NETWORKS AND HETEROGENEOUS MEDIA
©American Institute of Mathematical Sciences
Volume **5**, Number **1**, March **2010**

doi:10.3934/nhm.2010.5.143

pp. **143–162**

THE COOLEST PATH PROBLEM

Martin Frank

Rheinisch-Westfälische Technische Hochschule Aachen
Department of Mathematics & Center for Computational Engineering Science
Schinkelstraße 2, 52056 Aachen, Germany

Armin Fügenschuh

Konrad-Zuse-Zentrum für Informationstechnik (ZIB)
Department of Optimization
Takustraße 7, 14195 Berlin, Germany

Michael Herty

Rheinisch-Westfälische Technische Hochschule Aachen
Department of Mathematics
Templergraben 55, 52056 Aachen, Germany

Lars Schewe

Technische Universität Darmstadt
Department of Mathematics
Dolivostraße 15, 64293 Darmstadt, Germany

(Communicated by Kenneth Karlsen)

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 with inherent 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.

Our research is motivated by the fact that both worlds can inspire the respective other by sharing ideas and methods. To start the discussion at some point we

2000 *Mathematics Subject Classification.* Primary: 35K05, 90C10; Secondary: 90B10.
Key words and phrases. Heat Equation, Integer Programming, Shortest Path.

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*.

With our presentation we intend to address readers from both the continuous and the discrete world. That means that we sometimes present ideas and methods from one field that are so classical that they would never be mentioned in any recent research paper (for example, Dijkstra's shortest path algorithm or the derivation of the heat equation). The reader should keep in mind that what he believes is classical for him, is unknown for a reader from the respective "other side".

1. The coolest path problem.

We consider the following problem. Given is a directed graph $D = (V, E)$ with vertex set V and arc set E, 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 $\bar{u}_P := \frac{1}{vol(\Omega)} \int_\Omega u_P \, d\lambda$ 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 later modeled 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. Other possible applications are in the area of routing biomedical sensor in a thermal-sensitive environment such as the human body [3], or the motion trajectory planning for robots under thermal constraints [18].

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 once. Finding a HC in a graph is NP-complete, see the monograph by Garey and Johnson [16]. 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 remark that our methods are also able to solve CHC by identifying start- and end-node of the path.

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 [34] or Nemhauser and Wolsey [28] for an introduction).

2. **Mathematical background.**

Before actually solving the problem at hand we start with a survey of the shortest path problem and the heat equation.

2.1. **Shortest paths in graphs.**

One major ingredient in the coolest path problem 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 $D = (V, E, c)$, where $c : E \to \mathbb{R}$ 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.

The mathematical study of the combinatorial shortest path problem in graphs can be dated back to the 1950s (see Schrijver [35]). There exists several *efficient*, i.e., polynomial time, algorithms for its solution. The key observation that leads to efficient 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 [26], Bellman [7], and Ford [14], 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 [12] which has only a quadratic running time. Dijkstra's algorithm returns two functions. Function l yields for each node $s \in V$ the length of a shortest path from the designated start node v to node s, and $p(s)$ is the successor node of s on such a path. Note that Dijkstra's algorithm implicitly computes shortest path from v to all other nodes of the graph, and not only to a single destination node.

```
dijkstra($V, E, c, v$)
Input:  node set $V$, arc set $E$, weights $c \geq 0$, start node $v$
(1)   Let $l(v) := 0, l(s) := \infty$ for all $s \in V \backslash \{v\}, W := \emptyset$
(2)   Repeat
(3)       Let $s :=$ argmin$\{l(r) : r \in V \backslash W\}$
(4)       Let $W := W \cup \{s\}$
(5)       For Each $r \in V \backslash W$ with $(s, r) \in E$ Do
(6)           If $l(r) > l(s) + c_{s,r}$ Then
(7)               Let $l(r) := l(s) + c_{s,r}$
(8)               Let $p(r) := s$
(9)           End If
(10)      End For
(11)  Until $W = V$
Output: functions $l : V \to \mathbb{R} \cup \{\infty\}, p : V \to V$
```

