, as a rule, expensive and time-consuming, and in many cases impossible to realize. The experiments are often also limited by the measurement equipment. The alternative is the use of appropriate computational modelling. The well tested and validated simulations can provide detailed insight into the tackled phenomena and can be easily coupled with optimization algorithms.

In this paper we focus on the numerical optimization of coupled transport phenomena described by three coupled Partial Differential Equations (PDEs) and a supporting constitutive equation. A momentum transport modelled with Navier-Stokes equation and coupled with a mass continuity equation form a fluid flow part of the model, which is additionally coupled with a heat transport, modelled with diffusion-convection equation. There are many natural and technological problems that can be tackled with similar diffusive-convective based models, e.g., weather dynamics, aerodynamics, solidification, semiconductor simulations Kosec and Trobec (2015), and many more.

The principal incitement in this work is the minimization of energy losses in a differentially-heated air-filled square cavity de Vahl Davis (1983) by means of obstructing

*Corresponding author. Email: matjaz.depolli@ijs.si natural convection flow. The cavity is differentially heated on two sides and thermally isolated on the other two sides. The differences in air density due to the temperature gradients drive the fluid flow into pronounced natural convection flow patterns. The energy transport over the domain, i.e., energy losses, is therefore not governed solely by diffusion but also convection. The most straightforward solution of the problem at hand would be to simply fill the whole domain with a good non-permeable insulator. However, our goal here is to minimize the energy losses and insulating material consumption, therefore we fill only portion of the domain with non-permeable obstacles to alter the air flow, and consequently minimize the energy losses due to convection. There are numerous examples of similar systems where one would be interested in such optimization, e.g., designing windows or other insulating elements for buildings, optimizing heat storage systems, optimizing heat distribution within rooms, etc.

The problem is naturally not solvable in a closed form and therefore a numerical approach is required. We use the Meshless Local Strong form Method (MLSM) Kosec et al. (2014), a strong form variant of meshless method Šterk and Trobec (2008), for spatial discretization of governing PDEs, explicit time stepping and artificial compressibility method for treating the pressure velocity coupling Malan and Lewis (2011). The simulator is coupled with an optimizer that implements the Asynchronous Master Slave Differential Evolution for Multi-objective Optimization (AMS-DEMO) algorithm Depolli, Trobec, and Filipič (2013), which is a parallel evolutionary algorithm Eiben and Smith (2003) for multi-objective optimization Zitzler and Thiele (1998); Coello, Lamont, and Veldhuizen (2006); Abraham, Jain, and Goldberg (2005) of real-valued functions. The coupling of optimizer and simulator is done through the cost function. At given parameters, i.e., positions of the obstacles, simulator computes steady-state temperature, pressure and velocity fields. The value of cost function - the total heat flux through the domain with specified obstacles - can be easily determined from the computed fields. This value is then passed on to the optimizer, which computes a new input parameter set for the simulator. The optimizer iterates as long as the optimization convergence criterion is not met or the number of performed iterations grows too large.

One of the important aspects of the problem is the execution time. The simulation time can be controlled through the complexity of the simulation, by setting the output accuracy through spatial and temporal resolution. However, the results have to be reasonable, therefore the problem cannot be computed with a handful of computational nodes and a few time steps, but more likely with thousands of nodes and thousands of time steps, resulting in simulation times measured in, at least, minutes. In addition, stochastic optimization, such as implemented by an evolutionary algorithm, typically requires vast number of iterations to converge, counted in thousands or millions. Soon, the computational cost becomes too high for practical use. Consequently, the efficiency of the computer implementation and execution is of a grave importance if we want to acquire adequate results in a reasonable time.

This paper presents a coupling of parallel optimization (AMS-DEMO) and parallel simulation (MLSM) that work together to exploit a parallel computer system with high efficiency. Shared-memory parallel simulator is coupled with a distributed evolutionary optimizer and executed on a cluster of multi-core computers.

Several different cavity setups are considered to evaluate both the simulator and the optimizer. The results are presented in terms of temperature contour plots, velocity profiles, analysis of heat losses, Pareto fronts of optimal solutions, convergence of optimal solutions, and sensitivity analysis of the optimizer and parallel execution performance.

2. Test case definition

The natural convection is modelled by three coupled PDEs. Diffusion equation for energy transport, Navier Stokes equation for momentum transport, and mass continuity equation. The Boussinesq approximation is used for coupling the heat and momentum transport. The model is a well-know fluid flow benchmark test in the literature usually referred to as de Vahl Davis test de Vahl Davis (1983). The model is defined by the following system of equations

$$\nabla \cdot \mathbf{v} = 0, \tag{1}$$

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \rho \nabla \cdot (\mathbf{v}\mathbf{v}) = -\nabla P + \nabla \cdot (\mu \nabla \mathbf{v}) + \mathbf{b}, \tag{2}$$

$$\rho \frac{\partial (c_p T)}{\partial t} + \rho \nabla \cdot (c_p T \mathbf{v}) = \nabla \cdot (\lambda \nabla T), \tag{3}$$

$$\mathbf{b} = \rho \left[1 - \beta_T (T - T_{\text{ref}})\right] \mathbf{g}, \tag{4}$$

where λ stands the thermal conductivity, $\mathbf{v}(u, v)$ for the velocity, t for the time, c_p for the specific heat, ρ for the density, P for the pressure, μ for the viscosity, $\mathbf{b}$ for the body force, T for the temperature, β_T for the thermal expansion coefficient, T_{ref} for the reference temperature and $\mathbf{g}$ for the gravitational acceleration. The problem is fully characterized by two dimensionless numbers: the Prandtl number ($\mathrm{Pr} = \mu c_p/\lambda$) and the Rayleigh number ($\mathrm{Ra}= |\mathbf{g}|\, \beta_T\, (\Delta T)\, \Omega^3 \rho^2 c_p/\lambda\mu$). In this paper a 2D quadratic air filled cavity ($\mathrm{Pr} = 0.71$) at different Rayleigh numbers is considered, where obstacles are used to alter the flow structure (Figure 1). Besides standard de Vahl Davis test we also consider a test case where horizontal walls are differentially heated and vertical ones isolated - basically de Vahl Davis test, rotated by $\pi/2$. For further discussions we will name the de Vahl Davis case a *horizontal case* and the rotated case a *vertical case*. The Obstacles are rectangular with edges parallel to the edges of the cavity, built of material with different properties than the fluid. Most importantly, the material is non-permeable and is a better thermal insulator. Therefore obstacles can be used to break the fluid flow structures as well as to act as an insulation. Thermal conductivity of the obstacles is set to 25% of the free media thermal conductivity.

