Engineering Optimization

ISSN: 0305-215X (Print) 1029-0273 (Online) Journal homepage: http://www.tandfonline.com/loi/geno20

Assessment of differential evolution for multi-objective optimization in a natural convection problem solved by a local meshless method

Matjaž Depolli & Gregor Kosec

To cite this article: Matjaž Depolli & Gregor Kosec (2016): Assessment of differential evolution for multi-objective optimization in a natural convection problem solved by a local meshless method, Engineering Optimization, DOI: 10.1080/0305215X.2016.1197686

To link to this article: http://dx.doi.org/10.1080/0305215X.2016.1197686

Published online: 15 Jul 2016.

ENGINEERING OPTIMIZATION, 2016
http://dx.doi.org/10.1080/0305215X.2016.1197686

Taylor & Francis
Taylor & Francis Group

Assessment of differential evolution for multi-objective optimization in a natural convection problem solved by a local meshless method

Matjaž Depolli and Gregor Kosec

Jožef Stefan institute, Ljubljana, Slovenia

ABSTRACT
The performance of Differential Evolution for Multi-objective Optimization (DEMO) in a nonlinear 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 nonlinear natural convection problems. 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 a vast amount of computational power. The goal of optimization is partially to cover the differentially heated cavity with non-permeable obstacles so as maximally to obstruct the flow with minimal possible coverage. Different scenarios are taken into account to analyse the optimization performance. 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.

ARTICLE HISTORY
Received 11 March 2016
Accepted 27 May 2016

KEYWORDS
Diffusion; Navier Stokes; local meshless method; CFD; evolutionary strategy; parallel

1. Introduction

The most straightforward approach to technological development is experimental fabrication and testing of the products. Such an 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. Well tested and validated simulations can provide detailed insight into the phenomena under investigation and can be easily coupled with optimization algorithms.

This paper is focused on the numerical optimization of coupled transport phenomena described by three coupled Partial Differential Equations (PDEs) and a supporting constitutive equation. Momentum transport modelled with a Navier–Stokes equation and coupled with a mass continuity equation form the fluid flow part of the model, which is additionally coupled with heat transport, modelled by a 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 motivation of this work is the minimization of energy losses in a differentially-heated air-filled square cavity (de Vahl Davis 1983) by means of obstructing natural convection flow. The

CONTACT Matjaž Depolli matjaz.depolli@ijs.si

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 simply to fill the whole domain with a good non-permeable insulator. However, the objectives are to minimize both the energy losses and the insulating material consumption; therefore, only a portion of the domain is filled with non-permeable obstacles to change 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. The Meshless Local Strong form Method (MLSM) (Kosec *et al.* 2014), a strong form variant of the meshless method (Šterk and Trobec 2008), is used for spatial discretization of governing PDEs, explicit time stepping, while the artificial compressibility method is used 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 2007; Abraham, Jain and Goldberg 2005) of real-valued functions. The coupling of the optimizer and the simulator is done through the cost function. With given parameters, *i.e.* positions of the obstacles, the simulator computes the steady-state temperature, pressure and velocity fields. The value of the 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 iterations performed 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 a 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 computer implementation and execution is of a grave importance when one desires to acquire adequate results in a reasonable time frame.

This paper presents a coupling of parallel optimization (AMS-DEMO) and parallel simulation (MLSM), which work together to exploit a parallel computer system with high efficiency. A shared-memory parallel simulator is coupled with a distributed evolutionary optimizer, which are 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.

1.1. Related work

Both geometry optimization and solving fluid-flow problems are well-researched topics. It is not surprising that the combination of the two is a topic of vast interest too. This combination fits well within a class of problems described by *topology optimization* (Sigmund and Maute 2013).

Topology optimization is an approach that optimizes material layout or geometry under a given set of constraints and boundary conditions, to meet predefined performance or design goals.

Fluid-flow and optimization problems similar to this work appear in many fields of interest. Machinery is often a centre of research, such as in case of Sterling engine optimization (Toghyani, Kasaeian and Ahmadi 2014). In that work, an evolutionary algorithm was used for the simulation-based optimization of a heat engine. Two objectives were optimized via four control parameters of the simulator.

On a smaller scale, micro-machines that work with or within a fluid medium have been of interest lately. The geometry of micro heat exchangers was optimized in Foli *et al.* (2006), using an analytical technique for a simple optimization case and multi-objective evolutionary optimization for a more profound optimization. Significant gains were discovered in heat exchanger performance and proved the potential of evolutionary algorithms for optimization of the non-trivial relationship between geometry and performance of micro heat exchangers. In another research on a similar size scale, magnetically propelled microswimmers were optimized in Keaveny, Walker and Shelley (2013). A steepest descend optimization was performed on microswimmer shapes, which was enabled by the availability of several good initial solutions and a well designed shape derivation technique.

A numerical optimization technique—*Method of Moving Asymptotes* (MMA)—is often used in topology optimization. For example, MMA was employed with great success in optimizing (both single- and multi-objectively) several geometries composed of fluid and solid subdomains (Marck, Nemer and Harion 2013). The finite volume method was used for the simulator, and the discrete adjoint approach for sensitivity analysis, which was used to estimate the derivatives of given solution design parameters required by the numerical optimization methodology.

Topology optimization and other simulation-based optimizations are recognized as very hard to solve because of their time complexity; therefore, parallel implementations have already been considered. A parallel framework in C++ was presented in Aage and Lazarov (2013), based on any solver that works with linear or nonlinear sparse systems, and an MMA optimizer. This work recognizes the need to parallelize the whole simulation-based optimization framework to be able to achieve decent speedups when the number of processing units reaches several hundred.

