Received August 2003
Revised March 2004
Accepted March 2004

Assessing maximum possible damage for contaminant release events

Fernando Camelli and Rainald Löhner
School of Computational Science and Informatics, George Mason University, Fairfax, Virginia, USA

Keywords *Programming and algorithm theory, Finite element analysis, Contamination, Hazards, Fluid dynamics*

Abstract *The combined use of damage criteria, genetic algorithms and advanced CFD solvers provides an effective strategy to identify locations of releases that produce maximum damage. The implementation is simple and does not require any change to flow solvers. A rather general criterion has been formulated to determine the damage inflicted by the intentional or unintentional release of contaminants. Results of two typical cases show that damage can vary considerably as a function of release location, implying that genetic algorithms are perhaps the only techniques suited for this type of optimization problem.*

1. Introduction

The intentional or unintentional release of hazardous materials can lead to devastating consequences. In order to prepare effective countermeasures, place sensors, or legislate, it is imperative to know the maximum possible damage a release in certain region can have. This problem may be cast as the optimization problem: maximize the damage, given a region of release locations. Damage in this context is defined as the number of people and property affected by the release of a contaminant.

Assuming that the amount of contaminant is finite, and that the population density in the region of interest is given, for any given meteorological condition the location of the release becomes the main input variable. Damage as a function of space can have many local extrema, as pockets of high concentration can linger in recirculation zones, or diffuse slowly while being transported along street canyons. This implies that traditional gradient-based approximation techniques will fail. For this reason, recourse is taken to genetic optimization algorithms that can obtain the maxima of such arbitrary functions.

Recent advances in CFD codes and hardware have made it possible to perform large-scale, high-resolution 3D runs in a reasonable amount of time. The flowfields for different meteorological conditions (wind direction, atmospheric conditions, humidity, temperature of building walls, emissions of heat exchangers, etc.) are typically calculated and stored on supercomputers. Given that in some cases up to an hour of real time needs to be computed, these runs can consume a considerable amount of computer time. The effects of different release scenarios (amount, locations, people density) are then simulated for these pre-stored flowfields. It has become possible to

This work was partially supported by DTRA under the auspices of CAMP.

Engineering Computations
Vol. 21 No. 7, 2004
pp. 748-760

0264-4401
DOI 10.1108/02644400410548387

perform each one of the release scenarios in a matter of minutes on PC platforms, even for grids in excess of a million elements. This implies that on networks of PCs, or PC clusters, hundreds of release scenarios can be simulated per day.

The remainder of the paper is organized as follows: Section 2 defines the damage criterion used. Sections 3 and 4 describe the genetic optimization algorithm and solver employed. In Section 5 two examples are presented. Finally, in Section 6 some conclusions are drawn and an outlook for future work is given.

2. Damage criterion

Assessing the damage to people and property is difficult. For the present study, the damage to property was not considered. Damage to health was assumed to occur above a certain threshold concentration c_0 in regions occupied by pedestrians. In order to monitor the total damage in a given time-interval $[0, T]$, the following damage criterion was used:

$$I(\mathbf{x}_0) = \int_0^T \int_\Omega c^* \rho \, \mathrm{d}\Omega \, \mathrm{d}t, \tag{1}$$

where ρ, Ω, $\mathbf{x}_0$ denote the population density, volume of interest and location of source, respectively, and c^* is given by the Heaviside function:

$$c^* = 1 \text{ if } c \geq c_0, \quad c^* = 0 \text{ if } c < c_0. \tag{2}$$

The population density was assumed to be given for the geometry, time of day, and meteorological conditions considered. Obviously, the higher the population density, the higher the damage, meaning that rush-hour densities were favoured for "worst-case" scenarios.

3. Genetic optimization algorithm

Suppose we are given the optimization problem:

$$I(\beta) \rightarrow \max. \tag{3}$$

In order to norm the input variables (i.e. the release scenarios), a range $\beta^i_{\min} \leq \beta^i \leq \beta^i_{\max}$ is set for each variable defining a release scenario. In the simplest case, this can be the spatial coordinate of the release point. An instantiation is then given by:

$$\beta^i = (1 - \alpha^i)\beta^i_{\min} + \alpha^i \beta^i_{\max}, \tag{4}$$

implying $I(\beta) = I(\beta(\alpha))$. By working only with the α^i, an abstract, non-dimensional, bounded ([0, 1]) setting is achieved, that allows for a large degree of commonality among various optimization algorithms.

