Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARTIN FRANK[1]
ARMIN FÜGENSCHUH[2]
MICHAEL HERTY[3]
LARS SCHEWE[4]

The Coolest Path Problem[5]

[1]RWTH Aachen University, Mathematik, Templergraben 55, D-52056 Aachen
[2]Zuse Institut Berlin, Takustr. 7, D-14195 Berlin
[3]RWTH Aachen University, Mathematik, Templergraben 55, D-52056 Aachen
[4]TU Darmstadt, Fachbereich Mathematik, Schlossgartenstrasse 7, D-64289 Darmstadt
[5]to appear in: Networks&Heterogeneous Media

ZIB-Report 09-37 (November 2009)

THE COOLEST PATH PROBLEM

MARTIN FRANK, ARMIN FÜGENSCHUH, MICHAEL HERTY, AND LARS SCHEWE

Abstract. We introduce the coolest path problem, which is a mixture of two well-known problems from distinct mathematical fields. One of them is the shortest path problem from combinatorial optimization. The other is the heat conduction problem from the field of partial differential equations. Together, they make up a control problem, where some geometrical object traverses a digraph in an optimal way, with constraints on intermediate or the final state. We discuss some properties of the problem and present numerical solution techniques. We demonstrate that the problem can be formulated as a linear mixed-integer program. Numerical solutions can thus be achieved within one hour for instances with up to 70 nodes in the graph.

Continuous and discrete optimization are at present two distinct areas of mathematics. From time to time, discrete optimizers stumble over a problem which has some intrinsic nonlinear continuous structure, sometimes modeled using partial differential equations. Then they most likely would try to get rid of these continuous parts, such that a pure combinatorial problem remains. Similarly, if a person with a background in continuous optimization gets involved with a problem that involves discrete decisions, he or she would most likely try to relax the discontinuities to some continuous constraints, in order to apply some well-understood methods of the field. For both of them it is true that *if one only owns a hammer then every problem must be a nail*. However it is also true that if one always stays within its own cosy corner of the world, nothing new can emerge from that.

Our research is motivated by the fact that both worlds can inspire the respective other by sharing ideas and methods. So to start the discussion at some point we combine two problems into a new one that was not studied before (to the best of our knowledge). From the discrete world we consider the shortest path problem on a directed graph. The contribution from the continuous world is the heat conduction problem. Both problems are combined into a new optimization problem, which we suggest to coin *the coolest path problem*.

1. The Coolest Path Problem

We consider the following problem. Given is a directed graph $D = (V, A)$ with vertex set V and arc set A, and two distinct nodes $v, w \in V$. An v-w-path P in D of length n is defined as a sequence of vertices and arcs of the form $P = (v_0, a_1, v_1, a_2, v_2, \ldots, v_{n-1}, a_n, v_n)$, where $v_0 = v, v_n = w, a_i = (v_{i-1}, v_i)$, and the arcs in P are pairwise different. Imagine that a geometric object $\Omega \subset \mathbb{R}^3$ traverses the network from v to w. The initial temperature of the object is given by $u_0 : \Omega \to \mathbb{R}_+$. Associated with each arc $a \in A$ is a temperature $T_a(x) \in \mathbb{R}_+$ for $x \in \partial\Omega$. On each arc a the boundary of the object $\partial\Omega$ is exposed to the prevalent temperature $T_a(x)$ for $x \in \partial\Omega$ for a certain, arc-dependent time, so that the object is heated up or cooled down. At the end, at vertex w, the temperature distribution within

the object is given by the path-dependent function $u_P : \Omega \to \mathbb{R}_+$. The *coolest path problem* (CPP, for short) asks for an v-w-path P such that the average temperature $\bar{u}_P := \frac{1}{vol(\Omega)} \int_\Omega u_P \, dV$ of the object at w is minimal. The coolest path problem thus combines the combinatorial problem of finding a shortest v-w-path, where "shortest" refers to the amount of absorbed heat on the path, which is modelled by the heat equation.

As a real-world application of this problem one might think of a production line where some product (the object) has to pass certain manufacturing steps. These steps impose some heating or cooling to the material. After the end of one step there is a number of other succeeding steps that have to be carried out afterwards, until the product reaches the output. At the end, the product should be as cool as possible.

Besides this basic version of the problem, there are natural variations and extensions which we also consider in the sequel.

(1) Other objective functions. For example, one can take the temperature $u_P(x)$ at a certain point $x \in \Omega$ or the maximum temperature $\max\{u_P(x) : x \in \Omega\}$ as objective functions.
(2) Temperature gradients. The goal here is to find a path P such that the norm of gradient $\|\text{grad}(u)(\cdot)\|$ is minimal, either at a given point x, at the maximum within Ω, or in the average.
(3) Restrictions along the path. The above objective functions, together with a lower or upper bound can be taken as constraints. In this case we have to deal with a feasibility problem (i.e., finding a path with the given property), or together with any other of the objective functions from above, as an optimization problem with further constraints on the path.
(4) Control problems. The goal is to achieve a final state, such as a desired heat distribution at vertex w, and finding a path such that the actual heat distribution is closest possible to the prescribed one.

Another interesting variant is the coolest Hamiltonian cycle (CHC, for short) in a digraph. In the classical Hamiltonian cycle (HC) problem one is interested in a tour (or cycle) through all nodes that starts and end at the same node, and enters and leaves every node exactly ones. We remark that HC is NP-complete, see the monograph by Garey and Johnson [7]. In the "cool" version, one wants to end up with the coolest possible object (with respect to some objective functional). The CHC can also be combined with all variations from the above list. We will demonstrate that our methods are also able to solve CHC as a by-product.