2. Test case definition

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

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

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

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

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

where λ stands for the thermal conductivity, $\vec{v}(u,v)$ 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, $\vec{b}$ for the body force, T for the temperature, β_T for the thermal expansion coefficient, T_{ref} for the reference temperature, and $\vec{g}$ 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 ($\mathrm{Ra} = |g| \beta_T \, (\Delta T) \, \Omega^3 \rho^2 c_p/\lambda\mu$). In this paper, a 2D quadratic air-filled cavity ($\mathrm{Pr} = 0.71$) at different Rayleigh numbers is considered, where obstacles

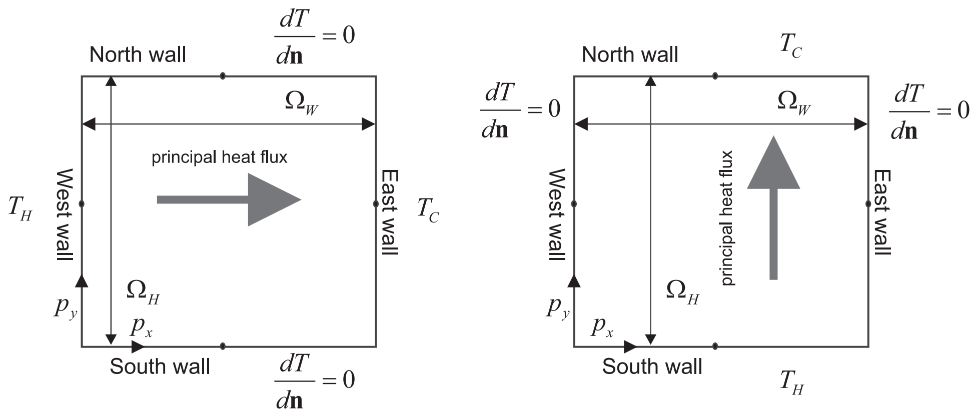


Figure 1. Principal scheme of the problem, horizontal case (left) and vertical case (right).

are used to alter the flow structure (Figure 1). Besides the standard de Vahl Davis test, a test case where the horizontal walls are differentially heated and the vertical ones isolated has also been considered—basically, a de Vahl Davis test rotated by $\pi/2$. For further discussion herein, the de Vahl Davis case will be called the *horizontal case* and the rotated case will be called the *vertical case*. The obstacles are rectangular with edges parallel to the edges of the cavity, built of material having properties different from those of 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. The 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 the governing PDEs to acquire solution. Since the 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 are employed. The considered fields, namely velocity, pressure, and temperature, over a local subset of computational nodes, *i.e.* the support domain, are approximated as

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

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

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

where $\vec{u} = (u(\vec{x}_1), \ldots, u(\vec{x}_m))$ stands for the vector of support field values, $A_{i,j} = p_i(\vec{x}_j)$ for the system matrix, $\vec{x}_j$ for the position of the jth support node, and $\vec{\alpha} = (a_1, \ldots, a_m)$ stands for the vector of coefficients. Now, all the required operators can be formulated $\left(\nabla = (\partial/\partial x, \partial/\partial y)\right.$ and $\nabla^2 = \partial/\partial x^2 + \partial/\partial y^2)$ to solve the governing system of equations

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

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

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

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

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

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

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 $\mathcal{O}\left(\mathcal{N}_D \Updownarrow^{\ni}\right)$, 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 $\mathcal{O}\left(\mathcal{N}_D \log \mathcal{N}_D\right)$ and then an additional $\mathcal{O}\left(\mathcal{N}_D \left(\log \mathcal{N}_D + \Updownarrow\right)\right)$ for collecting m supporting nodes. Two types of boundary conditions are considered in the governing system, namely Dirichlet and Neumann. The Dirichlet are trivial to apply. The Neumann boundary conditions are computed as

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

where $j = 1$ 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 (Equation 1) and momentum (Equation 2) equations. In general, one solves a Poisson pressure or a pressure correction equation (Ferziger and Perić 2002). Here, only the steady-state solution is of interest, and the local approach is preferred, which can be easily parallelized. Therefore the Artificial Compressibility Method (ACM) (Zienkiewicz, Taylor and Zhu 2005) is employed. First, the velocity and temperature are computed from the previous time step (Equations 12 and 13). Second, the velocity is driven towards a solenoidal field by correcting the pressure equation (14),