Given the optimization problem (equation (1)), a simple and very general way to proceed is by copying evolution in nature: try variations of β, and keep the ones that maximize (i.e. improve) the cost function $I(\beta)$. This class of optimization techniques are denoted as genetic algorithms (Goldberg, 1989). Although costly, many function evaluations are required, genetic algorithms offer important advantages: they represent a completely general technique, able to go beyond local minima and hence suitable for "rough" cost functions I with multiple local minima, and only require

function evaluations (i.e. no gradient). Genetic algorithms have been used on many occasions in many fields (Goldberg, 1989; Quagliarella *et al.*, 1998; Winter *et al.*, 1995). The key elements of genetic algorithms are:

- a fitness measure, given by $I(\beta)$, to measure different release scenarios against each other;
- chromosome coding, to parametrize the release locations given by β;
- population size required to achieve robust optimization;
- selection, to decide which members of the present/next generation are to be kept/used for reproductive purpose; and
- mutation to obtain "offspring" not present in the current population.

The most straightforward way to code the release locations into chromosomes is by defining them to be the parameters $0 \leq \alpha^i \leq 1$. An instantiation is then given by equation (2). The population required for a robust selection needs to be sufficiently large. A typical choice for the number of individuals in the population M as compared to the number of chromosomes (release locations) N is: $M > \mathrm{O}(2N)$ for large N. Given a population and a fitness measure associated with each individual, the next generation has to be determined. In order to achieve a monotonic improvement in release scenarios, a percentage of "best individuals" c_k of each generation is kept (value used here: $c_k = \mathrm{O}$ (10 percent)). Furthermore, a percentage of "worst individuals" c_c are not admitted for reproductive purposes (value used here: $c_c = \mathrm{O}$ (75 percent)). Each new individual is generated by selecting randomly a pair i, j from the allowed list of individuals, and combining the chromosomes randomly. Many possible ways have been proposed to combine chromosomes, such as chromosome splicing, arithmetic and random pairing, etc. (Goldberg, 1989). In the present case, arithmetic pairing was chosen. A random pairing factor $-\xi < \gamma < 1 + \xi$ is selected and applied to all variables of the chromosomes in a uniform way. The chromosomes for the new individuals are given by:

$$\alpha = (1 - \gamma)\alpha^i + \gamma\alpha^j. \tag{5}$$

We remark that γ lies outside [0, 1] (a typical value is $\xi = 0.2$). This is required, as otherwise the only way to reach locally outside the chromosome interval given by the pair i, j (or the present population) is via mutation, a slow and therefore expensive process. On the other hand, a population that is not modified continuously by mutations tends to become uniform, implying that the optimization may end in a local minimum. Therefore, a mutation frequency: $c_{\mathrm{m}} = \mathrm{O}(0.25/N)$ has to be applied to the new generation, modifying chromosomes randomly. Summarizing, the basic steps required per generation for generic algorithms are the following:

Ga1 Evaluate the fitness function $I(\beta(\alpha))$ for all individuals;

Ga2 Sort the population in ascending (descending) order of I;

Ga3 Retain the c_k best individuals for the next generation;

Ga4 `while:` Population incomplete

Select randomly a pair i, j from c_{c} list

Obtain random pairing factors $-\xi < \gamma_k < 1 + \xi$

Obtain the chromosomes for the new individual: $\alpha = (1-\gamma)\alpha^i + \gamma\alpha^j$
Limit the values of α to be in admissible range: $\alpha = \max(0, \min(1, \alpha))$
`end while.`

4. Dispersion solver

Any contaminant transport or release simulation solves the classic advection-diffusion equation

$$c_{,t} + \mathbf{v}\cdot\nabla c = \nabla k \nabla c + S. \tag{6}$$

Here c, $\mathbf{v}$, k denote the concentration, velocity and diffusivity of the medium, and S the source-term. The (in most cases unsteady) velocity field $\mathbf{v}$ is assumed to be divergence free ($\nabla\cdot\mathbf{v} = 0$) and precomputed, i.e. given as an input. The spatial discretization of the computational domain is performed with linear tetrahedral elements. Denoting by N^i the shape-function of point i, the Galerkin weighted residual method for equation (6) results in:

$$\mathbf{M}\cdot c_{,t} + \mathbf{A}\cdot c = -\mathbf{K}\cdot c + \mathbf{S}, \tag{7}$$

with

$$\mathbf{M} = \int N^i N^j \,\mathrm{d}\Omega, \quad \mathbf{A} = \int N^i \mathbf{v}\cdot\nabla N^j \,\mathrm{d}\Omega, \quad \mathbf{K} = \int \nabla N^i k \nabla N^j \,\mathrm{d}\Omega,$$
$$\mathbf{S} = \int N^i S \,\mathrm{d}\Omega. \tag{8}$$

In most instances, the Galerkin weighted residual method will not yield (physically correct) monotone results for advection dominated cases. A number of schemes have been devised to replace the "Galerkin fluxes" by so-called "numerically consistent fluxes". Upwinding (van Leer, 1974; Sweby, 1984; Hirsch, 1991), anisotrophic balancing dissipation (Kelly *et al.*, 1980; Brookes and Hughes, 1982) and flux-corrected transport (FCT) (Parrott and Christie, 1986; Löhner *et al.*, 1987) are examples of such schemes. In the present case, edge-based limiting (Löhner, 2001) is employed.