The combination of shortest path with heat conduction gives rise to the question whether some of the combinatorial shortest path algorithms can be modified to solve this new problem. We will demonstrate in the sequel that this is only possible in a very special case. In the general case we are in the same situation as with the general shortest path problem with negative arc weights and negative cycles. Thus we cannot give a simple combinatorial algorithm for its solution. Instead we will formulate the problem as a mixed-integer linear program, which can be solved numerically within the general linear programming (simplex) based branch-and-cut framework (see Schrijver [14] or Nemhauser and Wolsey [11] for an introduction).

To formally state the coolest path (or coolest cycle) problem we introduce some more notations. Denote by F one of the objective functionals from above. Select two distinct nodes $v, w \in V$. Let $\mathcal{P}_{v,w}$ be the set of all paths from v to w in D. The

time for traversing arc $a \in P$ is denoted by τ_a. If we assume that the object Ω starts at time $t_0 := 0$ then the end time is $t^* := \sum_{a \in P} \tau_a$. Function $u(x,t)$ describes the heat distribution in the object at location x and time t, depending on path P. To be more precise, $t \mapsto u(x,t)$ depends only on those arcs that were traversed before time t, for all $x \in \Omega$ and $t \in [0,t^*]$. Note that each point in time $t \in [0,t^*]$ can be mapped onto an arc $a(t) \in P$ which the object traverses at time t.

Using this notation the problem can be formally stated as follows:

$$\min_{u;P \in \mathcal{P}_{v,w}} \quad F(u(x,t^*)) \tag{1}$$

$$\text{such that} \quad \frac{\partial u}{\partial t}(x,t) = k \cdot \frac{\partial^2 u}{\partial x^2}(x,t), \quad \forall\, x \in \Omega, \forall\, t \in [0,t^*], \tag{2}$$

$$\frac{\partial}{\partial n} u(x,t) = h \cdot (T_{a(t)}(x) - u(x,t)), \quad \forall\, x \in \partial\Omega, \forall\, t \in [0,t^*], \tag{3}$$

$$u(x,0) = u_0(x), \quad \forall\, x \in \Omega. \tag{4}$$

2. Mathematical Background

Before actually solving the problem at hand we start with a survey of the shortest path problem and the heat equation. From this study we can also show in which directions our methods can be extended.

2.1. **Shortest Paths in Graphs.** One major ingredient is the classical shortest path problem on (directed or undirected) graphs (SPP, for short). An instance of the SPP is defined by a directed weighted graph $D = (V, A, c)$, where $c : A \to \mathbb{R}$ are arc weights, and two distinct nodes $v, w \in V$. The cost (or length) of an v-w-path P is hereby defined as the sum of weights of its arcs, i.e., $c(P) := \sum_{a \in P} c_a$. The problem asks for an v-w-path P of minimal length. In this spirit the coolest path problem can be seen as a combination of a pure combinatorial problem with an objective function that takes the amount of absorbed heat along the path into account.

The mathematical study of the combinatorial shortest path problem in graphs can be dated back to the 1950s (see Schrijver [15]). There exists several algorithms for its solution.

The key observation that leads to *efficient*, i.e., polynomial time algorithms, is the property that all subpaths of a shortest path are as well shortest paths. However, this property holds if and only if the graph does not contain a negative cycle. In this case we can use an algorithm due to Moore [10], Bellman [2], and Ford [6], which has a running time proportional to the number of vertices cubed. In the special case that all edge weights are non-negative one can use Dijkstra's algorithm [4] which has only a quadratic running time.

In the general shortest path case negative weights (and negative circles) are allowed. There is no efficient combinatorial algorithm known in that case, and it is most likely that no such algorithm exists (unless $P = NP$). A special case of the shortest path with negative cycles is the longest path problem, which asks for the longest possible path (also called critical path) between two distinct nodes. More general than this, one can consider the path problem with given length, where a path is sought which connects the two nodes with a path of a prescribed length (or to decide that no such path exists). This problem is also NP-hard. Later on, this problem will occur as a subproblem in one of our solution methods for the CPP.

The solution of the corresponding linear program from above is still integral, but most likely cycles (with negative sum of its arcs) will occur. In order to obtain cycle-free solutions, one can use the following model. Let a weighted digraph $D = (V, A, c)$ with arbitrary arc weights $c_a \in \mathbb{R}$ for all $a \in A$ be given. Select two distinct nodes $v, w \in V$. We define a set $A^* := A \times \{1, \dots, |A|\}$ and introduce binary variables $z_{i,j,p} \in \{0, 1\}$ for all $(i, j, p) \in A^*$. If $z_{i,j,p} = 1$ then arc (i, j) is selected as the p-th arc in the v-w-path. Every arc of A can in principle occur in this v-w-path. Hence the number of elements in A, i.e., $|A|$, is an upper bound on the number of arcs in the path. Moreover we introduce binary variables $y_p \in \{0, 1\}$ for all $p \in \{1, \dots, |A|\}$, where $y_p = 1$ indicates that the path consists of exactly $(|A| - p)$ arcs.