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

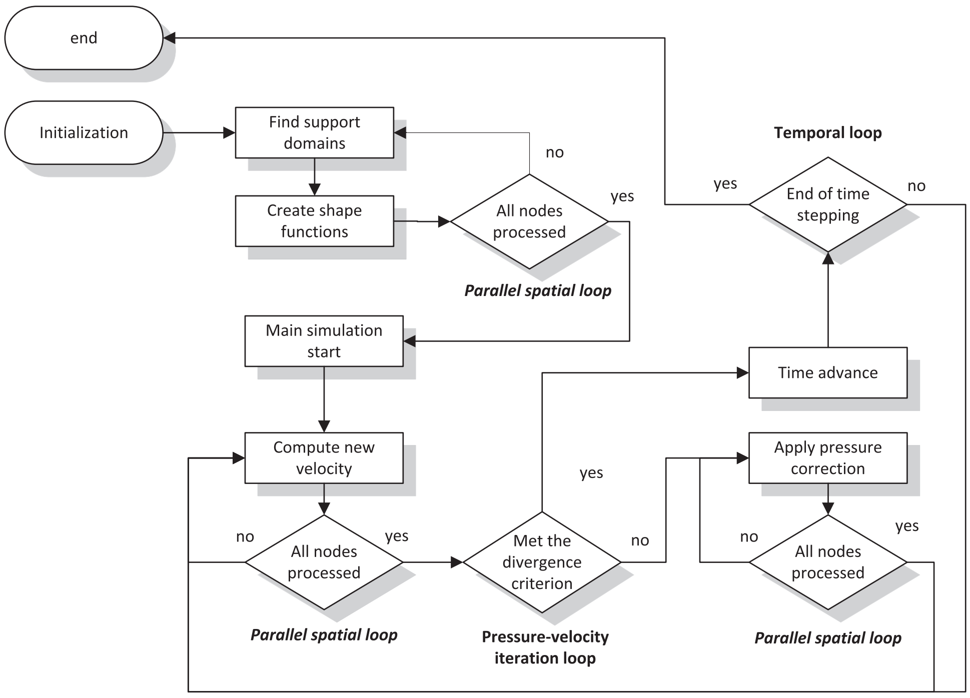


Figure 2. Scheme of simulator implementation.

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

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

where Δt and ς stand for the time step 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 have been done for all the computational nodes, the simulation proceeds to the 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 $|T_1 - T_0|$, is not below a threshold value in all nodes. A scheme of implementation is presented in Figure 2.

2.2. Optimization

The global optimization requires the use of a stochastic optimization procedure. Furthermore, since both the amount of material used for obstacles and the energy losses are to be minimized, a multi-objective optimization technique has to be employed. In this work the AMS-DEMO algorithm is used, a parallel evolutionary algorithm for the 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 a subsection on the AMS-DEMO algorithms.

2.2.1. Multi-objective optimization

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

$$\mathbf{y} = \mathbf{f}(\mathbf{x}), \tag{15}$$

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:

$$g_i(\mathbf{x}) \geq 0, \quad i = 1, 2, \ldots, I,$$
$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, J.$$

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

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

Note that, for a pair of solutions $\mathbf{a}$ and $\mathbf{b}$, there are three Pareto dominance relation combinations possible:

$$\mathbf{a} \prec \mathbf{b} \text{ and } \mathbf{b} \not\prec \mathbf{a} \text{ or}$$
$$\mathbf{a} \not\prec \mathbf{b} \text{ and } \mathbf{b} \prec \mathbf{a} \text{ or}$$
$$\mathbf{a} \not\prec \mathbf{b} \text{ and } \mathbf{b} \not\prec \mathbf{a}.$$

Namely, either $\mathbf{a}$ dominates $\mathbf{b}$, $\mathbf{b}$ dominates $\mathbf{a}$, 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 the *population*. In each iteration, every solution from the population acts as a parent $\mathbf{p}$ to a newly created trial solution (also called candidate) $\mathbf{c}$. 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 $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, [\mathbf{x}_4, \ldots, \mathbf{x}_{n_p}] \in \mathbb{P}$, where n_p is the population size, to help construct a mutation vector $\mathbf{v}$ by vector operations, such as addition, subtraction 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:

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

The trial solution is then evaluated, *i.e.* the value of the cost function is calculated using Equation (15). Finally, the selection is performed by testing Pareto dominance in objective space between the trial solutions and their parents in a pair-wise manner. The dominated solutions are discarded while the dominating solutions form the population of the next iteration. In a case where 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 algorithm NSGA-II (Deb *et al.* 2002) to discard the worst solutions. The algorithm finishes either after a fixed number of cost function evaluations performed, after a solution of predefined quality is found, after the solutions have converged with a satisfactorily high confidence, or after a similar terminating condition is met.

Asynchronous Master-Slave DEMO (AMS-DEMO) (Depolli, Trobec and Filipič 2013) is a parallel extension of DEMO. AMS-DEMO distributes the tasks of DEMO between the slave processes, which are evaluating trial solutions in parallel and asynchronously, and the master process, which is performing all other tasks of DEMO and management of the slave processes. The master first generates trial solutions and sends them to slaves into asynchronous evaluation until all slaves are busy. Then it waits until one of the evaluations is complete. When a slave finishes the evaluation of a solution, it sends the results to the master. The master performs a selection on the solution and then generates a new trial solution, which it returns to the same slave. The slave then starts evaluating the newly received solution, while the master goes back to waiting. AMS-DEMO is able to exploit heterogeneous computer architectures with a large number of compute nodes, while retaining very good parallel efficiency. Its only real limitation is that to gain any benefits of parallel execution it requires non-trivial cost functions, *i.e.* the time it takes to evaluate the cost function is at least several milliseconds.

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 a 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 a 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 a Message Passing Interface (MPI). Both simulator and optimizer executables are written in C++, compiled with GCC 4.8 with optimizations enabled by -O3 switch . The code for the AMS-DEMO program is the same as described in Depolli, Trobec and Filipič (2013), while the code for the simulator is the same as used in Kosec and Šarler (2008). OpenMP® is built into the compiler while the MPI is implemented through the Open MPI library. Optimizer and simulator communicate via a file system, bash scripts, and the 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 $dt = 2.5 \times 10^{-5}$, maximal allowed dimensionless time $t_{\max} = 20$ and steady state criteria $|T_1 - T_0| < 10^{-7}$. The variable parameters, *i.e.* the x- and y-coordinates and widths and lengths of obstacles, are provided by the optimizer via a text file. If an obstacle is placed in such a way that its surface is outside the cavity, *e.g.* its x-coordinate equals one, then such an obstacle is not used in the calculations.