Many combinatorial optimization problems can be formulated as integer or mixed-integer linear or nonlinear programming problems. For the shortest path problem we introduce variables $z_{i,j} \in \{0,1\}$ for all $(i,j) \in E$, with $z_{i,j} = 1$ if and only if arc (i,j) is used somewhere in the v-w-path. If all arcs have non-negative weight, $c_{i,j} \geq 0$ for all $(i,j) \in E$, then the shortest path problem can be formulated as the following integer program:

$$\min \quad \sum_{(i,j)\in E} c_{ij} \cdot z_{i,j}, \tag{1a}$$

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

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

$$\sum_{i:(i,k)\in E} z_{i,k} = \sum_{j:(k,j)\in E} z_{k,j}, \ \forall\, k \in V \backslash \{v, w\}, \tag{1d}$$

$$z_{i,j} \in \{0,1\}, \ \forall\, (i,j) \in E, \tag{1e}$$

where $c_{i,j}$ are some arc weight coefficients. Constraints (1b) and (1c) ensure that the path starts in v and ends in w, respectively. Constraints (1d) 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 (1a) 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 (1e) to its continuous counterpart

$$z_{i,j} \in [0,1], \ \forall\, (i,j) \in E. \tag{2}$$

Since the constraint system (1b), (1c), and (1d) 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 [28]. 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.

In the general shortest path case negative weights (and negative circles) are allowed. There is no efficient combinatorial algorithm known in that case. 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 (1) would still be integral, but in general subcycles with negative sum of its arcs that are not connected to the path will occur. In order to obtain subcycle-free solutions, one can use the following model. Let a weighted digraph $D = (V, E, c)$ with arbitrary arc weights $c_a \in \mathbb{R}$ for all $a \in E$ be given. Select two distinct nodes $v, w \in V$. We introduce binary variables $y_p \in \{0, 1\}$ for all $p \in \{0, \dots, |E|-1\} =: E^*$, where $y_p = 1$ indicates that the path consists of exactly $(|E| - p)$ arcs. To this end, we have to ensure by imposing a suitable constraint that exactly one of these variables is set to one and all others are zero. Denote by p_0 the unique index with $y_{p_0} = 1$. We define a set $\mathcal{E} := E \times E^*$ and introduce binary variables $z_{i,j,p} \in \{0, 1\}$ for all $(i, j, p) \in \mathcal{E}$. If $z_{i,j,p} = 1$ then arc (i, j) is selected as the $(p - p_0)$-th arc in the v-w-path. Every arc of E can in principle occur in this v-w-path. Hence only the number of elements in E, i.e., $|E|$, is an upper bound on the number of arcs in the path.

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

$$\min \quad \sum_{(i,j,p)\in\mathcal{E}} c_{ij} \cdot z_{i,j,p}, \tag{3a}$$

$$\text{s.t.} \quad \sum_{i:(i,w)\in E} z_{i,w,|E|} = 1, \tag{3b}$$

$$\sum_{j:(v,j)\in E} z_{v,j,p} = y_p, \ \forall\, p \in E^*, \tag{3c}$$

$$\sum_{p\in E^*} y_p = 1, \tag{3d}$$

$$\sum_{p\in E^*} z_{i,j,p} \le 1, \ \forall\, (i,j) \in E, \tag{3e}$$

$$\sum_{i:(i,k)\in E} z_{i,k,p-1} = \sum_{j:(k,j)\in E} z_{k,j,p}, \ \forall\, k \in V\backslash\{v,w\}, \forall\, p \in E^*\backslash\{0\}, \tag{3f}$$

$$y_p \in \{0,1\}, z_{i,j,p} \in \{0,1\}, \ \forall\, (i,j) \in E, \forall\, p \in E^*. \tag{3g}$$

Constraint (3b) forces the last arc of the path to end at node w. By constraints (3b) 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 (3c) and (3d). Constraints (3e) 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 (3f).

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 constraints to the model. This polyhedron is studied in Stephan [37]. However, for our later CPP models we will need an explicit encoding of the order of the arcs in the path to compute the temperature distribution.