Using these definitions the shortest path problem with arbitrary arc weights can be formulated as follows:

$$\text{(5)} \quad \min \sum_{(i,j,p)\in A^*} c_{ij} \cdot z_{i,j,p},$$

$$\text{(6)} \quad \text{s.t.} \sum_{i:(i,w)\in A} z_{i,w,|A|} = 1,$$

$$\text{(7)} \quad \sum_{j:(v,j)\in A} z_{v,j,p} = y_p, \quad \forall\, p \in \{1, \dots, |A|\},$$

$$\text{(8)} \quad \sum_{p\in\{1,\dots,|A|\}} y_p = 1,$$

$$\text{(9)} \quad \sum_{p\in\{1,\dots,|A|\}} z_{i,j,p} \leq 1, \quad \forall\, (i, j) \in A,$$

$$\text{(10)} \quad \sum_{i:(i,k)\in A} z_{i,k,p-1} = \sum_{j:(k,j)\in A} z_{k,j,p}, \quad \forall\, k \in V\backslash\{v, w\}, \forall\, p \in \{2, \dots, |A|\},$$

$$\text{(11)} \quad y_p \in \{0, 1\}, z_{i,j,p} \in \{0, 1\}, \quad \forall\, (i, j) \in A, \forall\, p \in \{1, \dots, |A|\}.$$

Constraint (6) forces the last arc of the path to end at node w. By constraints (6) the last arc of the path, which is the p-th arc within the path, connects node w. Exactly one arc is the first arc, which is modeled by (7) and (8). Constraints (9) ensure that every arc occurs at most once in the v-w-path. The connectivity of the paths is due to the flow conservation constraints (10).

We note, that it is possible to formulate this problem only using a set of variables that indicates whether an arc is in the path or not. In that case we can ensure the condition that no cycle occurs by adding suitable cut constratints to the model. This polyhedron has also recently been studied by Stephan [17]. However, for our later models we will need an explicit encoding of the order of the arcs in the path to help us compute the temperature distribution.

As a possible solution technique one can apply branch-and-bound or branch-and-cut, which re-introduces the integrality after its relaxation. This technique will be briefly described in the subsequent section.

This model can be replaced with a much simpler one if no negative cycles occur. We introduce variables

$$\text{(12)} \quad z_{i,j} \in \{0, 1\}, \quad \forall\, (i, j) \in A.$$

Then the shortest path problem can be formulated as the following integer program:

$$\min \quad \sum_{(i,j)\in A} c_{ij} \cdot z_{i,j}, \tag{13}$$

$$\text{such that} \quad \sum_{j:(v,j)\in A} z_{v,j} = 1, \tag{14}$$

$$\sum_{i:(i,w)\in A} z_{i,w} = 1, \tag{15}$$

$$\sum_{i:(i,k)\in A} z_{i,k} = \sum_{j:(k,j)\in A} z_{k,j}, \quad \forall\, k \in V\backslash\{v,w\}, \tag{16}$$

where $c_{i,j}$ are some arc weight coefficients. Constraints (14) and (15) ensure that the path starts in v and ends in w, respectively. Constraints (16) ensure that the path is leaving a node as many times as it was entered. These constraints are also called *flow conservation constraints*. The objective function (13) guarantees that the path is of minimum cost. Since there are by definition no cycles with negative costs in the digraph the objective ensures that the optimal path is simple, i.e., it has no node repetitions.

Despite being an integer program the above formulation of the shortest path problem can be solved by relaxing the integrality constraints (12) to its continuous counterpart

$$z_{i,j} \in [0,1], \quad \forall\,(i,j) \in A. \tag{17}$$

Since the constraint system (14), (15), and (16) is total unimodular, i.e., for every square submatrix of the constraint system its determinant is in $\{-1,0,1\}$, and the right-hand side is integral (if all variable terms are on the left-hand side, only 0 or 1 remains as constants on the right-hand side), all feasible solutions are automatically integral [11]. Hence the shortest path problem with non-negative weights can be solved by linear programming. From a computational point of view it is of course preferred to use one of the above mentioned special purpose algorithms (Dijkstra or Moore-Bellman-Ford) to solve the shortest path problem, instead of using a general purpose linear program solver.

2.2. **Solving Mixed-Integer Programs.** The usual way one follows when solving general integer programs

$$\min\{c^T x : Ax \leq b, x \in \{0,1\}^n\} \tag{18}$$

(where $c \in \mathbb{Q}^n, A \in \mathbb{Q}^{n\times m}, m, n \in \mathbb{N}$) is to relax the integrality constraints on the variables, and thus to replace it by their continuous counterparts, $x \in [0,1]^n$. In this way one obtains a linear program, which can be efficiently solved with Dantzig's simplex algorithm [3] or with Karmarkar's interior point method [8]. In either case one obtains a solution vector x^*, and $c^T x^*$ is a lower bound on the objective function value of the integer program. Now one of the following two cases will occur. Either the solution vector x^* is already integral, i.e., $x^*_{i,j} \in \{0,1\}$. In this case the solution is global optimal. Otherwise x^*_i is not integral for some index i. Then there are basically two methods to successively re-introduce the integrality constraints.

One can show that there always exists a linear inequality that separates x^* from the convex hull of all feasible solutions of (18). If one can find such an inequality (which in general is difficult both theoretically and in practice), then it is added to

the LP relaxation. Such an inequality is also called cutting plane, since it cuts off x^* from the set of feasible (integral) solutions. In this way we obtain a stronger LP relaxation which is again solved with a linear program solver. This procedure is carried out until no further cutting plane is found (or some other termination criterion is reached).

The other method is to select a fractional variable, say $x_i^* \notin \{0,1\}$. Then the entire problem is split into two sub-problems. In one sub-problem we enforce $x_i := 0$, and $x_i := 1$ in the other. So we now have two linear problems, which are solved separately. From there on, the procedure is iterated, which is called branch-and-bound algorithm.

If in addition cutting planes are used within branch-and-bound, then the whole procedure is called branch-and-cut algorithm. This approach is today the most successful way to numerically solve a general linear mixed-integer programming problem. Several computer codes are implementing this framework, such as the commercial XPress from Dash or CPlex from ILOG, or the academic codes SCIP from ZIB [1] or BCP from COIN-OR [9].