All the optimizations (unless specified otherwise) were performed with the parameters set to 'rand/1/bin' schema, $F=0.5$, $c_p = 0.2$ and stopping criterion set to the maximum number of evaluations performed (which differ between cases).

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 nonlinear 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 humans, the same does not hold for the stochastic optimization logic and this test serves as a benchmark of the optimization procedure. In other words, a 'closed form' solution exists, against which the solutions obtained by the optimization procedure can be compared, *i.e.* the performance of the optimization can be objectively assessed. 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.

It should be noted that increasing the number of obstacles results in a more complex scenario. The scenarios with more obstacles are therefore optimized with increased population size, which should help counter their increased complexities. Yet, as the results show, this is insufficient, and for the more complex scenarios, the optimization finds less optimal solutions, *i.e.* it covers the cavity less.

The optimization procedure can place one obstacle almost optimally and it performs very well with four obstacles. For a higher number of obstacles, *i.e.* Figures 3(c) and 3(d), a slight drop in solution

Table 1. Settings of the optimizer.

	Sub-cases			
Number of obstacles	1	4	9	16
Maximum obstacle size	1×1	$1/2\times1/2$	$1/3\times1/3$	$1/4\times1/4$
Number of cost function parameters	4	16	36	64
Population size	20	30	40	50
Number of simulations	2,000	6,000	12,000	18,000

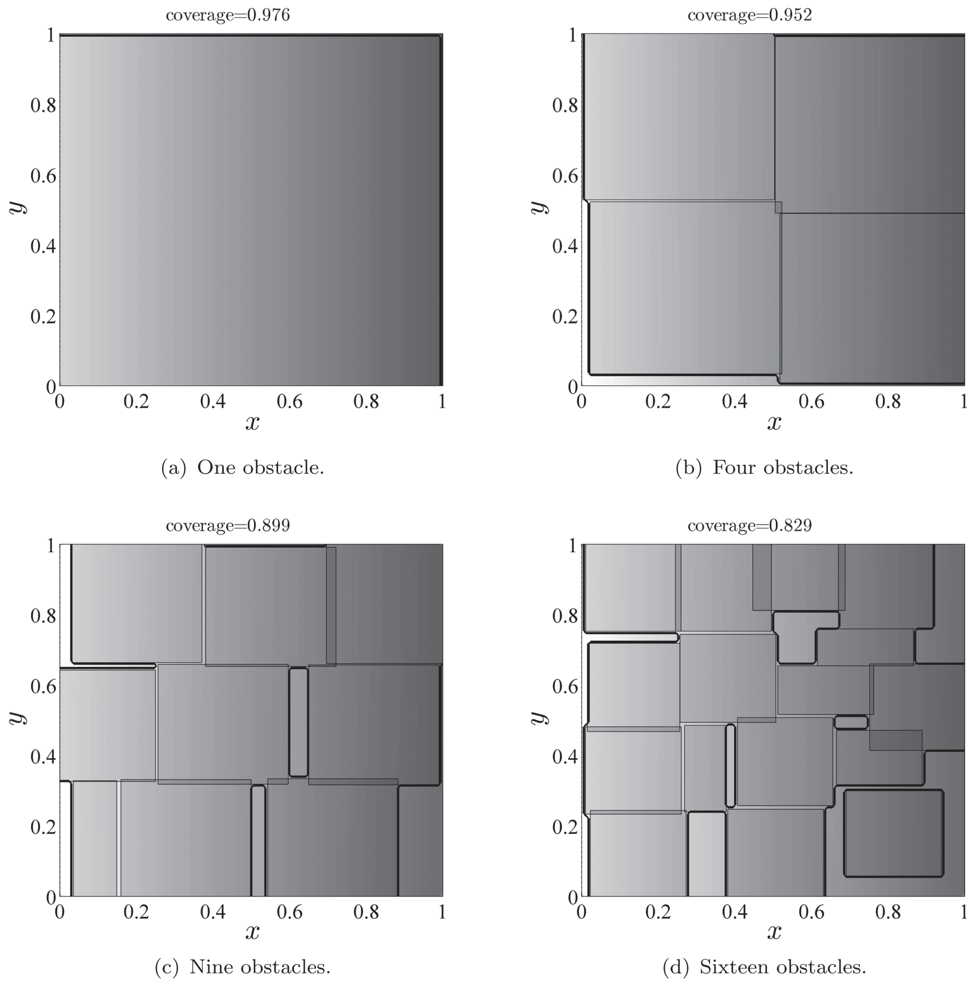


Figure 3. Temperature contour plot and obstacle positions for Case 1.

quality is noticeable. Although optimization settings could be optimized to produce better results, the fine tuning of several optimization parameters is beyond the scope of this work. It should also be be noted that better results could be obtained by running the optimization longer.

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 fewer obstacles. Much larger populations would probably help in obtaining better results in terms of heat flux, but at the price of an even higher number of simulations performed 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 a search for the best insulator with minimal material consumed is performed; therefore the solutions are measured using two criteria for minimization: the area covered by obstacles and the heat flux through the cavity.