Given that one is interested in high temporal fidelity, and that the diffusive terms are typically very small, explicit time integration schemes are used for equation (7). Without loss of generality, consider the following m-stage Runge-Kutta scheme to go from timestep n to $n+1$:

$$\mathbf{M}\cdot c^i = \mathbf{r}^i = \mathbf{M}\cdot c^0 + \alpha^i(\mathbf{S} - (\mathbf{A}+\mathbf{K})\cdot c^{i-1}), \quad i = 1, m, \tag{9}$$

where, $c^0 = c^n$, $\alpha^i = 1/(m+1-i)$ and $c^{n+1} = c^m$. The bulk of the CPU requirements for a scheme of this kind is in the evaluation of the right-hand sides $\mathbf{r}^i$. These required matrix-vector multiplications can be performed in a variety of ways. The simplest is by performing loops over the elements, evaluating all matrix-vector products at the element level (Bathe, 1995; Zienkiewicz and Taylor, 2000; Löhner, 2001). A more efficient way to accomplish this for low-order elements can be achieved by switching to an edge-based data structure (Löhner, 2001), and this option is employed here.

5. Examples

The evaluation of maximum possible damage using the genetic algorithm is presented for two cases:

(1) An urban area composed of several buildings;

(2) A generic subway station.

The finite element code FEFLO, a general-purpose flow solver code, has been used for these two cases. This particular code has been repeatedly benchmarked and compared to experiments (Camelli and Löhner, 2000; Hanna *et al.*, 2002; Camelli *et al.*, 2003, 2004) and is routinely used for production runs. However, we emphasize that any other CFD tool capable of accurate dispersion calculations could have been used instead. The flow field was precalculated and stored in both cases in order to speed up the time for each of the evaluations of the fitness measure. These cases were run on PC and workstation platforms. Some of the results obtained are counter-intuitive, revealing interesting worst-case scenarios which would not have been presumed at first hand.

5.1 Urban area

The first example considers the intentional release in an area representative of an inner city composed of three by two blocks. The geometry definition and the surface mesh are shown in Figure 1. The incompressible Navier-Stokes equations were solved with Smagorinsky (1963) turbulence model. A logarithmic profile was applied as inflow boundary condition with a mean velocity of 2 m/s at a height of 10 m:

$$u = \frac{u_*}{\kappa} \ln\left(\frac{z}{z_0}\right) \tag{10}$$

where u_* is the friction velocity, κ the von Karman constant, z the vertical height and z_0 the roughness coefficient. The slip velocity condition with Law of the Wall was used for all walls. The release time was assumed to be 10 s, and the simulations were carried out for at least 800 s of real time. Each of the fitness evaluations (i.e. dispersion simulation runs) took approximately 80 min using a PC with Intel P4 chip running at 2.53 GHz with 1 GB RAM, Linux OS and Intel compiler.

Three areas of release have been studied (Figure 2): the upwind zone of the complex of buildings, the street level, and the core of one of the blocks. In all cases, the height of release was set to $z = 1.5$ m. The fitness/cost function as defined in Section 2 requires a volume of integration, as well as a population density. The volume of integration is the area shown in Figure 2 with a height of 3 m above the ground. The population density was assumed to be constant. The genetic optimization was carried out for 20 generations, with two chromosomes (x/y location of release point) and ten individuals in the population. The fitness/cost function evaluations for each of the zones of interest are shown in Figure 3(a)-(c). In all the areas a rapid convergence to maxima is reached.

The location and associated damage function for each of the releases computed during the optimization process are shown in Figure 4(a)-(d). Interestingly, the maximum damage is produced in the street area close to one of the side corners of the blocks. The cloud (Figure 5(a)-(f)) shows the suction effect produced by the streets that are in the direction of the wind, allowing for a very long residence time of contaminants close to the ground for this particular release location.