2.3. **Heat conduction.** The heat equation models heat conduction in a solid or a fluid at rest. It describes one of the three modes of energy transport. The other two are convection and radiation. The heat equation is based on Fourier's law which states that the heat flux is antiproportional to the temperature gradient. This leads to a so-called diffusion process, which is modeled by the partial differential equation for the unknown temperature u:

$$\frac{\partial}{\partial t}u(t,x) = k\Delta u(t,x). \tag{19}$$

The time evolution of the temperature is governed by the second-order space derivative of the temperature. The heat equation is the classic example of a parabolic partial differential equation. To obtain a well-posed problem, i.e., a uniquely solvable problem whose solution depends continuously on the data, one has to supplement the heat equation with an initial condition

$$u(0,x) = u_0(x)$$

and boundary conditions. The heat exchange of a body with an external reservoir can be modeled by the Robin type boundary conditions

$$\frac{\partial}{\partial n}u(t,x) = h(u_b(t,x) - u(t,x)),$$

which state that the normal derivative of the temperature at the boundary is proportional to the difference in temperatures. Altogether, this problem admits a unique solution, which, given appropriate boundary conditions, after an infinitesimally small time is arbitrarily smooth, i.e., infinitely differentiable in space.

The theory of the heat equation is very well-developed, and its properties are well-known, see Evans [5], for instance. Thus it often serves as a test case for both new theoretical or computational methods, and it will be our first model to investigate the interplay between discrete decisions and continuous processes.

There are several properties of the heat equation (and of partial differential equations in general) that might be of use when designing algorithms as in Section 3.2. For simplicity we assume $\Omega := [0,1]$, and thus $\partial\Omega = \{0,1\}$.

2.3.1. *Maximum principle.* Let u be a solution to $\frac{\partial}{\partial t}u(t,x) = k\frac{\partial^2}{\partial x^2}u(t,x)$ for $(t,x) \in]0,1[\times]0,1[$ and let u be continuous on the closure of this set. Then u attains its maximum and its minimum on the parabolic boundary $\Sigma_p = \{(t,x) : t = 0 \text{ or } x = 0,1\}$. This corresponds to the well-known fact that the maximal temperature cannot be greater than the maximal initial or boundary temperature.

2.3.2. *L^1-estimate.* If we integrate the heat equation over space and time and use integration by parts, we get

$$\int_0^1 u(1,x)dx = \int_0^1 u(0,x)dx + 2hu_b - h\int_0^1 \left(u(t,0) + u(t,1)\right) dt$$

Since the temperature is positive, we obtain an estimate for the average temperature

$$\int_0^1 u(1,x)dx \leq \int_0^1 u(0,x)dx + 2hu_b,$$

which says that the average temperature increases at most by $2hu_b$.

2.4. **Evolution Equations and Semigroup Theory.** There exist a variety of approaches to treat the heat equation (19). While there are other well-established methods and approaches, in the following we focus on semigroup theory. We only present basic ideas of the semigroup theory and refer the reader to [5, 12, 13] for more details.

We think of equation (19) as an initial-value problem for an ordinary differential equation in a suitable Banach space. Let

$$\frac{d}{dt}u = Au, \qquad u(0) = u_0, \tag{20}$$

where $X = L^2([0,1]^n)$ and let the operator A be a mapping $A : D(A) = \{u \in H^2([0,1]^n)\} \to X$ by $A = -k\Delta_x$. For fixed time t the value of the solution $u(t)$ is viewed as element in $D(A)$. Intuitively, one expects the solution to (20) to be

$$u(t) = \exp(At)u_0.$$

Semigroup theory now gives a meaning to $\exp(At)$ when A is an operator. The properties of the exponential $\exp(At)$ motivate the following definition of a semigroup [13, Chapter 11]:

Definition. Let X be a Banach space. A family $\{T(t)\}$ of bounded linear operators in X is called a *strongly continuous semigroup*, if $T(t+s) = T(t)T(s)$ and $T(0) = \mathrm{Id}$ and if for every x, $t \to T(t)\,x \in X$ is continuous.

The relation between the semigroup $T(t)$ and the exponential $\exp(At)$ for some operator A can be stated as follows. We expect that a strongly continuous semigroup generalizing the exponential fulfills $\frac{d}{dt}\exp(At) = \exp(At)A$. Also, we ask if any semigroup can be given as $\exp(At)$ for some operator A. In fact, for every strongly continuous semigroup $\{T(t)\}$ of bounded linear operators we can define the *infinitesimal generator* of the semigroup by

$$Ax = \lim_{h\to 0+} \frac{T(h)(x) - T(0)x}{h}$$

and the domain of A where the above limit exists is dense in X. Furthermore, it holds [13, Lemma 11.11]

$$\frac{d}{dt}T(t)x = T(t)Ax \;\forall x \in D(A). \tag{21}$$

Hence, whenever we have a semigroup $\{T(t)\}$ it behaves in the previous sense like $\exp(At)$ for A being the infinitesimal generator.

Using property (21) one proves that if $T(t)$ is a semigroup with generator A, then the initial-value problem (20) has a classical solution $u \in C^1([0,T];X) \cap C([0,T];D(A))$ for any $u_0 \in D(A)$ which is given by $u(t) = T(t)u_0$.