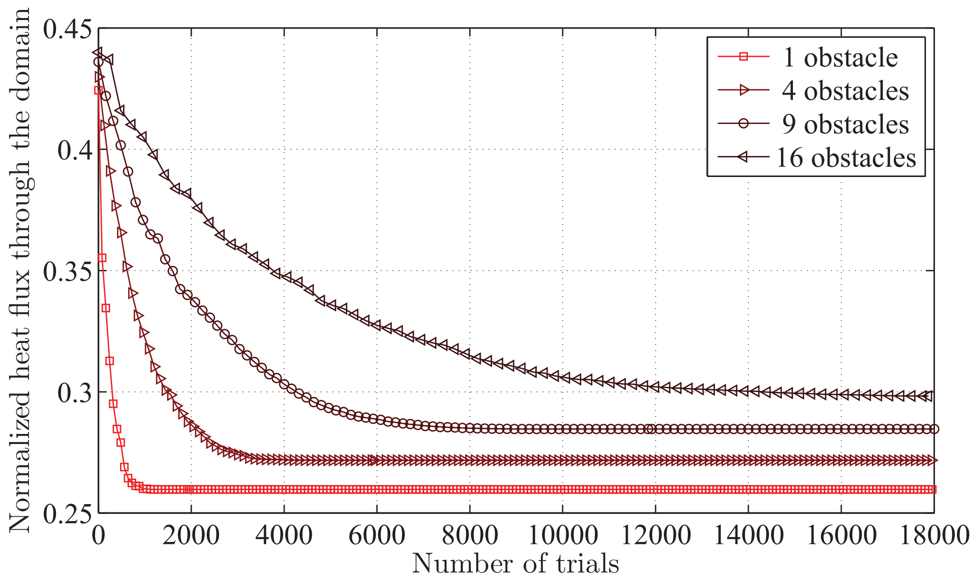


Figure 4. Convergence of the optimal solution expressed by the heat flux as a function of the number of simulations performed. Statistics are made over 10 repetitions of the optimization task.

Table 2. Settings of the optimizer.

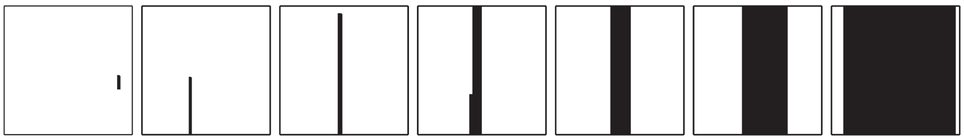
	Sub-cases					
Heat flow direction	Horizontal			Vertical		
Rayleigh number	10^4	10^5	10^6	10^4	10^5	10^6
Number of obstacles	10					
Maximum obstacle size	1×1					
Number of parameters	40					
Population size	50					
Number of simulations	5000					

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.

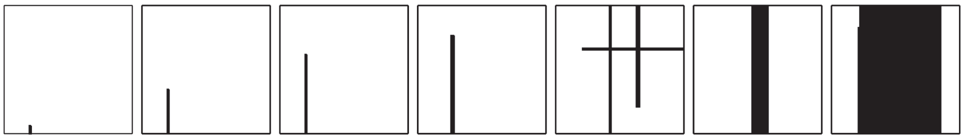
From the figure it follows that the optimization favours long and thin obstacles placed either vertically for limiting horizontal flux or horizontally for limiting vertical flux while keeping coverage low. Note that the heat flux itself is governed by the convective air flow, which always follows circular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

For the two lower Rayleigh numbers (Ra)—which should translate to an easier problem—the Pareto front shows a seemingly even distribution of solutions with variable trade-off between the obstacle coverage or the heat flux optimization. For 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.

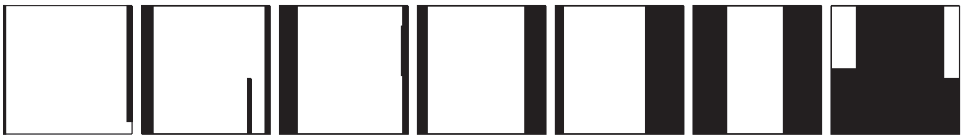
For this two-objective case, the convergence of results is checked via a hypervolume indicator (Zitzler and Thiele 1998), which transforms a set of objective vectors into a single number, which



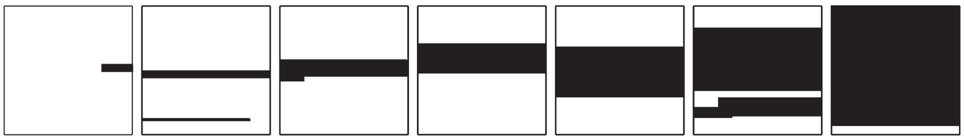
(a) Horizontal, $\mathrm{Ra} = 10^4$.



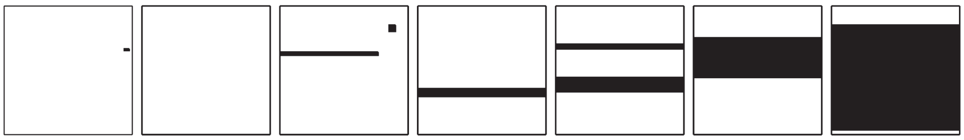
(b) Horizontal, $\mathrm{Ra} = 10^5$.



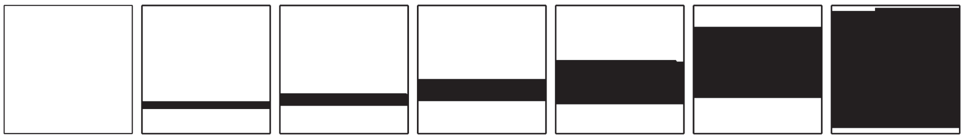
(c) Horizontal, $\mathrm{Ra} = 10^6$.



(d) Vertical, $\mathrm{Ra} = 10^4$.



(e) Vertical, $\mathrm{Ra} = 10^5$.



(f) Vertical, $\mathrm{Ra} = 10^6$.