As a possible solution technique for the mixed-integer problem (3) 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 next section.

2.2. **Solving mixed-integer linear programs.** The usual way one follows when solving general mixed-integer programs

$$\min\{c^T x : Dx \leq d, x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p\}, \tag{4}$$

where $c \in \mathbb{Q}^n, D \in \mathbb{Q}^{n\times m}, d \in \mathbb{Q}^m, m, n, p \in \mathbb{N}, 1 \leq p \leq n$, is to relax the integrality constraints on the variables, and thus to replace it by their continuous counterparts, $x \in \mathbb{R}^n$. In this way one obtains a linear program, which can be efficiently solved with Dantzig's simplex algorithm [11] or with Karmarkar's interior point method [19]. Due to the progress in the last decades in terms of faster hardware and better implementations of the algorithms it is today possible to routinely solve LPs with up to several million constraints and variables to proven global optimality in reasonable time [8]. From solving the LP relaxation one obtains that the LP relaxation is unbounded or infeasible, or a solution vector x^* is returned, and $c^T x^*$ is a lower bound on the objective function value of the integer program. In the first case, the solution process can be terminated, since the integer program (4) is also unbounded or infeasible. In the latter case one of the following two cases will occur. Either the solution vector x^* is already integral where required, i.e., $x^* \in \mathbb{Z}^{n-p} \times \mathbb{R}^p$. 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 (4). 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). Various families of cutting planes are described in the literature. Some apply to general mixed-integer problems, such as Gomory's mixed-integer cuts [6, 17], mixed-integer rounding cuts [23, 24, 27], or lift-and-project cuts [4, 5]. Other families of cuts only apply for special structured problems, such as flow cover inequalities for capacitated network flow problems [31], cover inequalities for knapsacks [10], or cuts for set packing problems, see [9] for a survey. In any case, adding cuts might lead to numerical problems, see the discussion in [29].

The other method is to select a fractional variable $x_i^* \notin \mathbb{Z}$. Then the entire problem is split into two sub-problems. In one sub-problem we enforce $x_i := \lfloor x_i^* \rfloor$, and $x_i := \lceil x_i^* \rceil$ in the other. Then we obtain two linear problems, which are solved separately. From there on, the procedure is iterated, which is called branch-and-bound algorithm. Branch-and-bound with LP relaxations for bounding were first used by Land and Doig [20], and are refined ever since, see [21, 22] for a survey.

If in addition cutting planes are used within branch-and-bound, then this hybrid procedure is called branch-and-cut algorithm. Among the first applications of this method was a solution of a 532-city instance of the traveling salesman problem by Padberg and Rinaldi [30]. 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 [2] or CBC and BCP from COIN-OR [15, 25]. However it should be mentioned that solving IPs and MIPs belongs to the class of NP-hard problems, and there are also problem instances with just a single constraint and less then ten variables that are not solvable with these codes in reasonable time [1].

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. For simplicity we assume $\Omega := [0,1]$, and thus $\partial\Omega = \{0,1\}$ for the remainder of this article. Think of Ω as a long, thin, solid rod made of homogeneous material (glass or metal, for example). We use a scalar variable $x \in [0,1]$ for the position of a point of the rod. The rod is very thin and perfectly isolated on its side, hence its physical properties only depend on the position x and time $t \geq 0$. We describe the thermal state using the following functions: $u(x,t)$ is the internal heat energy density per unit length (or temperature, for short) of the rod at position x and time t, and $q(x,t)$ is the rate of heat flow in positive x-direction at x and time t. One of the most fundamental physical assumption is energy conservation, which says that for each interval $[a,b] \subseteq \Omega$ and each time interval from t_1 to t_2 the heat energy can only enter or leave through its boundary $\{a,b\}$:

$$\int_a^b u(x,t_2)\,dx = \int_{t_1}^{t_2} q(a,t)\,dt + \int_a^b u(x,t_1)\,dx - \int_{t_1}^{t_2} q(b,t)\,dt. \tag{5}$$