Finally, we need to discuss the following question: Given the partial differential equation (20), under which conditions is the operator A an infinitesimal generator of a semigroup? The answer is given by the Hille-Yosida theorem [13, Theorem 11.17]:

Theorem. *Let A be an operator in the Banach space X. Then A is the infinitesimal generator of a strongly continuous semigroup $T(t)$ satisfying $\|T(t)\|_X \leq M\exp(\omega t)$ if and only if $D(A)$ is dense in X, A is a closed operator and every $\lambda > \omega$ is in the resolvent set of A and $\|(A-\lambda I)^{-n}\| \leq M(\lambda-\omega)^{-n}$ for every n.*

If the previous conditions are satisfied, one typically denotes the semigroup $T(t)$ with generator A as $\exp(At)$. It can be proven that (20) satisfies the assumptions of the Hille-Yosida theorem and hence the initial-value problem for the heat equation is a well-posed problem with solution $u(t) = \exp(At)u_0$, where $\exp(At)$ is a strongly continuous semigroup (in fact, it is an analytic semigroup).

3. Solving the Coolest Path Problem

To solve the CPP we will transform it into a mixed-integer programming problem. We will discretize the heat equation and starting from this discretization we will derive two mixed-integer programming models based on the shortest path models described above. Of these models the second one will be clearly preferably to the first one which can also be seen in the computational results.

3.1. Semi-discretization.

As for the theory the numerics for the heat equation is well-known and simple schemes can be found in standard textbooks, e.g. [18]. Here, we apply a second-order finite difference approximation to discretize the Laplace operator and solve the resulting semi-discretize ordinary differential equation exact. Since we are only interested in the interplay of the discrete and the continuous dynamics, we do not investigate other numerical discretizations.

In order to make the coolest path problem solvable numerically, we use a semi-discretization in space using finite differences and solve exactly in time. To that end, we consider an equidistant grid $x_i = i \cdot \Delta x$ with a gridsize $\Delta x = \frac{1}{n}$ for $i \in N := \{0,\dots,n\}$. The function $u_i(t)$ approximates $u(t,x_i)$. This can be written as a system of ordinary differential equations (ODEs) in the form

$$\frac{d}{dt}u(t) = Au(t) + b(t), \tag{22}$$

where

$$A = \frac{k}{\Delta x^2}\begin{pmatrix} -1-h\Delta x & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -1-h\Delta x \end{pmatrix} \quad \text{and} \quad b = \frac{kh}{\Delta x}\begin{pmatrix} u_b \\ 0 \\ \vdots \\ 0 \\ u_b \end{pmatrix}.$$

Note that any semi-discretization of the heat equation can be written in the form (22). Furthermore, this formula is written in semi-group notation. In fact, the

discretization is only needed for the explicit computations, whereas the following algorithm can be formulated for semi-groups. If u_b does not depend on time, the solution to the ODE system is

$$u(t) = exp(At)u_0 + (exp(At) - I)\, A^{-1}b.$$

The solution at time $t = 1$ and thus the new temperature in the final node of the arc can be written as

$$u^{\text{new}} = Ru^{\text{old}} + Sb,$$

where $R = exp(A)$ and $S = (R - I)A^{-1}$.

3.2. **Modifying Combinatorial Shortest Path Algorithms.** To solve the coolest path problem we will try to transform it into a variant of the shortest path problem. We will propose two such variants later on. First, however, we will describe a case where we can give a particularly simple algorithm for a PDE-constrained path problem.

Proposition. *Assume the objective function F is a monotone functional and we are given initial values $l_s = u_0(x)$. Then an optimal assignment of values $l : V \to \mathbb{R}^\Omega$ is given by:*

$$l_w = \begin{cases} Rl_v + Sb, \text{ where } v = \operatorname{argmin}\{F(Rl_v + Sb) : (v, w) \in A\}, & w \neq s, \\ u_0(x), & w = s. \end{cases} \tag{23}$$

We can now use a variant of Dijkstra's shortest path algorithm to solve the coolest path problem. To this end, each node $i \in V\backslash\{v\}$ is labeled by a function $l_i(x) := +\infty$ for all $x \in \Omega$, and node v is labeled by the initial temperature $l_v(x) := u_0(x)$. Then the update formula for the re-labeling in each step of the algorithm has to be modified to

$$l_j := \begin{cases} Rl_i + Sb & F(Rl_i + Sb) \leq F(l_j) \\ l_j & \text{otherwise.} \end{cases} \tag{24}$$

3.3. **Shortest Path Based Mixed-Integer Model.** We will now direct our attention to the general coolest path problem. In this case we will need more general techniques. As a first approach we will try to adapt the integer programming model for the shortest path problem without negative cycles to our problem.

We introduce continuous non-negative variables for the space-discrete temperature distribution

$$u_{i,k} \in \mathbb{R}_+, \quad \forall\, i \in V, \forall\, k \in N. \tag{25}$$

Then we include the following constraint family, which originates from the semi-discretization of the heat equation:

$$|u_{j,k} - (Ru_i - Sb_{ij})_k| \leq M_{i,j,k} \cdot (1 - z_{ij}), \quad \forall\, (i, j) \in A, \forall\, k \in N. \tag{26}$$

Here $M_{i,j,k}$ is a large constant with $M_{i,j,k} > |u_{j,k} - (Ru_i - Sb_{ij})_k|$ for all possible temperature distributions for u_i and u_j. For example, one might set $M_{i,j,k} := 2 \cdot \max\{T_a(x) : x \in \Omega, a \in A\}$.

As objective function we select the minimization of the average end temperature:

$$\min \frac{1}{n+1} \sum_{k \in N} u_{w,k}. \tag{27}$$

Summing up, our first coolest path model is to minimize (27) subject to the constraints (14), (15), (16), and (26), the integrality constraints (12), and the non-negativity constraints (25).

The weak point of this model is the coupling of the heat distribution and the shortest path equations via the infamous big-M-constraints (26). Such constraints are known to lead to weak LP relaxations and other numerical difficulties. Due to the weak LP relaxation one has to examine many nodes in the branching tree, which leads to an almost full enumeration of all v-w-paths. Thus, the approach becomes computationally infeasible for more than, say, 10 nodes.