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

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 the two-objective cases, with higher numbers indicating more optimal solutions. The hypervolume indicator is used to visualize the convergence of solutions for Case 2 in Figure 7. To emphasize the different convergence rates, the number of simulations performed before the solution front converged is also summarized in Table 3. The criterion for

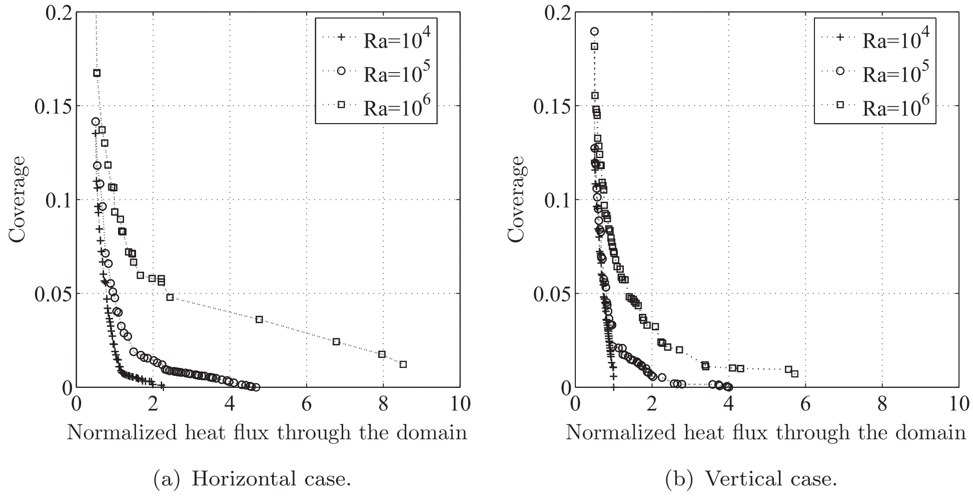


Figure 6. Pareto fronts discovered by the optimization.

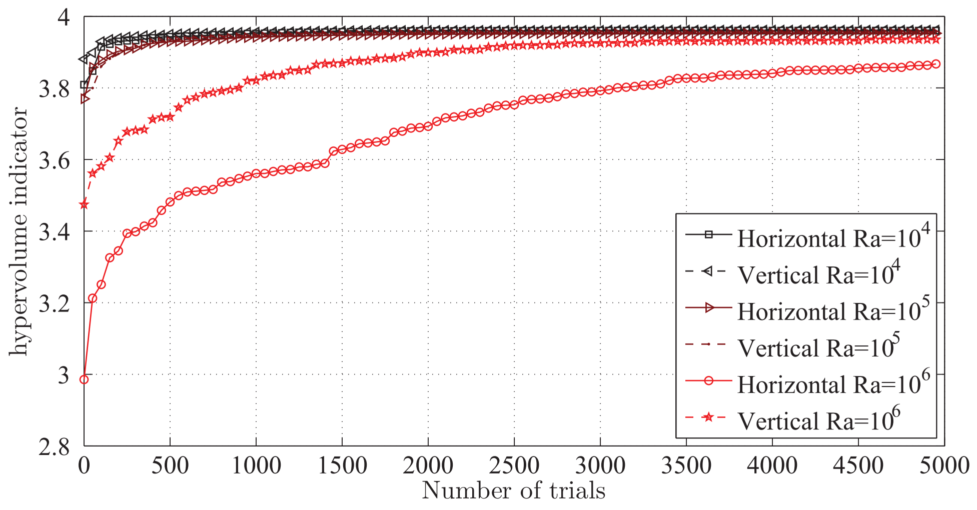


Figure 7. Convergence of the optimal solution expressed by the hypervolume indicator as a function of the number of simulations performed. Statistics are gathered over 10 repetitions of the optimization.

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

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

convergence is set 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.

3.4. Sensitivity analysis of the optimizer

The optimizer has several parameters 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, the optimizer will work well enough with almost any combination of parameter values, except for the most extreme ones. This means that the solutions will converge, but the convergence rate will not be optimal, and optimization runs will have to be very long to reach sufficient solution quality. In the worst case of sub-optimal parameter selection, the convergence could even be premature, which basically means that the optimizer will get stuck in a local optimum. Therefore, one can start experimenting with all the parameters set to the middle of their ranges and then tweak their values, until the gains from tweaking become too small to be worthwhile.

To shed some light on the problem of parameter selection, a local sensitivity analysis of the parameters is performed. The task of filling the whole cavity with four obstacles is selected as the test case for the optimizer. Note that the results would differ if either a different number of obstacles were selected or the task of optimization were different.

The three most important parameters of the optimizer are taken into consideration for the one-at-a-time analysis: 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 $[0.1, 0.2, \ldots, 0.9]$, while for n_p, the input set of values is $[10, 20, \ldots, 50]$. 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 simulations performed is used to show the difference between the parameter selections.

Firstly, the results of varying c_p are shown in Figure 8. The parameter c_p 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 the minimum 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 in Figure 9. In contrast to the previous figure, this one is much less dynamic, which implies that the optimizer is far more robust to the selection of F than it is to the selection of c_p. There is no clear pattern in the figure, which is most probably 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.