From the fundamental theorem of calculus we first obtain

$$u(x,t_2) - u(x,t_1) = \int_{t_1}^{t_2} \frac{\partial}{\partial t} u(x,t)\,dt, \tag{6a}$$

$$q(b,t) - q(a,t) = \int_a^b \frac{\partial}{\partial x} q(x,t)\,dx, \tag{6b}$$

and then by an integration

$$\int_a^b u(x,t_2) - u(x,t_1)\,dx = \int_a^b \int_{t_1}^{t_2} \frac{\partial}{\partial t} u(x,t)\,dt\,dx, \tag{7a}$$

$$\int_{t_1}^{t_2} q(b,t) - q(a,t)\,dt = \int_{t_1}^{t_2} \int_a^b \frac{\partial}{\partial x} q(x,t)\,dx\,dt. \tag{7b}$$

Together with energy conservation (5) we get from this

$$\int_a^b \int_{t_1}^{t_2} \frac{\partial}{\partial t} u(x,t)\,dt\,dx = -\int_{t_1}^{t_2} \int_a^b \frac{\partial}{\partial x} q(x,t)\,dx\,dt. \tag{8}$$

Since (8) is valid for all a,b and all t_1,t_2 with $a \leq b$ and $t_1 \leq t_2$, we obtain from (8) using an elementary property from real integration theory that the integrants have to be equal (assuming that the integrants are continuous),

$$\frac{\partial}{\partial t} u(x,t) = -\frac{\partial}{\partial x} q(x,t). \tag{9}$$

Now Fourier's law comes into play which states that the heat flux is antiproportional to the temperature gradient, that is,

$$q(x,t) = -k \frac{\partial}{\partial x} u(x,t). \tag{10}$$

Here k is a material dependent constant called thermal diffusivity. Putting (9) and (10) together leads to a partial differential equation for the unknown temperature u, called the (one dimensional) heat equation:

$$\frac{\partial}{\partial t} u(t,x) = k \frac{\partial^2}{\partial x^2} u(t,x) = k \Delta u(t,x), \tag{11}$$

where Δ is the so-called Laplace operator. We remark that the derivation of the heat equation can be generalized to two and three dimensions.

In some cases, linear partial differential equations with second derivates (also called second-order PDEs) can be classified as parabolic, hyperbolic, or elliptic, depending on the coefficients in front of the derivates. This classification gives a first initial idea on how difficult it is to find a solution, and how smooth a solution might be. 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

$$u(0,x) = u_0(x) \tag{12}$$

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

$$\frac{\partial}{\partial n} u(t,x) = h(u_b(t,x) - u(t,x)), \tag{13}$$

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 [13], 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.

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

$$\int_0^1 u(1,x)dx = \int_0^1 u(0,x)dx + 2hu_b - h \int_0^1 (u(t,0) + u(t,1))\, dt \tag{14}$$

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

$$\int_0^1 u(1,x)dx \leq \int_0^1 u(0,x)dx + 2hu_b, \tag{15}$$

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

3. **Solving the coolest path problem.** To formally state the coolest path (or coolest cycle) problem we introduce some more notations. Denote by F one of the objective functionals mentioned in the introduction. 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:

$$\min_{u;P\in\mathcal{P}_{v,w}} \quad F(u(x,t^*)) \tag{16}$$

$$\text{such that} \quad \frac{\partial u}{\partial t}(x,t) = k \cdot \frac{\partial^2 u}{\partial x^2}(x,t),\ \forall\, x \in \Omega, \forall\, t \in [0,t^*], \tag{17}$$

$$\frac{\partial}{\partial n}u(x,t) = h \cdot (u_b(x) - u(x,t)),\ \forall\, x \in \partial\Omega, \forall\, t \in [0,t^*], \tag{18}$$

$$u(x,0) = u_0(x),\ \forall\, x \in \Omega. \tag{19}$$

To solve the CPP we will transform it into a mixed-integer programming problem. We will first discretize the heat equation. We focus then on a special case of the CPP by restricting the objective functional. In this case we can actually solve the CPP using a modification of Dijkstra's algorithm. For the general CPP, however, we use variants of the mixed-integer models described above. We propose two models, one that can be directly derived from a classical shortest path formulation for non-negative arc weights and the second one derived from the general shortest path model. Of these models the second one will be clearly preferable to the first one which can also be seen in the computational results.