The second – vastly superior – approach is to model the coolest path problem as a general shortest path problem in a graph with negative cycles. We can derive such a formulation directly from the semi-discretization.

For a given time t^* and all arcs on the path so far, the temperature distribution at time t^* can be explicitly computed as $u_{t^*} = R^{t^*}u_0 + \sum_{0\le t<t^*} R^{t^*-t}Sb_{a(t)}$, where $a(t)$ is the arc chosen at time t. Thus, we need to take into account not only if an arc was used in a path, but also at which point in time it was used. For a linear objective functional F we obtain, using our introductory remark, the equation

$$F(u_{t*}) = F(R^{t^*}u_0) + \sum_{0\le t<t^*} F(R^{t^*-t}Sb_{a(t)}).$$

If we select as before the minimization of the average end temperature as the ultimate goal then the objective function of corresponding MIP-formulation is:

$$\min \ \frac{1}{n+1}\sum_t R^{t^*-t}u_0 y_t + \frac{1}{n+1}\sum_t\sum_{a\in A} R^{t^*-t}Sb_a z_{a,t}, \tag{28}$$

subject to the constraints (6), (7), (8), (9), and (10).

A slight modification of the latter model allows us to tackle the coolest Hamiltonian cycle problem which was alluded to above.

4. Computational Results

The second mixed-integer programming model can be used to tackle networks of sizes up to 70 nodes. We will first give some examples which show that we will not be able to drastically simplify the given model.

4.1. **A Small Example Network.** The network in Figure 1 serves as an example in which the coolest path or the optimal path in some sense is different from the shortest path. The nodes are numbered from 1 to 5. On each arc, the external temperature is shown. There are four paths connecting nodes 1 and 5, namely $1 \to 5$, $1 \to 3 \to 5$, $1 \to 2 \to 3 \to 5$, $1 \to 2 \to 4 \to 5$. Figure 2 shows the temperature distribution in every node for each of the paths. It is an attempt to show how the different boundary conditions shape the temperature profile. Depending on the objective, we get the following optimal paths:

Shortest path w.r.t. unit weights:	$1 \to 5$,
shortest path w.r.t. temperature weights:	$1 \to 3 \to 5$,
lowest maximum temperature at node 5:	$1 \to 3 \to 5$,
highest minimum temperature at node 5:	$1 \to 2 \to 4 \to 5$,
lowest average temperature at node 5:	$1 \to 2 \to 3 \to 5$,
highest average temperature at node 5:	$1 \to 5$,
lowest temperature gradient over the path:	$1 \to 2 \to 3 \to 5$, and $1 \to 3 \to 5$.

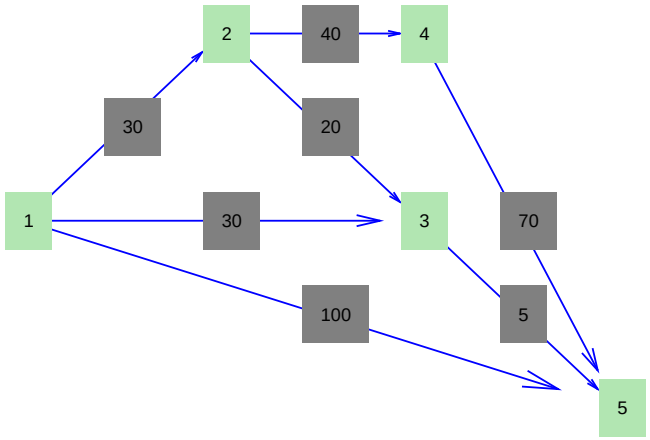


FIGURE 1. Example network.

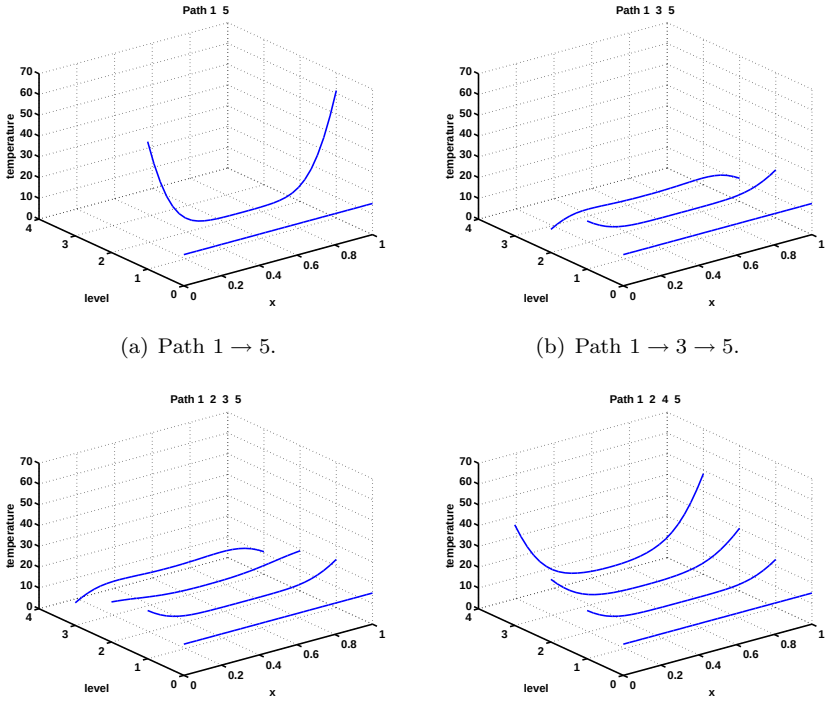
TABLE 1. Parameters of the random graphs