2.1 *Numerical solution*

The presented transport requires numerical integration of governing PDEs to acquire solution. Since our goal is to construct an effective parallel implementation, the local numerical approach is preferred, therefore a local meshless principle Wang, Sadat, and Prax (2012); Kosec and Šarler (2008) for spatial discretization and explicit time stepping for temporal discretization is employed. The considered fields, namely velocity, pressure, and temperature, over a local subset of computational nodes, i.e., support domain, are approximated as

$$u(\mathbf{x}) = \sum_{i=1}^{m} p_i(\mathbf{x})\, a_i\,, \tag{5}$$

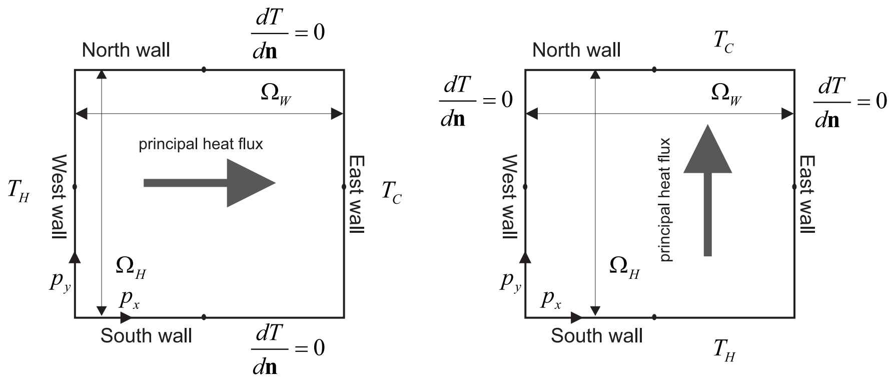


Figure 1.: Principal scheme of the problem, horizontal case (left) and vertical case (right).

where $a_i, p_i = [1, x, y, x^2, y^2, xy, ...]$ stand for approximation coefficients and monomial basis, respectively. The goal here is to solve system of second order PDEs, and to obtain non-trivial first and second derivatives a minimal basis of five monomials is used. Therefore to determine corresponding coefficients at least five support nodes are required. In such set-up, i.e. set-up with support domain size the same as the number of basis functions, the approximation coefficients can be determined exactly by solving the local system defined by one equation (5) for each support node

$$\vec{u} = \vec{A}\vec{\alpha}, \tag{6}$$

where $\vec{u} = (u(\vec{x}_1), ..., u(\vec{x}_{n_S}))$ stand for vector of support field values, $A_{i,j} = p_i(\vec{x}_j)$ for system matrix, $\vec{x}_j$ for position of j-th support node, and $\vec{\alpha} = (a_1, ...a_m)$ stands for vector of coefficients. Now, we can formulate all required operators $\big(\nabla = (\partial/\partial x, \partial/\partial y)$ and $\nabla^2 = \partial/\partial x^2 + \partial/\partial x^2\big)$ to solve governing system of equations

$$\frac{\partial}{\partial x^\epsilon} u(\mathbf{x}) = \sum_{j=1}^{n_S} \left(\sum_{i=1}^{m} \mathbf{A}_{i,j}^{-1} \frac{\partial}{\partial x^\epsilon} p_i(\mathbf{x}) \right) u(\vec{x}_j), \tag{7}$$

$$\nabla^2 u(\mathbf{x}) = \sum_{j=1}^{n_S} \left(\sum_{i=1}^{m} \mathbf{A}_{i,j}^{-1} \nabla^2 p_i(\mathbf{x}) \right) u(\vec{x}_j). \tag{8}$$

where $\epsilon = (x, y)$ denotes the coordinate. To simplify the notation, local shape functions are introduced

$$\chi_j^L(\vec{x}) = \sum_{i=1}^{m} \mathbf{A}_{i,j}^{-1} L p_i(\mathbf{x}_j), \tag{9}$$

where L stands for the general partial differential operator. Now, a general Operator L can be applied simply by multiplying the shape functions with values of the corresponding field in support domain nodes, i.e.

$$Lu(\mathbf{x}) = \sum_{j=1}^{n_S} \chi_j^L u_j. \tag{10}$$

All the information about the nodal topology and the differential operator is now stored in the shape functions. The presented formulation is convenient for implementation since most of the complex operations, i.e. finding support nodes and building shape functions, are computed in advance. In the main simulation, the pre-computed shape functions are then convoluted with the vector of field values in the support to evaluate the desired operator. The shape function construction has asymptotical complexity of $O\left(N_D n_S m^2\right)$, where N_D stands for total number of discretization nodes. In addition, the determination of support domain nodes also consumes some time, for example, if a kD-tree data structure is used, first the tree is built with $O\left(N_D \log N_D\right)$ and then additional $O\left(N_D\left(logN_D + n_S\right)\right)$ for collecting n_S supporting nodes. Two types of boundary conditions are considered in the governing system, namely Dirichlet and Neumann. Dirichlet are trivial to apply. The Neumann boundary conditions are computed as

$$\frac{v - \sum_{j=2}^{n_S} \chi_j^{\partial/\partial\epsilon} u(\vec{x}_j)}{\chi_1^{\partial/\partial\epsilon}} = u(\vec{x}_1), \tag{11}$$

where we assumed that $j = 1$ node is a boundary node and v stands for boundary value.