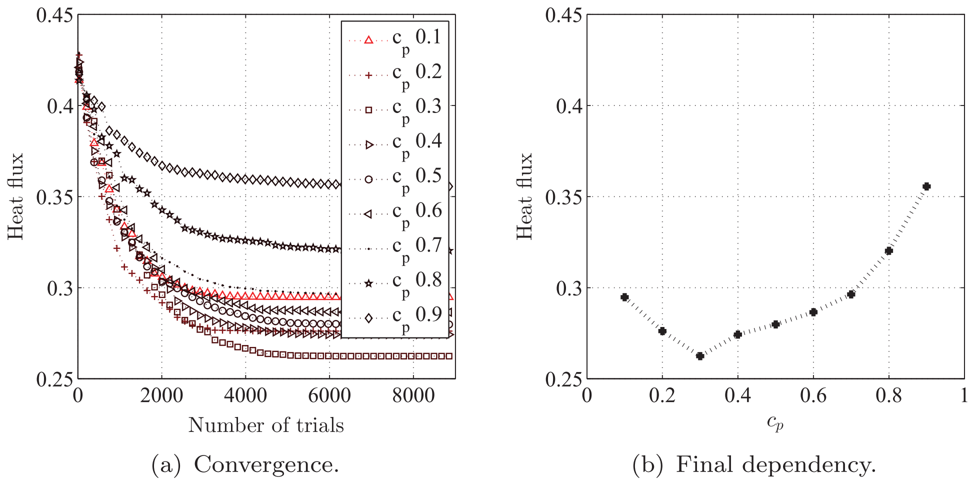
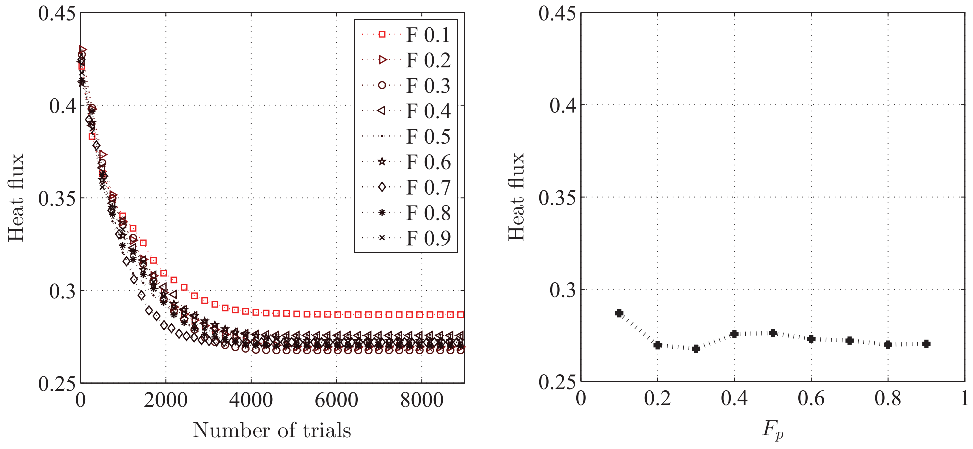
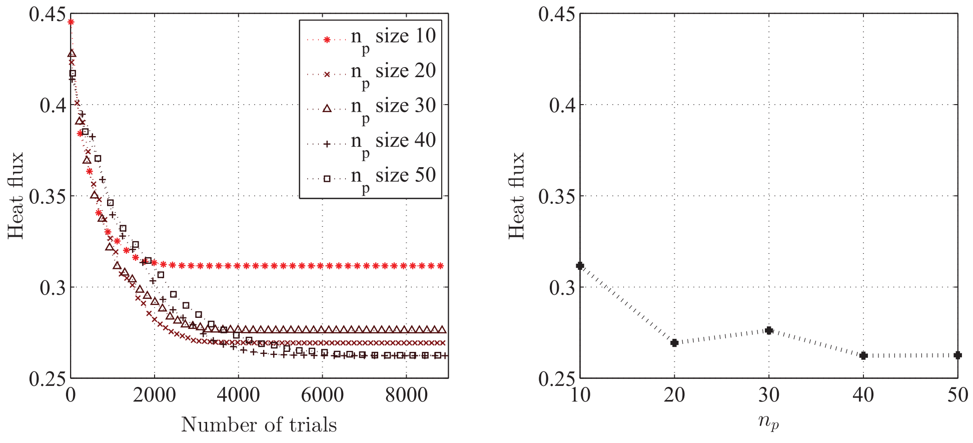


Figure 8. Convergence of the optimal solution for varying crossover probability, c_p.



(a) Convergence. (b) Final dependency.

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



(a) Convergence. (b) Final dependency.

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

Lastly, the results of varying n_p are shown in Figure 10. Population size is known to have a pronounced effect on 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 more slowly, since it makes it more random. The final limit of n_p is the total number of simulations performed, when the algorithm degenerates into ordinary Monte Carlo optimization. The experimental results confirm all the expectations, with the exception of $n_p = 20$ being worse than $n_p = 30$. This exception is almost certainly the result of noise caused by the low number of optimization repetitions.

3.5. Parallel speedup and efficiency

The efficiency of parallelization is measured through parallel speedup. Since solver and optimizer are parallelized separately, the parallel 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 $s = t_1/t_n$, where t_1 is the execution time on one 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 in the experiments made so far. One hundred parameter sets are randomly taken from one of the optimization runs performed in 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 in Figure 11(a). Speedups are calculated from total execution times, which include serial pre-processing, post-processing, and input and output operations. The 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 numbers of computers used. The detailed statistical analysis of a large number of runs of AMS-DEMO has been performed in Depolli, Trobec and Filipič (2013), with the main finding of no statistically significant differences 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—cluster nodes in this case—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 fewer computers than the population size, the calculated speedup is called *weak speedup*, to account for