3.1. **Semi-discretization.** Similar to the theory, also the numerics for the heat equation is well-known and simple schemes can be found in standard textbooks, e.g. [38]. 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.

Here, we apply a second-order finite difference approximation to discretize the Laplace operator. Since we are only interested in the interplay of the discrete and the continuous dynamics, we do not investigate other numerical discretizations. We consider an equidistant grid $x_i = i \cdot \Delta x$ with a gridsize $\Delta x = \frac{1}{n}$ for $i \in N := \{0,\ldots,n\}$. 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

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

where

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

and with a slight abuse of notation

$$u_0 = (u_0)(x_i) \in \mathbb{R}^{n+1},\ u_b = u_b(x), x \in \partial\Omega.$$

Note that any semi-discretization of the heat equation can be written in the form (20). If u_b does not depend on time, the solution to the ODE system is

$$u(t) = \exp(At)u_0 + \int_0^t \exp(A(t-s))bds = \exp(At)u_0 + (\exp(At) - I)\, A^{-1}b. \quad (21)$$

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

$$u^{\text{new}} = Ru^{\text{old}} + Sb, \quad (22)$$

where $u^{\text{old}} \equiv u_0$ and

$$R = \exp(A) \text{ and } S = (R - I)A^{-1}. \quad (23)$$

The previously introduced matrices R and S will be used in Dijksta's algorithm below. The setting of our problem (CPP) is such that each arc $(r, s) \in E$ has unit length. Furthermore, we have assumed that the object travels at a constant speed along this arc and such that at time $t = 1$ the object reached the node s, if it started at $t = 0$ in r. Hence, formula (22) gives the discretized temperature u^{new} at node s for an object which has started at node r with temperature $u_{0,i} = u^{old}$ and which has traveled along the single arc (r, s). During this time it has been heated according to u_b (entering in (22) through the vector b). If we denote the temperature at node r by u^{old}, the temperature at node s is given by (22). If we have a path P given, the temperature at the terminal node is obtained by subsequent application of the operators R and S using equation (22). Provided that we have a comparison principle we can state a modification of Dijkstra's algorithm in order to find a 'shortest path' as detailed in section 3.2. The computation of the temperature using the previous matrices enters there in step (7) of the algorithm where the new 'costs' for a single connection (r, s) is computed.

Remark 1. A generalization of the previous discussion can be given in the framework of semi-groups theory [13, 32, 33]. In fact, in the above computations do not depend on the particular discretization in space of the heat equation or on the discretization at all. Assume that we have an operator equation

$$\frac{d}{dt}u = Au + b,\ u(0) = u_0. \quad (24)$$

where $A : X \to Y$ is a linear operator defined on a set $D(A)$ being a subset of a Banach space X. Let Y be another Banach space and $u_0 \in X$ and $b \in Y$. Then, the previous computations are based on formula (21). A similar formula can be established in this more general setting using the theory of semi-groups. To be more precise, if we assume that A is the infinitesimal generator [33, Chapter 11] of a strongly continuous semi-group $T(t)$, then, equation (24) admits a mild solution [33, Chapter 11] given by

$$u(t) = T(t)u_0 + \int_0^t T(t-s)bds \quad (25)$$

The solution belongs to the space $C(0, 1; X)$, i.e., continuous in time with values in X. Formally, we may write $T(t) = \exp(At)$ and it can be shown that T satisfies the functional equation $\frac{d}{dt}T(t)x = AT(t)x$ for all $x \in D(A)$. Hence, redefining $R = T(1)$ and $S = \int_0^1 T(t-s)ds$ we may apply the same techniques as in the subsequent section to operator equations of the type (24) under the assumption that A is the infinitesimal generator of a semi-group $T(t)$ on a Banach space X.