The well-known problem of solving fluid flow problems is a pressure-velocity coupling. There are different approaches towards coupling governing equations, namely continuity (1) and momentum (2) equations. In general, one solves a Poisson pressure or pressure correction equation Ferziger and Perić (2002). Here, we are interested only in the steady-state solution and also prefer the local approach, which can be easily parallelized. Therefore we employ the artificial Compressibility Method (ACM) Zienkiewicz, Taylor, and Zhu (2005). First, the velocity and temperature are computed from the previous time step (12,13). Second, the velocity is driven towards solenoidal field by correcting the pressure (14).

$$T_1 = T_0 + \frac{\Delta t}{\rho c_p}\left[\nabla\cdot(\lambda\nabla T_0) - \nabla\cdot(\rho c_p T_0 \mathbf{v}_0)\right] \tag{12}$$

$$\hat{\mathbf{v}}_1 = \mathbf{v}_0 + \frac{\Delta t}{\rho}\left[-\nabla P_0 + \nabla\cdot(\mu\nabla\mathbf{v}_0) + \mathbf{b}_0 - \nabla\cdot(\rho\mathbf{v}_0\mathbf{v}_0)\right]. \tag{13}$$

$$P_1 = P_0 - \varsigma\Delta t\nabla\hat{\mathbf{v}}, \tag{14}$$

where Δt and ς stand for the time step length and the relaxation parameter, respectively. Indices 0 and 1 stand for the current and the next time step, respectively. Note that no special boundary conditions for pressure are used, i.e., the pressure on boundaries is computed with the same approach as in the interior of the domain. When all the computations are done for all the computational nodes, the simulation proceeds to next time step, i.e., the values of the fields in the next time step override the values of the current time step. The simulation proceeds as long as the convergence criterion, i.e., the difference between $\mid T_1 - T_0 \mid$, is not below a threshold value in all nodes. A scheme of implementation is presented in Figure 2.

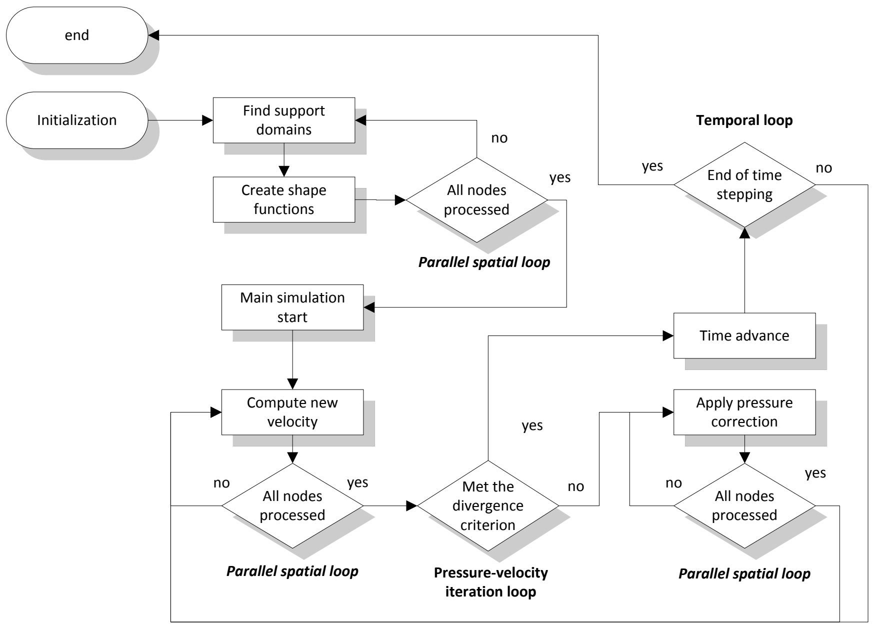


Figure 2.: Scheme of simulator implementation.

2.2 *Optimization*

The global optimization requires the use of stochastic optimization procedure. Furthermore, since we wish to minimize both the amount of material used for obstacles and the energy losses, we must employ a multi-objective optimization technique. In this work we use the AMS-DEMO algorithm, a parallel evolutionary algorithm for multi-objective optimization of real-valued functions. Evolutionary algorithms are a subgroup of stochastic optimization algorithms Burke and Kendall (2003) and can solve problems for which the analytical form of the cost function is unknown, but the function can be numerically evaluated for any given set of input parameters. In the presented problem, the numerical simulation serves as the cost function evaluator.

The description of optimization methodology starts with a subsection that focuses on the multi-objective optimization methodology, and continues with the subsection on the AMS-DEMO algorithms.

2.2.1 Multi-objective optimization

Multi-objective optimization problems are tasks that require optimizing a vector function:

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

where $\mathbf{x}$ is a vector of n decision variables defined over $\mathbb{R}$, and $\mathbf{y}$ is a vector of m objectives:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n),$$
$$\mathbf{y} = (y_1, y_2, \ldots, y_m).$$

There are two Euclidean spaces associated with multi-objective optimization. These are the n-dimensional *decision variable space* of solutions to the problem, and the m-dimensional *objective space* of their images under $\mathbf{f}$. Feasible solutions are vectors $\mathbf{x}$ in decision variable space that satisfy I inequality and J equality constraints

$$\begin{aligned} g_i(\mathbf{x}) &\geq 0,\ i = 1, 2, \dots, I, \\ h_i(\mathbf{x}) &= 0,\ j = 1, 2, \dots, J. \end{aligned}$$

The objective space is partially ordered according to the *Pareto dominance* relation Abraham, Jain, and Goldberg (2005). Given two objective vectors, **a** and **b**; **a** is said to *dominate* **b** iff **a** is not worse than **b** in all objectives and is better than **b** in at least one objective. Formally, assuming a minimization problem, this can be written as:

$$\begin{aligned} &\mathbf{a} \prec \mathbf{b} \text{ iff} \\ \forall k \in \{1, 2, \dots, m\} &: a_k \leq b_k \text{ and} \\ \exists l \in \{1, 2, \dots, m\} &: a_l < b_l. \end{aligned}$$

Note that for a pair of solutions **a** and **b**, there are three Pareto dominance relation combinations possible:

$$\begin{aligned} &\mathbf{a} \prec \mathbf{b} \text{ and } \mathbf{b} \not\prec \mathbf{a} \text{ or} \\ &\mathbf{a} \not\prec \mathbf{b} \text{ and } \mathbf{b} \prec \mathbf{a} \text{ or} \\ &\mathbf{a} \not\prec \mathbf{b} \text{ and } \mathbf{b} \not\prec \mathbf{a} \end{aligned}$$

Namely, either **a** dominates **b**, **b** dominates **a**, or neither dominates the other, thus making them not comparable. The solution to a multi-objective optimization problem, called the Pareto optimal set Abraham, Jain, and Goldberg (2005), is a set of feasible solutions, whose images in the objective space, called the Pareto front, are not comparable with each other and are not dominated by any other feasible solution. The Pareto optimal front forms a hyper surface in the objective space. The task of multi-objective optimization is to find a non-dominated set of solutions, representing an approximation for the Pareto front, rather than finding one absolutely best solution. This is to assist the user of multi-objective optimization in deciding on the final solution, using additional preferences.