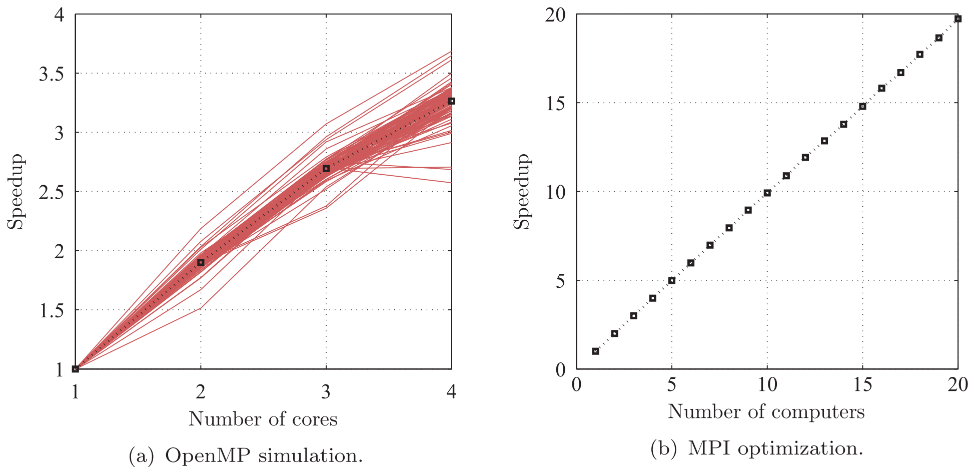


Figure 11. Speedup of the simulation and the optimization execution.

the fact that the solutions of different runs are not completely equivalent. Figure 11(b) shows the weak speedup calculated as the fraction of the total execution time on n computers relative to the 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 (the 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 number of computers tested.

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 with four cores and multiplying it by the speedup with 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 potential of simulation-based optimization using AMS-DEMO. 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 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 could be migrated to GPUs, which was demonstrated in Kosec and Zinterhof (2013), or, theoretically, to other computer accelerators that proliferate in 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.

The presented optimization technique might seem crude for the problem at hand and several improvements seem reasonable. The first improvement would be a better parametrization of the problem, since current parametrization contains several symmetries (*e.g.* the order of obstacles is irrelevant and the cavity is symmetric). Next would be to extend the optimizer by including local optimization of good solutions after the stochastic optimization has converged. Finally, the connection between optimizer and simulator could be improved by allowing the simulator to reuse past similar simulations and allowing the optimizer to monitor the running simulations and stop those with obviously sub-optimal setups in advance.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The authors acknowledge the financial support from the Slovenian Research Agency [programme group P2-0095].

References

Aage, N., and B. S. Lazarov. 2013. "Parallel Framework for Topology Optimization Using the Method of Moving Asymptotes." *Structural and Multidisciplinary Optimization* 47 (4): 493–505.
Abraham, A., L. Jain, and R. Goldberg, eds. 2005. *Evolutionary Multiobjective Optimization*. London: Springer-Verlag.
Burke, E. K., and G. Kendall, eds. 2003. *Introduction to Stochastic Search and Optimization*. Hoboken, NJ: Wiley.
Coello Coello, C. A., G. B. Lamont, and D. A. Van Veldhuizen. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation Series. Secaucus, NJ: Springer US.

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 (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, M., R. Trobec, and B. Filipič. 2013. "Asynchronous Master-Slave Parallelization of Differential Evolution for Multiobjective Optimization." *Evolutionary Computation* 21 (2): 261–291. doi:10.1162/EVCO_a_00076.
Eiben, A. 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.
Foli, K., T. Okabe, M. Olhofer, Y. Jin, and B. Sendhoff. 2006. "Optimization of Micro Heat Exchanger: CFD, Analytical Approach and Multi-Objective Evolutionary Algorithms." *International Journal of Heat and Mass Transfer* 49 (56): 1090–1099.
Keaveny, E. E., S. W. Walker, and M. J. Shelley. 2013. "Optimization of Chiral Structures for Microscale Propulsion." *Nano Letters* 13 (2): 531–537.
Kosec, G., M. Depolli, A. Rashkovska, and R. Trobec. 2014. "Super Linear Speedup in a Local Parallel Meshless Solution of Thermo-Fluid Problems." *Computers & Structures* 133: 30–38.
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 (7/8): 868–882.
Kosec, G., and R. Trobec. 2015. "Simulation of Semiconductor Devices with a Local Numerical Approach." *Engineering Analysis with Boundary Elements* 50: 69–75. doi:10.1016/j.enganabound.2014.07.013.
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 (1-5): 412–423.
Marck, G., M. Nemer, and J.-L. Harion. 2013. "Topology Optimization of Heat and Mass Transfer Problems: Laminar Flow." *Numerical Heat Transfer, Part B: Fundamentals* 63 (6): 508–539.
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, T., and B. Filipič. 2005. "DEMO: Differential Evolution for Multiobjective Optimization." In *Proceedings of the Third Conference on Evolutionary Multi-Criterion Optimization (EM 2005)*, 9–11 March 2005, Guanajuato, Mexico, Vol. 3410 of *Lecture Notes in Computer Science*, 520–533. Heidelberg: Springer-Verlag.
Sigmund, O., and K. Maute. 2013. "Topology Optimization Approaches." *Structural and Multidisciplinary Optimization* 48 (6): 1031–1055.
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.
Toghyani, S., A. Kasaeian, and M. H. Ahmadi. 2014. "Multi-Objective Optimization of Stirling Engine Using Non-Ideal Adiabatic Method." *Energy Conversion and Management* 80: 54–62.
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, UK: Elsevier.
Zitzler, E., and L. 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)*, 27–30 September 1998, Amsterdam, The Netherlands, Vol. 1498 of Lecture Notes in Computer Science Series, 292–301. Heidelberg: Springer.