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. Indeed, in very special cases it is possible to generalize Dijkstra's algorithm to problems of our class. Sufficient for this is that we are given an objective functional that is monotone on the trajectory of the solution, meaning that

$$F(u^1) \leq F(u^2) \quad \text{implies} \quad F(Ru^1 + Sb) \leq F(Ru^2 + Sb).$$

Proposition 1. *Assume the objective function F for the coolest path problem (cf. p. 151) is a monotone functional and we are given initial values u_0 for the temperature of Ω at the start node v. Then an optimal assignment of values $u : V \times \Omega \to \mathbb{R}$ is given by:*

$$u_r = \begin{cases} Ru_s + Sb_{s,r}, \ \textit{where } r = \operatorname{argmin}\{F(Ru_s + Sb_{s,r}) : (s,r) \in E\}, & r \neq v, \\ u_0, & r = v. \end{cases} \tag{26}$$

Note that we write $b_{i,j}$ for an encoding of the temperature on arc (i,j). The idea behind this proposition is that when using a functional that is monotone in the sense defined above, we can obtain bounds on the distances of the nodes in the network simply by looking at the functional values instead of the temperature distributions.

Example. A nontrivial example for a monotone objective function F can be constructed as follows. Assume that F is linear. Then the question of finding a monotone functional can be reformulated as: Find F such that

$$F(u) \leq 0 \quad \text{implies} \quad F(Ru) \leq 0. \tag{27}$$

Then it is straightforward to prove: If $G(Rx) = \lambda G(x)$ (i.e. G is an eigenvector of the adjoint of R) with positive λ, then the functional defined by G is monotone in the sense above.

If R is a matrix, then possible G's are $G(x) = v^T x$, where v is a left eigenvector of R. On the function space $L^2(0,1)$, we can write the functional as $G(x) = \int_0^1 v(t)x(t)dt$, where the weight v again is an eigenfunction of the adjoint operator of R. A possible weight function (or vector if you will) for the heat conduction problem is shown in Figure 1. Here points in the interior have a higher value than points closer to the boundary of $\Omega = [0,1]$.

We can then use the following variant of Dijkstra's shortest path algorithm to solve the coolest path problem:

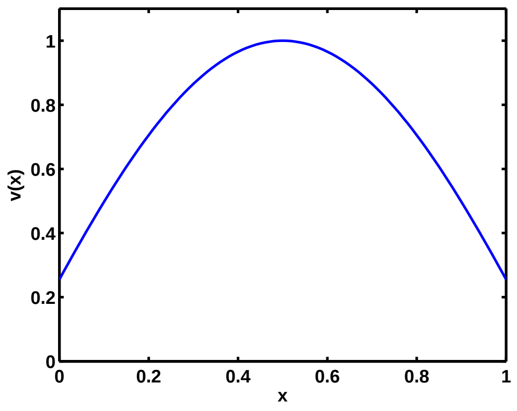


FIGURE 1. Weight function for monotone functional.

```
coolestPathDijksta$(V, E, \Omega, u_0, b, v, F, R, S)$
   Input:  node set $V$, arc set $E$, object $\Omega$, initial temp. $u_0$, arc temp. $b$,
           start node $v$, monotone objective $F$, matrices $R, S$ defined in (23)
     (1)   Let $u_v(x) := u_0(x), u_s(x) := \infty$ for all $s \in V \setminus \{v\}, x \in \Omega$, and $W := \emptyset$
     (2)   Repeat
     (3)       Let $s := \text{argmin}\{F(u_r) : r \in V \setminus W\}$
     (4)       Let $W := W \cup \{s\}$
     (5)       For Each $r \in V \setminus W$ with $(r, s) \in E$ Do
     (6)           If $F(u_r) > F(Ru_s + Sb_{s,r})$ Then
     (7)               Let $u_r := Ru_s + Sb_{s,r}$
     (8)               Let $p(r) := s$
     (9)           End If
    (10)       End For
    (11)   Until $W = V$
  Output:  functions $u : V \times \Omega \to \mathbb{R} \cup \{\infty\}, p : V \to V$
```

Here the output of the coolest path version of Dijkstra's algorithm are functions u_s for every node $s \in V$ that describe the temperature for all $x \in \Omega$ at this node s on a shortest path starting in v.