2.2.2 AMS-DEMO

Differential Evolution for Multi-objective Optimization (DEMO) Robič and Filipič (2005) is an evolutionary strategy for solving multi-objective optimization problems Price, Storn, and Lampinen (2005). It is an iterative algorithm operating on a set of solutions called *population*. In each iteration, every solution from the population acts as a parent **p** to a newly created trial solution (also called candidate) **c**. To arrive at trial solutions, parents are modified by the application of differential mutation and uniform crossover. Differential mutation takes three or more members of the population $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, [\mathbf{x}_4 \dots \mathbf{x}_{n_p}] \in \mathbb{P}$, where n_p is the population size, to help construct a mutation vector **v** by vector addition and scalar multiplication. A common way of calculating the mutation vector, and also used here, is using the formula $\mathbf{v} = \mathbf{x}_1 + F \cdot (\mathbf{x}_2 - \mathbf{x}_3)$, where $F \in \mathbb{R}$ is a constant, most often from the interval $(0, 2]$. The mutation is followed by uniform crossover, which either takes the elements of the parent vector or the mutation vector, with a fixed probability P_c, creating a trial solution:

$$\begin{aligned} \forall i \in \{1, 2, \dots, n\} : \mathbf{c}_i = &\mathbf{p}_i \text{ with probability } 1 - P_c \\ &\mathbf{v}_i \text{ with probability } P_c \end{aligned}$$

Selection is then performed, by testing Pareto dominance between the trial solutions are their parents in a pair-wise manner. The dominated solutions are discarder while the dominating solutions form the population of the next iteration. In case neither solution dominates the other, both of the involved solutions are added to the population. Since more than one solution is sometimes added to the population, population size increases with time. To oppose this increase, the population is reduced back to its original size at the end of every iteration, by applying the non-dominated sorting and the crowding distance metric from the NSGA-II algorithm Deb et al. (2002) to discard the worst solutions. The algorithm finishes either after a fixed number of performed cost function evaluations, after a solution of predefined quality is found, after the solutions have converged with a satisfyingly high confidence or after a similar terminating condition is met.

Asynchronous Master-Slave DEMO (AMS-DEMO) is a parallel extension of DEMO. It is able to exploit heterogeneous computer architectures with large number of compute nodes, while retaining very good parallel efficiency Depolli, Trobec, and Filipič (2013).

3. Results

Four problem cases of varying difficulty are devised to test the performance of the coupling of the optimizer and simulator. Within each case, all the fluxes are normalized relative to the flux through an empty domain governed solely by diffusion Šterk and Trobec (2008), thus the solutions for vertical cases can be compared across cases but not to the solutions for the horizontal case, and vice-versa

3.1 *Experimental setup*

Experiments are executed on a homogeneous cluster of 20 computers interconnected with Gigabit Ethernet network. The heart of each computer is single quad-core processor Intel Xeon E5520. The execution of simulation based optimization is parallelized on two levels. First, the simulation exploits multi-core architecture of cluster nodes with shared-memory parallelism, using OpenMP. Second, the optimization (AMS-DEMO) makes use of all the available cluster processors by distributing separate simulations among the nodes of the cluster, using MPI. Both simulator and optimizer executables are written in C++, compiled with GCC 4.8 with optimizations enabled by `-O3` switch. OpenMP is built into the compiler while the MPI is implemented through the Open MPI library. Optimizer and simulator communicate via file system, bash scripts, and Ubuntu 12.04 operating system.

All simulations were executed with the following parameters: 81×81 uniformly distributed nodes on the edges and inside a cavity of dimensionless size 1×1, MLSM time step $dt = 2.5 \cdot 10^{-5}$, maximal allowed dimensionless time $t_{max} = 20$ and steady state criteria $\mid T_1 - T_0 \mid < 10^{-7}$.

3.2 *Case 1: fill whole domain with obstacles*

For the first case, a simplified model without fluid flow is used. The optimizer is given a differentially heated square cavity and the option to place one or several pieces of better insulator (obstacles) in the domain. Obstacle positions are unconstrained, while their size is constrained upwards by the size of the cavity divided by the number of obstacles. Since the solution of diffusion does not comprise non-linear responses, it is relatively easy to understand, numerically solve, and - most importantly - to optimize. The diffusion problem can be also understood as building an insulator out of two solid materials with different thermal properties.

Since the obstacle material is a better insulator than the cavity material, the optimal result is achieved when the obstacles fill the whole cavity. Although the optimal solution is obvious to human, the same does not hold for the stochastic optimization logic and this test serves as a benchmark of the optimization procedure. In other words, we have a "closed form" solution against which we can compare the solutions obtained by optimization procedure, i.e., we can assess the performance of the optimization. This task is further divided into four increasingly difficult subtasks, each given a different number and size of obstacles to work with, listed in Table 1. For each subtask, the single optimum solution is to set obstacle sizes to their maximum and to arrange them in a grid. The best solutions (out of 10 runs) are plotted in Figure 3.

With increasing complexity of the optimization problem, the optimization finds less optimal solutions and is unable to cover the cavity completely. This remains so even though the increase in number of parameters is always countered by the increased population size. The optimization procedure can place one obstacle almost optimally and it performs very well with four obstacles. For higher number of obstacles, i.e., 9 and 16, we can notice a slight drop in solution quality. Although optimization settings could be optimized to produce better results, the fine tuning of several parameters of optimization is beyond the scope of this work. It should also be be noted that better results could be found on the account of running optimization for a longer time.

The convergence of solutions is shown in Figure 4, and is slower for the sub-cases with more obstacles than for the sub-cases with less obstacles. Much larger populations would likely help in obtaining better results in terms of heat flux but for the price of even higher number of performed simulations for a single optimization run.

3.3 *Case 2: Efficiently obstruct the fluid inside a cavity*

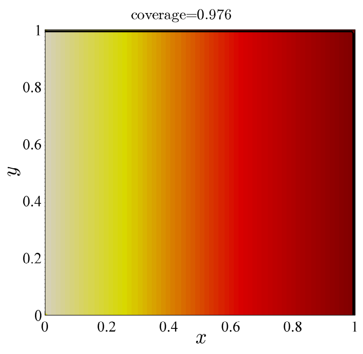
For the second experiment, a two-objective problem with conflicting criteria is tackled. The cavity is filled with fluid, which can be obstructed by non-permeable obstacles. The optimizer is given the freedom of positioning 10 obstacles within the cavity with no restriction on their sizes. Six sub-cases are prepared (see Table 2), where the Rayleigh number and heat flow direction are varied. This time we search for the best insulator with minimal material consumed, therefore the solutions are measured using two criteria for minimization: the area covered by obstacles and the heat flux through the cavity.