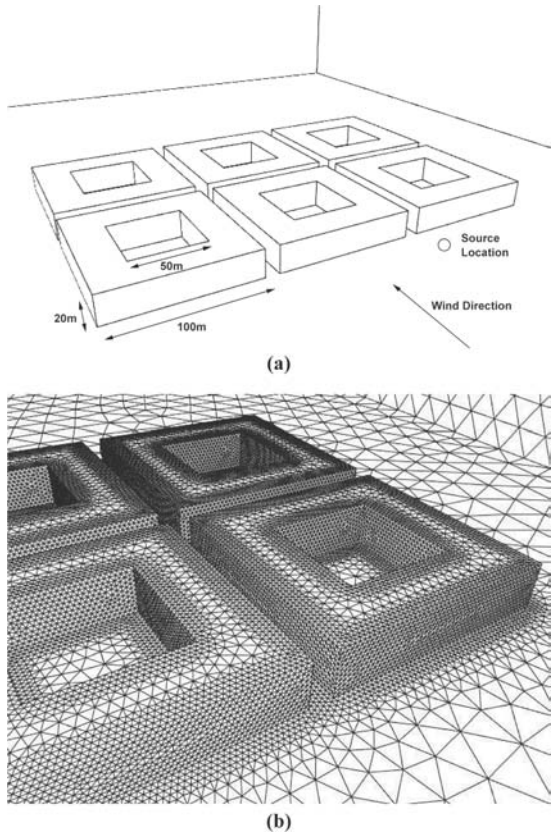

Figure 1.
Problem definition and detail of surface mesh

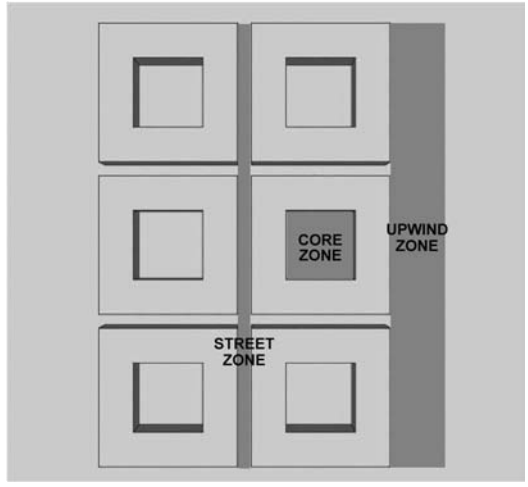

Figure 2.
Zones considered for release

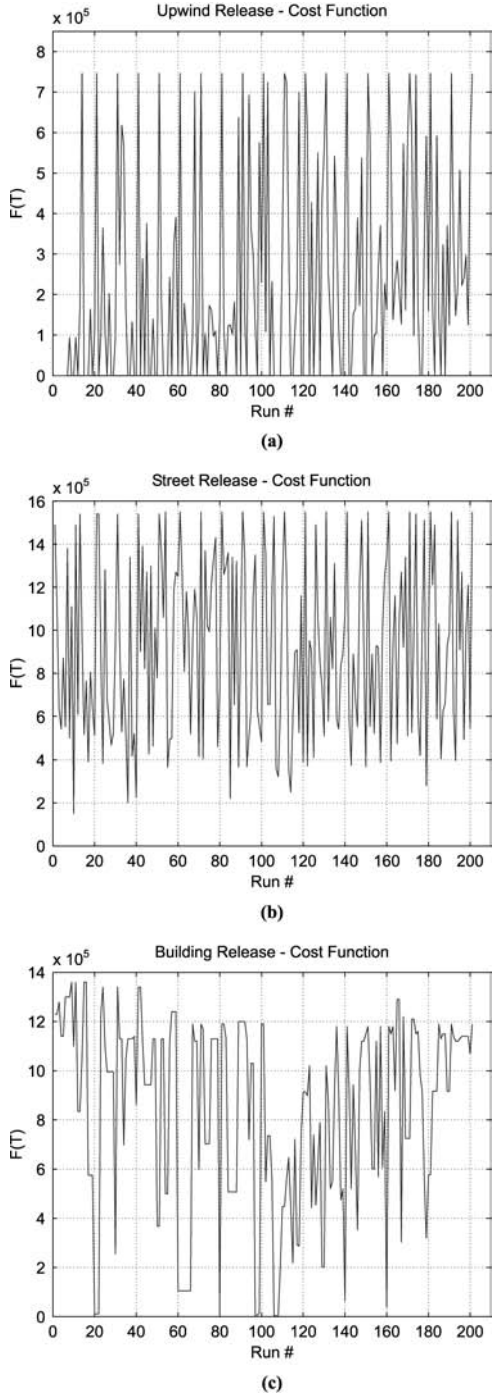

Figure 3.
Fitness/cost function evaluation

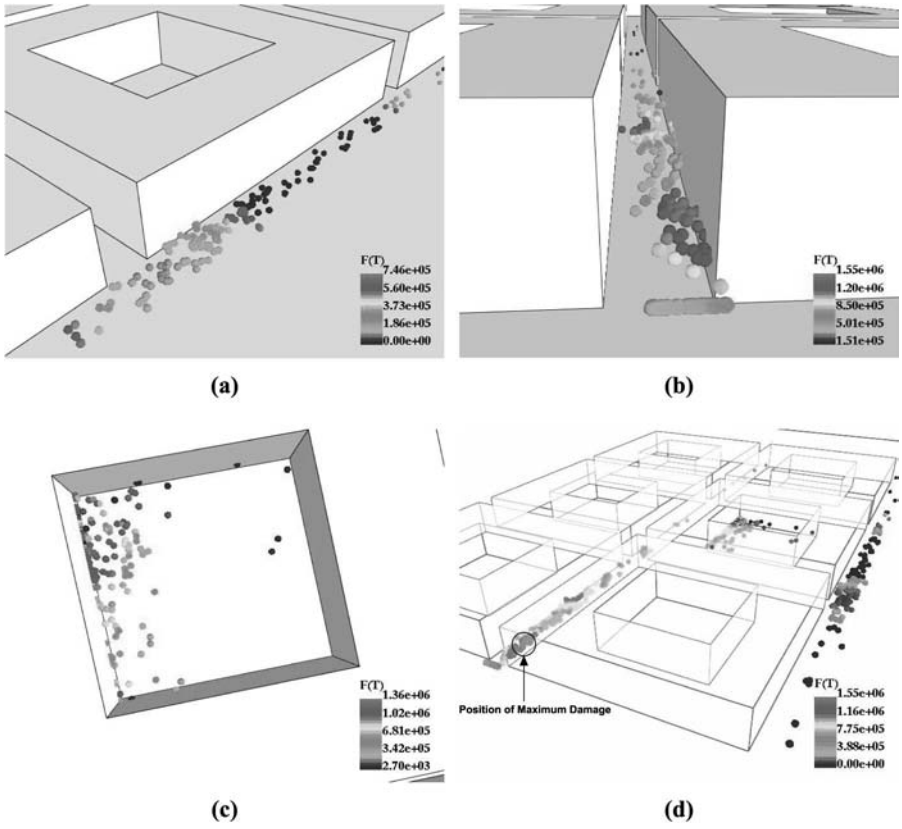

Figure 4.
Release positions coloured according to damage

5.2 Generic subway station

The second example studies the release of contaminants in a generic subway station. The geometry definition is shown in Figure 6. As before, the incompressible Navier-Stokes equations were solved with the Smagorinsky turbulence model with Law of the Wall. On one of the end sides a time dependent inflow of the following form:

$$u(t) = b(t - 60)^3 \, \mathrm{e}^{-a(t-60)} + u_0 \tag{11}$$

is applied. Here, $b = 0.46\,\mathrm{m/s}$, $a = 0.5\,1/\mathrm{s}$, and $u_0 = 0.4\,\mathrm{m/s}$. This inflow velocity, shown in Figure 7, corresponds approximately to the velocities measured at a New York City subway station (Pflitsch *et al.*, 2000), and simulates the incoming subway to the station platform without modeling the actual train. A puff release of 10 s was assumed. The real time modeled was of 450 s. All the runs were performed on a workstation with DEC Alpha chip running at 666 MHz, with 1 GB of RAM, Linux OS and Compaq compiler. Each run lasted between 60 and 90 min. The strategy for the genetic algorithm was the same as for the urban area example: 20 generations and a population of ten individuals. Two areas of release were studied: platform and