$\|V\|$	20	30	40	50	60	70
$\|E\|$	80	200	350	500	900	1200

4.2. **Medium Scale Networks.** As basis for our computations we only look at the case that we want to minimize the average temperature at the sink of the network. To compare the different modelling approaches we used a testbed of randomly generated graphs. The number of vertices and edges can be found in Table 1. In all the cases we used 31 discretization points in space and set $h := 10^{-2}$ and $k := 10$. The temperatures were chosen randomly between 20 and 200. The starting temperature was set to 110. The naive model cannot solve instances with 20 vertices, because the lower bounds that are produced by the relaxations are extremely bad. With the second model, we can treat examples of up to 70 vertices in reasonable time. The combinatorial problems then start to take over. From Figure 3 one can see that the time needed to solve an instance grows roughly exponentially with the instance size. However, with growing size of the graph, the variability between the instances becomes much bigger, which hints at the fact that the combinatorial problems become more difficult in general.

In Figure 4 we show a detailed comparision of 100 random networks with 50 vertices and 500 edges each. The running times for the instances vary greatly. After only a modest 98 seconds half the instances have been solved to optimality. However, it takes 955 seconds until the last instance has been solved.

4.3. **A Large Example Network.** With the methods presented in this article, and the use of modern MILP solver codes and recent computer hardware, we are able to solve instances of the coolest path problem having about 100 nodes. The



(a) Path $1 \to 5$.

(b) Path $1 \to 3 \to 5$.

(c) Path $1 \to 2 \to 3 \to 5$.

(d) Path $1 \to 2 \to 4 \to 5$.

FIGURE 2. Temperature distributions in the nodes of the four paths of the example network, form the first node (level 0) to the final node.

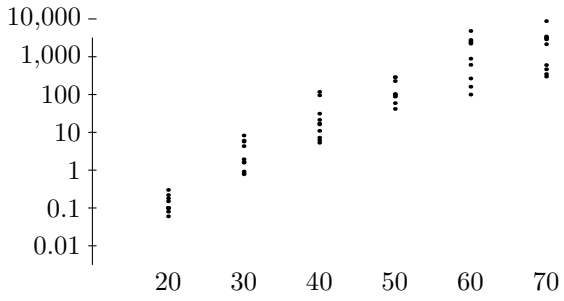


FIGURE 3. Running time for random samples

example shown in Figure 5 needed about four weeks of computation on a 2.4Ghz AMD Opteron computer with 32GByte RAM. From the 8 available CPU cores the MILP solver Cplex11 used one (i.e., we only had a single-core license of this software). To compute the matrix exponential we used the library Expokit (cf. [16]). The branch-and-bound tree cosists of approximately 2.4 mill. nodes, among them 14 nodes with feasible solutions. At its peak, there were 828 000 unresolved nodes in the tree, needing 8.3 GByte storage memory. The overall computation

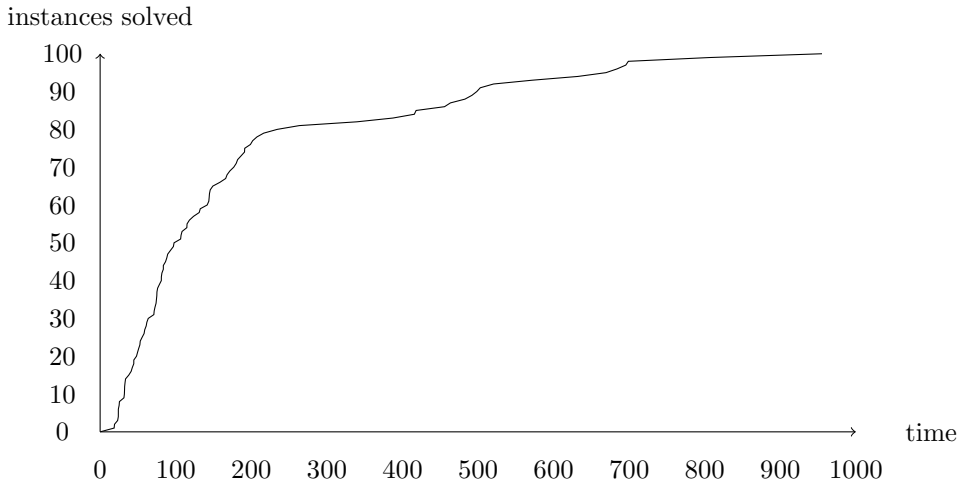


FIGURE 4. Comparison of running times for networks with 50 vertices and 500 edges (time given in seconds)

time was 2.2 mill. seconds (25 days). The solid lines in Figure 5 indicate the coolest path (with respect to a certain initial temperature of the traversing object). The path starts in the top-left corner and ends at the vertex in the bottom-right corner. It consists of 64 arcs. The widths of the arcs in the network are scaled with respect to their temperature: thin lines are "cool", thick lines correspond to "hot" arcs.

Informally speaking, one can see the following overall "behaviour" of an optimal solution to the coolest path problem: If the initial temperature is low in comparison with the temperatures of the arcs, then the coolest path usually is short in the sense that it consists of few arcs. Vice versa, if the initial temperature of the object is relatively high, then each arc is cooling it down. Hence the coolest path will then consist of many arcs. For initial temperature that are in the same range as the temperatures within the network the path will be of "average length" (as shown in our example in Figure 5). Moreover one can observe that the coolest path might well include some "hot" arcs, as long they are at the beginning of the path, so that the increase in the object's temperature can still be compensated towards the end. Finally, we observed that the solution times of the numerical MILP solver CPlex are higher the longer the coolest path is.