A few examples of dominant solutions are presented in Figure 5. The plots are arranged in rows, one row for each sub-case. Within rows, seven trade-off solutions from the Pareto front are plotted, from the one with the minimal area coverage and the greatest flux on the left to the one with the maximal area coverage and the smallest flux on the right.

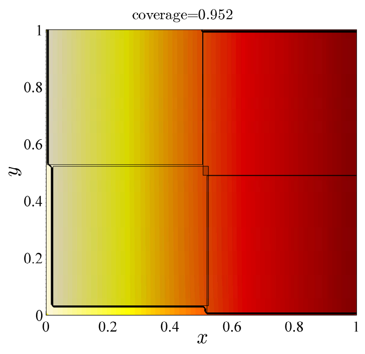
From the figure it follows that the optimization favors long and thin obstacles placed either vertically for limiting horizontal flux or horizontally for limiting vertical flux while keeping coverage low. Note that the heat flux itself is governed by the convective air

Table 1.: Settings of the optimizer

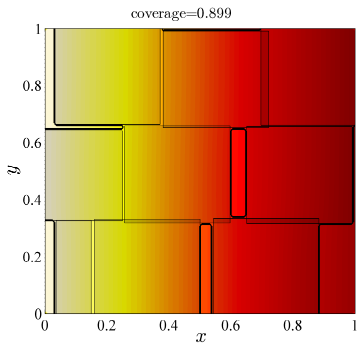
	Sub-cases			
Number of obstacles	1	4	9	16
Obstacle max size	1 x 1	1/2 x 1/2	1/3 x 1/3	1/4 x 1/4
Number of cost function parameters	4	16	36	64
Population size	20	30	40	50
Number of simulations	2000	6000	12000	18000



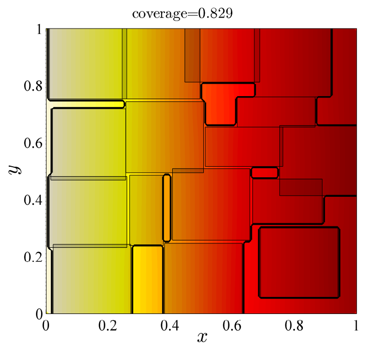
(a) 1 obstacle



(b) 4 obstacles



(c) 9 obstacles



(d) 16 obstacles

Figure 3.: Temperature contour plot and obstacle positions for Case 1.

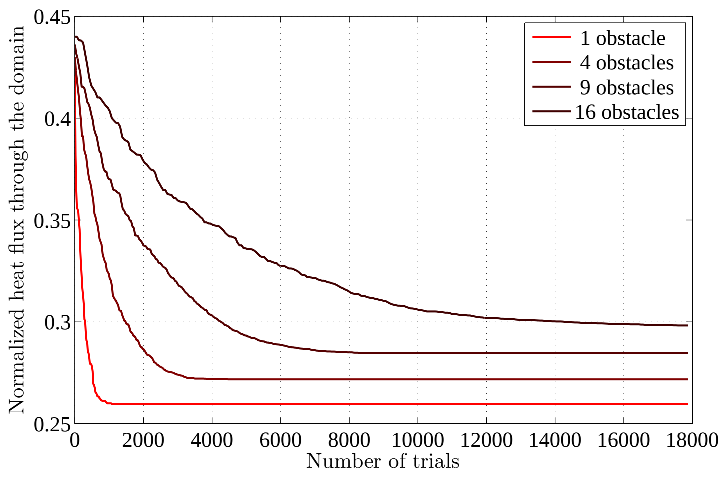


Figure 4.: Convergence of the optimal solution expressed by the heat flux as a function of the number of performed simulations. Statistics are made over 10 repetitions of the optimization task.

flow, which always follows circular patterns, and thus the direction of obstacles is not directly implied by the heating direction. To limit flux even more, coverage is sacrificed and obstacles get thicker until they fill the cavity almost completely. Usage of several obstacles is less pronounced, in most cases only one or two obstacles are used. To get the full picture of how the results cover the flux and coverage trade-off, the Pareto optimal fronts are plotted on Figure 6.

Table 2.: Settings of the optimizer

	Sub-cases					
Heat flow direction	Horizontal			Vertical		
Rayleigh number	10^4	10^5	10^6	10^4	10^5	10^6
Number of obstacles	10					
Obstacle max size	1 x 1					
Number of parameters	40					
Population size	50					
Number of simulations	5000					

For the lower two Rayleigh numbers (Ra) - which should translate to an easier problem - Pareto front shows seemingly even distribution of solutions with variable trade-off between the obstacle coverage or the heat flux optimization. For the Ra= 10^6, however, the solutions are clearly not evenly distributed, which hints at an incomplete convergence of the algorithm due to the more difficult problem.

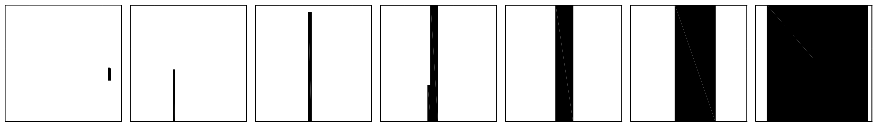
For this two-objective case the convergence of results is checked via hypervolume indicator Zitzler and Thiele (1998). Hypervolume indicator transforms a set of objective vectors into a single number, which in essence summarizes the coverage and distribution of the set of solutions in the objective space. It is a dimensionless number on the interval [0, 4] for two-objective cases, with higher numbers indicating more optimal solutions. Using hypervolume indicator, we show the convergence of solutions for Case 2 in Figure 7. To emphasize the different convergence rates we also summarize the number of performed simulations before the front of solutions converged in Table 3. We set the criterion for convergence to a hypervolume indicator change of less than 0.1 % per 100 simulations; but the results are robust to small changes in the criterion - the relative differences between sub-cases remain similar. From the figure and the table it is clear that the convergence is still incomplete for the case with horizontal flow and Ra=10^6, after 5000 evaluations.

Table 3.: Convergence of solutions in sub-cases expressed as the number of performed simulations before the convergence criterion is met.