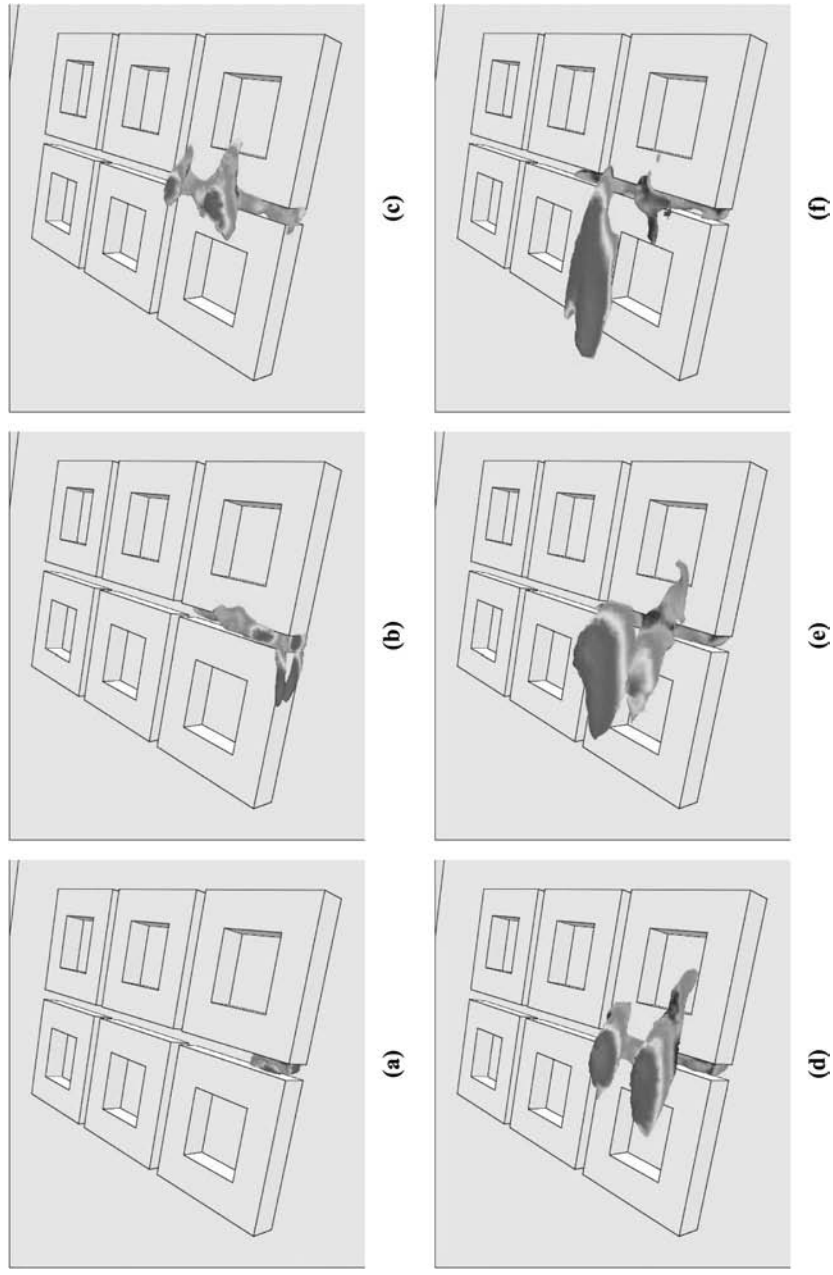

Figure 5.
Evolution of maximum damage cloud

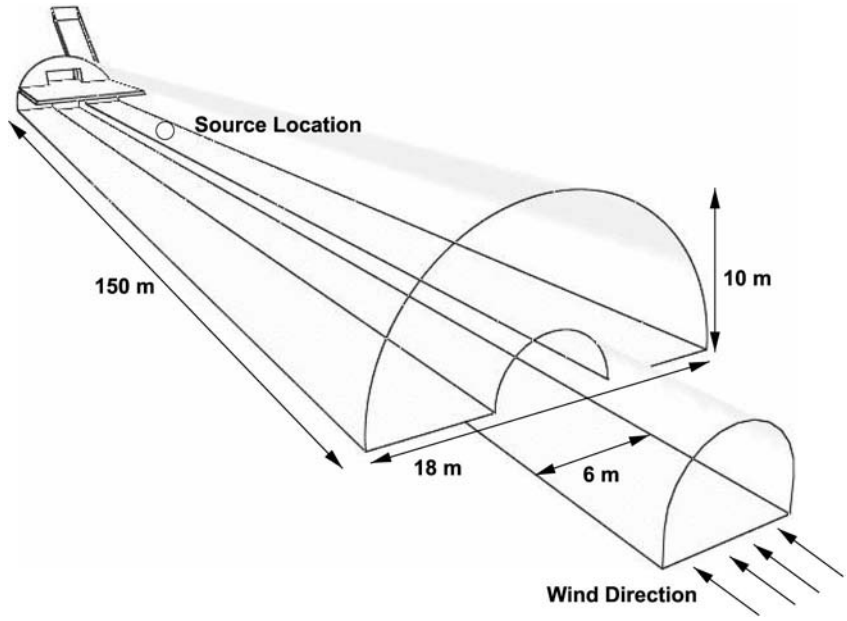

Figure 6.
Generic subway station: geometry definition

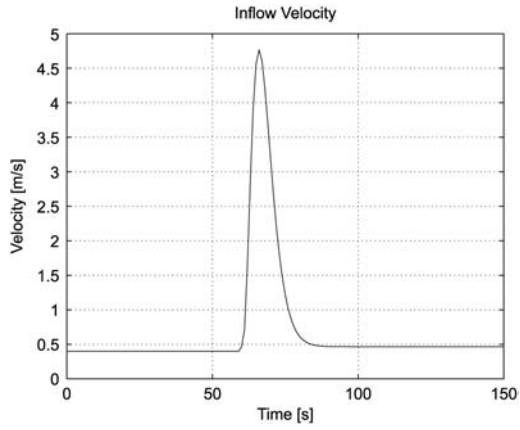

Figure 7.
Prescribed inflow velocity as a function of time

mezzanine. The volume of integration for the damage evaluation comprised both platforms, the mezzanine level, and the exit stars. The convergence of the fitness/cost function is shown in Figure 8(a) and (b). The convergence is slower in the platform than in the mezzanine. The position of the releases and the associated damage function is shown in Figure 9(a) and (b). Note that the maximum damage is produced by a release in the platform, whose contaminant cloud at $T = 450\,s$ is shown in Figure 10(a) and (b).