In Figure 6 we show the relative gap between the current best primal solution and the current known lower bound on the optimal value after n nodes have been processed. One can clearly see the exponential relation between the number of nodes and the decrease of the relative gap.

5. Conclusions and Outlook

In this article we introduced the coolest path problem, which is in part a combinatorial MIP and in part a continuous PDE problem. We demonstrated that an understanding of both worlds is necessary for its solution. With our techniques we were able to solve problem instances with up to 100 nodes to proven global optimality.

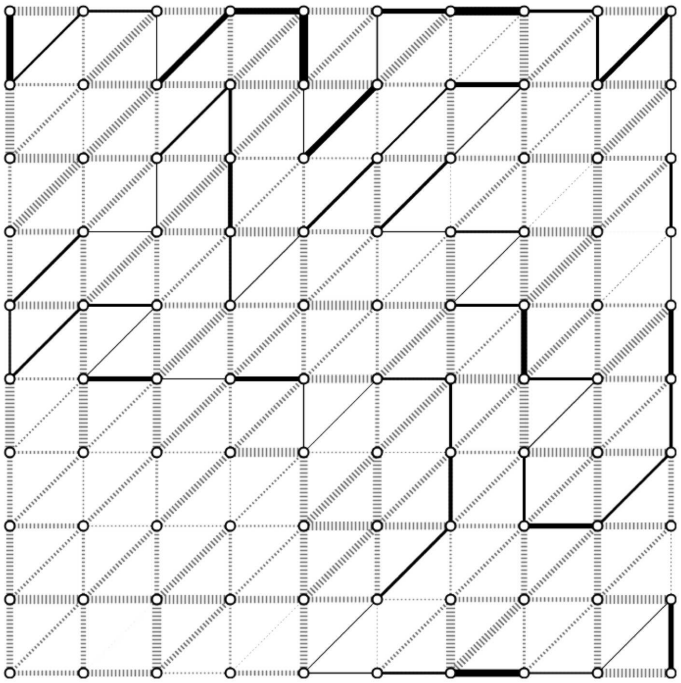


FIGURE 5. A 10×10 network with random temperatures, and a coolest path.

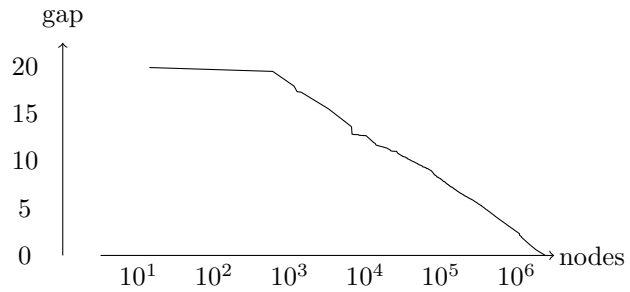


FIGURE 6. Computation of the 10×10 network: remaining gap (in percent) after n nodes

There are a number of possibilities to further speed up the solution process. One can imagine that specialized cutting planes or branching rules can lead to a reduction of the solution times. On the other hand the structure of the mixed-integer problem formulation allows a decomposition into a shortest path problem and additional set packing constraints. Thus one can apply a Langrangian relaxation scheme to compute the value of the LP-relaxation by solving a couple of shortest-path problems on the time-expanded network each of which can be solved in time $(|V|^2 + |V| \cdot |A|)$. The master problem of the relaxation can then be used using a bundle or a subgradient method. Although the linear relaxation are expected

to be solved faster, the combinatorial structure of the problem will be always the bottleneck for larger instances.

Acknowledgements

The main work on this article was done during a research stay of the authors at the Hausdorff Institute in Bonn. This work has further been supported by the DAAD research grants no. D/06/28176, D/06/19852, HE5386/5-1, and Seed Funds (RWTH Aachen, 2009).

References

[1] T. Achterberg, SCIP - a framework to integrate Constraint and Mixed Integer Programming, Technical Report ZR-04-19, ZIB, 2004.
[2] R.E. Bellman, On a Routing Problem, Quarterly of Applied Mathematics, Vol. 16(1), pp. 87 – 90, 1958.
[3] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, 1963.
[4] E.W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Numerische Mathematik 1, pp. 269 – 271, 1959.
[5] L.C. Evans, Partial Differential Equations, AMS, 1999.
[6] L.R. Ford, Network flow theory, Technical Report P-923, The Rand Corporation, Santa Monica, 1956.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[8] N. Karmarkar, A New Polynomial Time Algorithm for Linear Programming, Combinatorica, Vol. 4(4), pp. 373 – 395, 1984.
[9] F. Margot, BAC: A BCP based Branch-and-Cut Example, IBM Research Report RC22799 (W0305-064), 2003, revised 2008.
[10] E.F. Moore, The shortest path through a maze, Procedings of the International Symposium on the Theory of Switching, Havard University Press, pp. 285 – 292, 1959.
[11] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience, 1999.
[12] A. Pazy, Semigroups of Linear Operators and Applications to Partial Differential Equations, Springer, 1983.
[13] M. Renardy, R.C. Rogers, An Introduction to Partial Differential Equations, Springer, 1996.
[14] A. Schrijver, Theory of Linear and Integer Programming, Wiley, 1998.
[15] A. Schrijver, On the history of combinatorial optimization (till 1960), in: "Handbook of Discrete Optimization" (K. Aardal, G.L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam, pp. 1–68, 2005.
[16] R.B. Sidje, Expokit: a software package for computing matrix exponentials, ACM Transactions on Mathematical Software, Vol. 24(1), pp. 130–156, 1998
[17] R. Stephan, Facets of the $(s,t)-p$-path polytope. Online available at `arxiv:math/0606308v1`.
[18] R. Stoer, D. Burlisch, Numerische Mathematik, Springer, 2000.