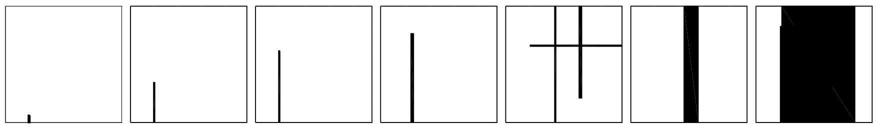
		Flow direction	
		Horizontal	Vertical
	10^4	1850	1650
Rayleigh number	10^5	1800	1350
	10^6	5000+	4650

3.4 *Parameter sensitivity of the optimizer*

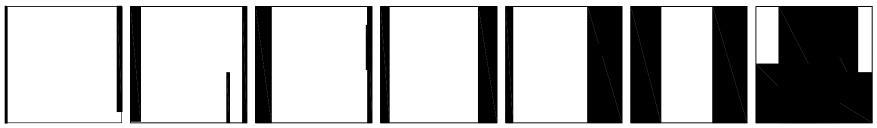
There are several parameters to the optimizer and they are quite difficult to set up in an optimal manner. To make things worse, there is no known mapping between the problem definition and the optimal parameter set for the optimizer. Therefore, the optimal parameter set cannot be determined prior to the actual optimization runs. Fortunately, though, optimizer will work well enough with almost any combination of parameter values, except for the most extreme ones. To shed some light onto the problem of parameter selection, a scan over the parameter space is performed. The task of filling the whole



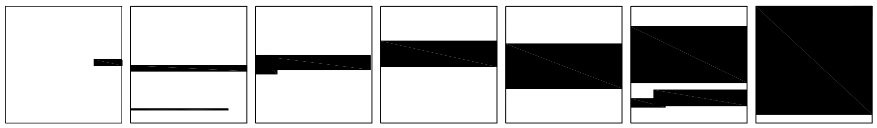
(a) Horizontal, Ra=10^4



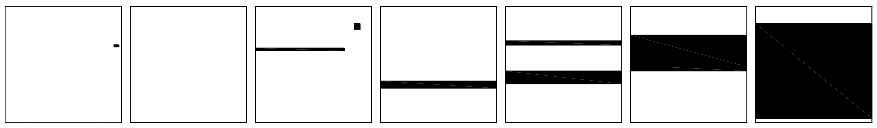
(b) Horizontal, Ra=10^5



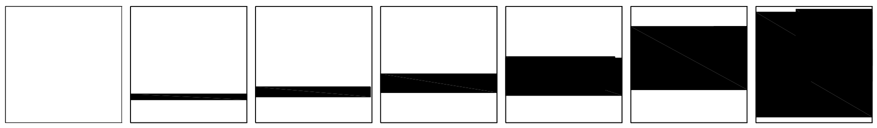
(c) Horizontal, Ra=10^6



(d) Vertical, Ra=10^4



(e) Vertical, Ra=10^5



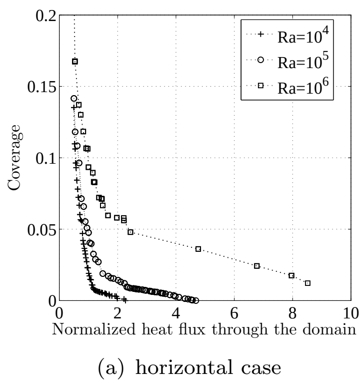
(f) Vertical, Ra=10^6

Figure 5.: Obstacle positions for Case 2. Plots (a) through (c) represent the horizontal flux sub-case for different trade-offs between the coverage and flux optimization, while plots (d) through (f) do the same for the vertical flux sub-cases.

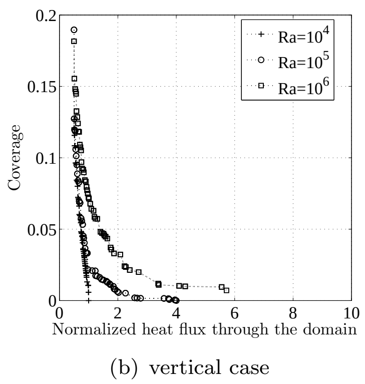
cavity with four obstacles is selected as the test case for optimizer. Note that the results would differ if either a different number of obstacles was selected or the task of optimization was different.

The three most important parameters of the optimizer are taken into consideration for the scan: crossover probability c_p, scaling factor F, and population size n_p. For both c_p and F, the input set of values is set to $[0.1, 0.2, ..., 0.9]$, while for n_p, the input set of values is $[10, 20, ..., 50]$. Each of the parameters is varied across its input set of values individually with five optimization runs executed for each value. The convergence of the mean solution as a function of the number of performed simulations is used to show the difference between the parameter selections.

Firstly, the results of varying c_p are shown on Figure 8. c_p is a parameter that has the greatest influence on the convergence rate of the solutions and on the quality of the best solutions found. Low values lead to the best results (lowest heat flux), with minimum



(a) horizontal case



(b) vertical case

Figure 6.: Pareto fronts discovered by the optimization.

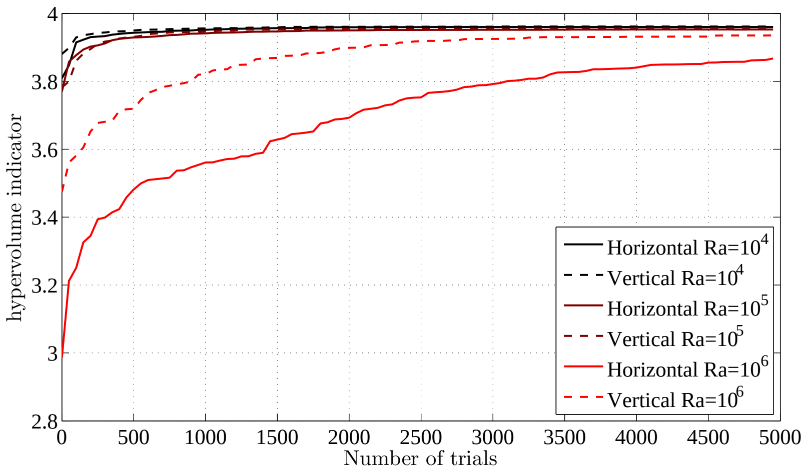


Figure 7.: Convergence of the optimal solution expressed by the hypervolume indicator as a function of the number of performed simulations. Statistics are made over 10 repetitions of the optimization.

being around 0.3. Only the highest values, namely 0.8 and 0.9 are extremely bad choices.