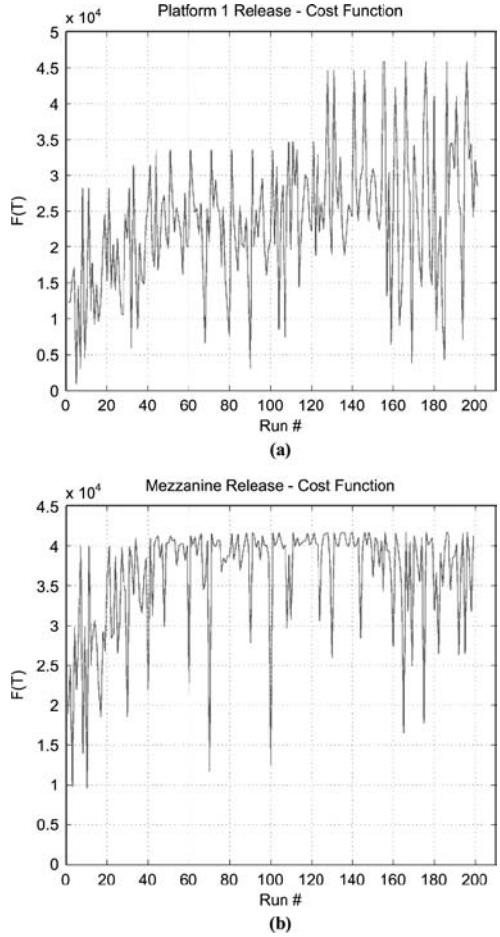

Figure 8.
Fitness/cost function evaluation in the platform and mezzanine

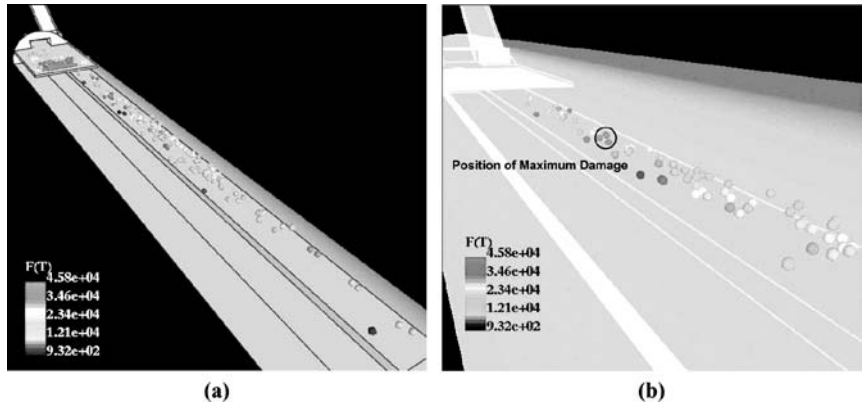

Figure 9.
Release positions coloured according to damage

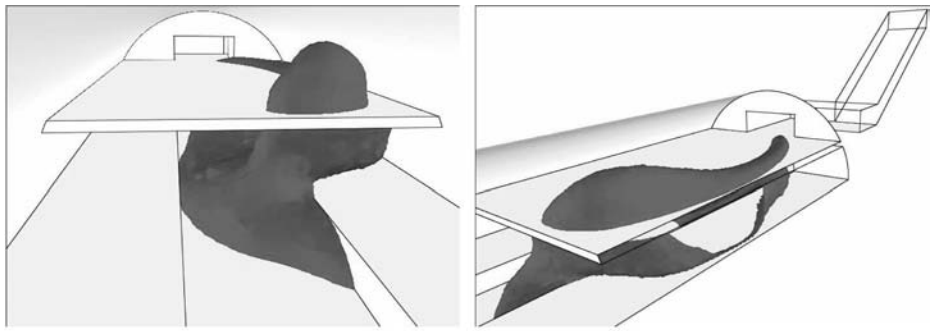

Figure 10.
"Maximum damage" cloud at $T = 450$ s

6. Conclusions and outlook

The use of damage criteria, genetic algorithms and advanced CFD solvers has been proven to be effective to identify locations of releases that produce maximum damage. The implementation is simple and does not require any changes to flow solvers. The main difficulty is the design of proper damage (fitness/cost) functions to account for the different toxic agents, their effect on health, the particular circumstances of the release location and time, people density and atmospheric conditions. The locations of the releases in both examples show a function with multiple local maxima and minima, making this a difficult optimization problem if treated with more conventional gradient methods. The use of precomputed flow fields, as well as recent advances in flow solvers, has resulted in a considerable reduction of runtime, making genetic optimization possible on networks of PCs.