Secondly, the results of varying F are shown on Figure 9. In contrast to the previous figure, this one is much less dynamic, which implies that the optimizer is far more robust to selection of F than it is to the selection of c_p. There is no clear pattern in the figure, which is most likely caused by a very low number of repetitions. Given more repetitions, one value might peak as the best, and less likely, there could be more than one near-optimal peak hiding in the value of F. However, the results are clear enough in showing that just about any value of F between 0.2 and 0.9 works well.

Lastly, the results of varying n_p are shown in Figure 10. Population size is known to have a pronounced effect on the evolutionary algorithms, and AMS-DEMO is no exception. Although almost any value works, low values represent a trade-off of faster convergence rate for the price of finding less optimal solutions. Raising the value moves the trade-off towards slower convergence rate but convergence to better solutions. There is of course a sensible limit to the value of n_p - the value at which the absolute optimal solution or a solution below a certain cost threshold is found, depending on how the optimizer is used. Raising n_p above this limit would only make the algorithm converge slower, since it makes it more random. The final limit of n_p is the total number of simulations performed, at which the algorithm degenerates to the ordinary Monte-Carlo

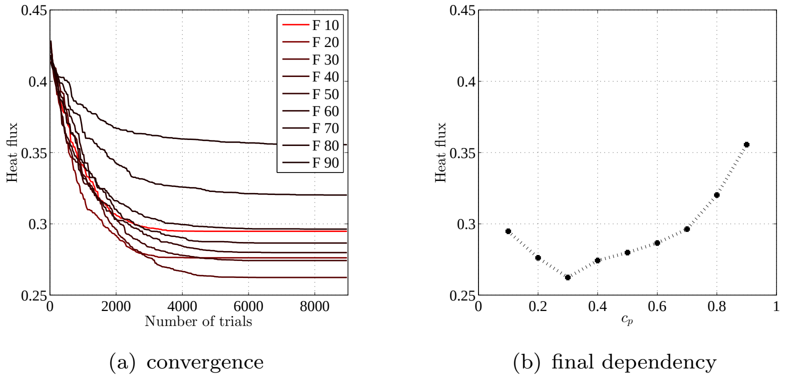


Figure 8.: Convergence of the optimal solution for varying crossover probability c_p.

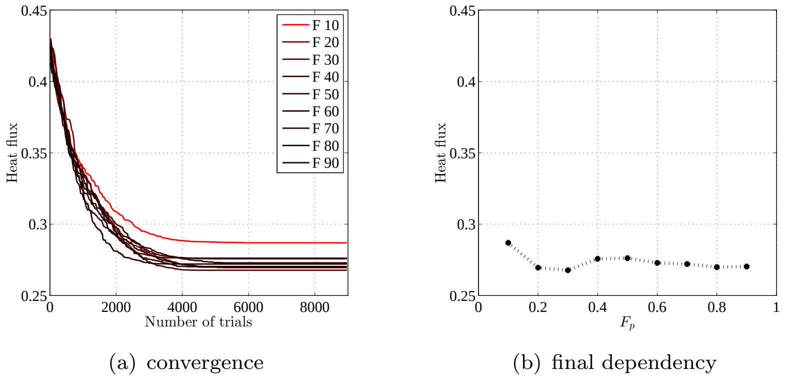


Figure 9.: Convergence of the optimal solution for varying scaling factor F.

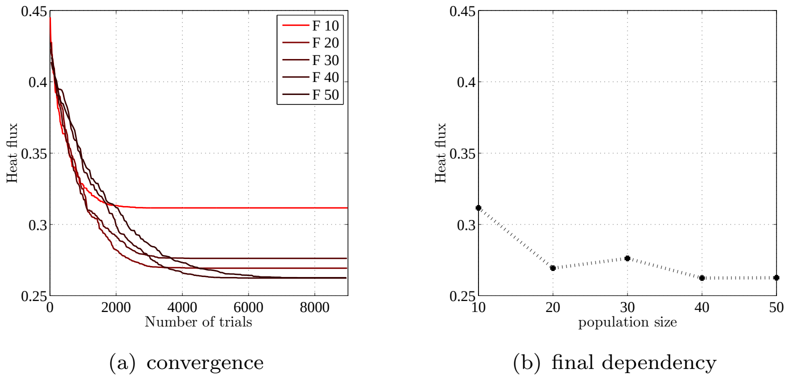


Figure 10.: Convergence of the optimal solution for varying population size.

optimization. The experimental results confirm all the above rules, with the exception of $n_p = 20$ being worse than $n_p = 30$, which is unexpected. Again, the divergence from the rules is almost certainly only the result of the noise caused by the low number of optimization repetitions.

3.5 *Parallel speedup and efficiency*

We are also interested in the efficiency of the parallelization, which we measure through parallel speedup. Since solver and optimizer are parallelized separately, the parallel

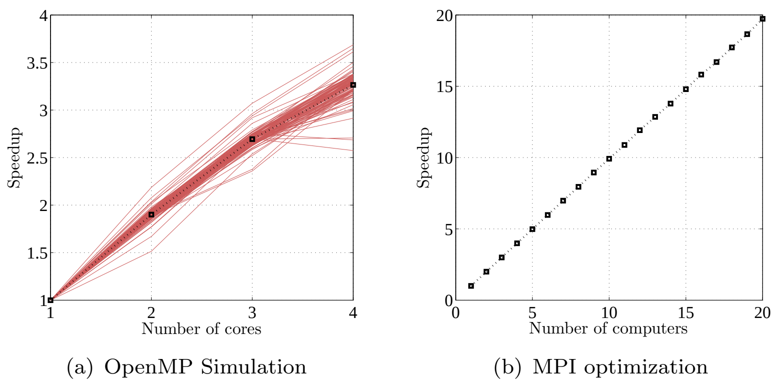


Figure 11.: Speedup of the simulation and the optimization execution.

speedup is also two-part. The first part of speedup is due to the execution of the parallel simulator on multiple cores of a single cluster node, while the second part is due to the distributed execution of simulations on the cluster. Speedup s is computed as:

$$s = \frac{t_1}{t_n},$$

where t_1 is the execution time on 1 unit and t_n is the execution time on n units. The unit is either core, for the simulation speedup, or computer, for the optimizer speedup. The two parts of speedup are tested in separate experiments and the total speedup is then calculated.

In the first experiment, a robust speedup measurement of the simulator is obtained, rooted on the experiments made so far. One hundred parameter sets are randomly taken from one of the performed optimization runs of each case, and used as input for the speedup experiment. This experiment comprises simulator executions that are set to execute on 1-4 cores of a single computer with each run repeated on the one hundred inputs. Since results do not differ significantly between the cases, only the joined results are shown on the left of Figure 11. Speedups are calculated from total execution times, which include serial pre-processing, post-processing, input and output operations. Simulator is capable of much higher speedups than shown here, especially when more complex, in terms of spatial resolution, simulations take place, reducing the relative ratio of the serial algorithm portion over the parallel portion Kosec et al. (2014).

In the second experiment, AMS-DEMO speedup is estimated. AMS-DEMO attempts to minimize computer idle time by allowing a non-deterministic execution, causing the convergence of solutions to differ for different number of computers used. The detailed statistical analysis a large number of runs of AMS-DEMO has been performed in Depolli, Trobec, and Filipič (2013), with the main finding that no statistically significant differences are found in the convergence rate of solutions produced by different runs as long as the number of processing units is lower than the population size. A justified simplification can thus be made to ease the speedup measurements for the number of processing units less than the population size. For each tested number of processing units – in our case cluster nodes, AMS-DEMO is left to run until a predefined number of simulations is executed. A modification of Case 2 is devised, set to stop after 1000 simulations to shorten the execution time, while keeping the overhead of the sequential parts of the algorithm similarly low, as it would be in a longer run. This experiment depends very little on the choice of case and other details of the simulation and is therefore performed only once. Although the experiment respects the rule of running on less computers than the population size, the calculated speedup is called *weak speedup*, to account for the fact that the solutions of different runs are not completely equivalent. The right part of Figure 11 shows the weak speedup calculated as the fraction of the total execution time on n computers relative to execution time on a single computer, with input, output, MPI environment setup and other serial overheads included. To eliminate noise introduced by the large variation in simulation times (standard deviation is 35% of the mean simulation time), the individual run times were normalized by the mean simulation time of the run. The result is a near-linear weak-speedup of the optimizer that scales well over the tested number of computers.

Finally, the two independently measured speedups are multiplied to get the theoretical speedup on 80 cores relative to a single core. This is done by taking the speedup mean value at 4 cores from and multiplying it by the speedup at 20 computers (both are available on Figure 11). The total speedup is thus approximately $3.25 \times 19.73 = 64.12$.

4. Discussion and conclusions

The paper is focused on displaying the potency of the simulation-based optimization. The presented case of obstructing air-flow with simple obstacles shows interesting arrangements resulting in optimal obstruction that would be hard to predict without the synergy between the numerical simulator and the multi-objective optimization. The test case could be extended in various areas of design: insulation, large living and working spaces, air conditioning, heat storage and heat engines.

Furthermore, a the simulation-based optimization is executed in a parallel manner on a computer cluster. Although a modestly sized cluster is used, the parallel optimization can be executed on a much larger number of computers, and is not limited by the population size Depolli, Trobec, and Filipič (2013). Simulation exploits the shared-memory model of a multi-core computer, and is also capable of switching to GPUs Kosec and Zinterhof (2013) or other accelerator boards that proliferate in the modern high performance computing hardware. The combination of the two is efficient at utilizing modern hardware resources and providing a tool for handling physics-based optimization problems of much larger scale.

5. Acknowledgements

We acknowledge the financial support from the Slovenian Research Agency under the programme group P2-0095.

References

Abraham, A., L. Jain, and R. Goldberg, eds . 2005. *Evolutionary Multiobjective Optimization*. London: Springer-Verlag.

Burke, Edmunt K., and Graham Kendall, eds . 2003. *Introduction to Stochastic Search and Optimization*. Hoboken: John Wiley & Sons.

Coello, Carlos A. Coello, Gary B. Lamont, and David A. Van Veldhuizen. 2006. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

de Vahl Davis, G. 1983. "Natural convection of air in a square cavity: a bench mark numerical solution." *International Journal for Numerical Methods in Fluids* 3: 249–264.

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

Depolli, Matjaž, Roman Trobec, and Bogdan Filipič. 2013. "Asynchronous master-slave parallelization of differential evolution for multiobjective optimization." *Evolutionary Computation* 21 (2): 261–291.

Eiben, Agoston E., and J. E. Smith. 2003. *Introduction to Evolutionary Computing*. Berlin: Springer-Verlag.

Ferziger, J. H., and M. Perić. 2002. *Computational Methods for Fluid Dynamics*. Berlin: Springer.

Kosec, Gregor, Matjaž Depolli, Aleksandra Rashkovska, and Roman Trobec. 2014. "Super linear speedup in a local parallel meshless solution of thermo-fluid problems." *Computers & Structures* 133: 30–38.

Kosec, G., and R. Trobec. 2015. "Simulation of semiconductor devices with a local numerical approach." *Engineering Analysis with Boundary Elements* 50: 69–75.

Kosec, G., and B. Šarler. 2008. "Solution of thermo-fluid problems by collocation with local pressure correction." *International Journal of Numerical Methods for Heat & Fluid Flow* 18: 868–882.

Kosec, G., and P. Zinterhof. 2013. "Local strong form meshless method on multiple Graphics Processing Units." *CMES: Computer Modeling in Engineering and Sciences* 91 (5): 377–396.

Malan, A. G., and R. W. Lewis. 2011. "An artificial compressibility CBS method for modelling heat transfer and fluid flow in heterogeneous porous materials." *International Journal for Numerical Methods in Engineering* 87: 412–423.

Price, K., R. M. Storn, and J. A. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Berlin: Springer-Verlag.

Robič, Tea, and Bogdan Filipič. 2005. "DEMO: Differential Evolution for Multiobjective Optimization." In *Proceedings of the Third Conference on Evolutionary Multi-Criterion Optimization - EMO 2005,* Vol. 3410 of *Lecture Notes in Computer Science*520–533. Springer.

Šterk, M., and R. Trobec. 2008. "Meshless solution of a diffusion equation with parameter optimization and error analysis." *Engineering Analysis with Boundary Elements* 32 (7): 567–577.

Wang, C. A., H. Sadat, and C. Prax. 2012. "A new meshless approach for three dimensional fluid flow and related heat transfer problems." *Computers and Fluids* 69: 136–146.

Zienkiewicz, O. C., R. L. Taylor, and J. Z. Zhu. 2005. *The Finite Element Method: Its Basis and Fundamentals*. Oxford: Elsevier.

Zitzler, Eckart, and Lothar Thiele. 1998. "Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study." In *Proceedings of the Fifth Conference on Parallel Problem Solving from Nature – PPSN V,* 292–301. Heidelberg: Springer.