Future work will focus on:

- further refinement of the damage (fitness/cost) functions;
- added realism for geometries as well as boundary conditions (in particular heated walls and HVAC exhaust and intake vents);
- added realism in the dispersion physics, such as deposition, vegetation effect, evaporation, chemical reactions in the atmosphere, etc.
- the use of grid computing to harness the maximum number of resources possible at any given time.

References

Bathe, K.J. (1995), *Finite Element Procedures*, Prentice-Hall, Englewood Cliffs, NJ.

Brooks, A.N. and Hughes, T.J.R. (1982), "Streamline upwind/Petrov Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations", *Comp. Meth. Appl. Mech. Eng.*, Vol. 32, pp. 199-259.

Camelli, F. and Löhner, R. (2000), "Flow and dispersion around buildings: an application with FEFLO", Proc. of ECCOMAS 2000 Conf., September, Barcelona.

Camelli, F., Löhner, R., Sandberg, W.C. and Ramamurti, R. (2004), "VLES study of ship stack gas dynamics", AIAA-04-0072.

Camelli, F., Soto, O., Löhner, R., Sandberg, W. and Ramamurti, R. (2003), "Topside LPD17 flow and temperature study with an implicit monolithic scheme", AIAA-03-0969.

Goldberg, D.E. (1989), *Genetic algorithms in search, Optimization and Machine Learning*, Addisonn-Wesley, Reading, MA.

Hanna, S.R., Tehranian, S., Carissimo, B., Macdonald, R.W. and Löhner, R. (2002), "Comparisons of model simulations with observations of mean flow and turbulence within simple obstacle arrays", *Atmospheric Environment*, Vol. 36, pp. 5067-79.

Hirsch, C. (1991), *Numerical Computation of Internal and External Flow*, Wiley, New York, NY.

Kelly, D.W., Nakazawa, S., Zienkiewicz, O.C. and Heinrich, J.C. (1980), "A note on anisotropic balancing dissipation in finite element approximation to convection diffusion problems", *Int. J. Num. Meth. Eng.*, Vol. 15, pp. 1705-11.

Löhner, R. (2001), *Applied CFD Techniques*, Wiley, New York, NY.

Löhner, R., Morgan, K., Peraire, J. and Vahdati, M. (1987), "Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier-Stokes equations", *Int. J. Num. Meth. Fluids*, Vol. 7, pp. 1093-109.

Parrott, A.K. and Christie, M.A. (1986), "FCT applied to the 2-D finite element solution of tracer transport by single phase flow in a porous medium", in Morton, K.W. and Baines, M.J. (Eds), *Proc. ICFD-Conf. Num. Meth. in Fluid Dyn.*, Academic Press, Reading, MA.

Pflitsch, A., Kleeberger, M. and Küsel, H. (2000), "On the vertical structure of air flow in the subway New York city and Dortmund (Germany)", Proc. 4th Annual George Mason Univ. Transport and Dispersion Modeling Workshop, July, Fairfax, VA.

Quagliarella, D. *et al.* (Eds) (1998), *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York, NY.

Smagorinsky, J. (1963), "General circulation experiments with the primitive equations, I. The basic experiment", *Mon. Weather Rev.*, Vol. 91, pp. 99-164.

Sweby, P.K. (1984), "High resolution schemes using flux limiters for hyperbolic conservation laws", *SIAM J. Num. Anal.*, Vol. 21, pp. 995-1011.

van Leer, B. (1974), "Towards the ultimate conservative scheme. II. Monotonicity and conservation combined in a second order scheme", *J. Comp. Phys.*, Vol. 14, pp. 361-70.

Winter, G., Periaux, J., Galan, M. and Cuesta, P. (Eds) (1995), *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York, NY.

Zienkiewicz, O.C. and Taylor, R. (2000), *The Finite Element Method*, 5th ed., Elsevier, Amsterdam.

Further reading

Arya, S.P. (1998), *Introduction to Micrometeorology*, Academic Press, New York, NY.

Arya, S.P. (1999), *Air Pollution Meteorology and Dispersion*, Oxford University Press, Oxford.

Hanna, S.R., Briggs, G.A. and Hosker, R.P. (1982), *Handbook on Atmospheric Diffusion*, NOAA DOE/TIC-11223.

Löhner, R., Chi, Y., Cebral, J.R., Soto, O., Camelli, F. and Waltz, J. (2003), "Improving the speed and accuracy of projection-type incompressible flow solvers", AIAA-03-3991-CP.

Stern, A.C., Boudel, R.W., Turner, D.B. and Fox, D.L. (1984), *Fundamentals of Air Pollution*, Academic Press, New York, NY.