3.3. **Shortest path based mixed-integer model.** We will now direct our attention to the general coolest path problem, that is, the CPP for objective functionals that can also be nonmonotone. 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

$$u_{i,k} \in \mathbb{R}_+, \ \forall\, i \in V, \forall\, k \in N. \tag{28}$$

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

$$|u_{j,k} - (Ru_i + Sb_{ij})_k| \leq M_{i,j,k} \cdot (1 - z_{ij}), \ \forall\, (i,j) \in E, \forall\, k \in N. \tag{29}$$

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 u_i and u_j. For example, one might set

$$M_{i,j,k} := 2 \cdot \max\{T_a(x) : x \in \Omega, a \in E\}. \tag{30}$$

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

$$\min \frac{1}{n+1} \sum_{k \in N} u_{w,k}. \tag{31}$$

Summing up, our first coolest path model is

$$\begin{aligned} &\text{minimize} && (31), && (32a)\\ &\text{subject to} && (1b), (1c), (1d), (1e), (28), \text{ and } (29). && (32b)\end{aligned}$$

The weak point of this model is the coupling of the heat distribution and the shortest path equations via the infamous big-M-constraints (29). Such constraints are known to lead to weak LP relaxations and numerical difficulties. The weak relaxations of the LP solution are rooted in the fact that in an optimal LP solution the value of z_{ij} is chosen in such a way that equality in (29) holds. In most cases the optimal value will be strictly less than 1. This does typically not change dramatically by the introduction of standard cutting planes. 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 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 $u_{t^*} = R^{t^*} u_0 + \sum_{0 \le t < t^*} R^{t^*-t} S b_{a(t)}$, 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

$$F(u_{t*}) = F(R^{t^*} u_0) + \sum_{0 \le t < t^*} F(R^{t^*-t} S b_{a(t)}). \tag{33}$$

Instead of t^* the final time in the MIP is $|E|$. Hence the summation over all times between 0 and t^* here translates into a summation over E^* there. If we select as before the minimization of the average end temperature as the ultimate goal, that is:

$$\min \frac{1}{n+1} \sum_{p \in E^*} R^{|E|-p} u_0 y_p + \frac{1}{n+1} \sum_{p \in E^*} \sum_{a \in E} R^{|E|-p} S b_a z_{a,p}, \tag{34}$$

then the MIP-formulation is

$$\begin{aligned} &\text{minimize} && (34), && (35a)\\ &\text{subject to} && (3b), (3c), (3d), (3e), (3f), \text{ and } (3g). && (35b)\end{aligned}$$

A slight modification of the latter model allows us to tackle the coolest Hamiltonian cycle problem which was alluded to above. This modification is achieved by identifying start node and destination node, $v = w$, a modification that means no other change neither to the model nor to the solution algorithm.

The integrality gap of the second model is typically much smaller than that of the first model. One special structure that yields the gap is the contained set-partitioning polytope that is described by equations (3d). These, however, are much better behaved than the big-M-constraints of the first model. The main problem with the second model is the large number variables in the model. We note, however, that an additional preprocessing phase to remove redundant variables can reduce the load on the solver. For larger problems also a column generation approach could be used.

4. **Computational results.**

We start with an illustrative small example where we discuss all paths from source to destination. Then we turn to medium-size instances to demonstrate the solution time behavior on a larger set of random graphs of varying size. Finally we solve a large instance of one particular grid graph to show the current state-of-the-art of MIP solvers concerning our model. At URL http://opus.kobv.de/zib/volltexte/2009/1216/ our test instances and solver log files are online available.

4.1. **A small example network.**

The network in Figure 2 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 3 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 $1 \to 3 \to 5$.

4.2. **Medium scale networks.**

As basis for our computations we look at the case that we want to minimize the average temperature at the sink of the network. To compare the different modeling approaches we used a testbed of randomly generated graphs. The number of vertices and edges can be found in Table 1. We generated 10 instances for each size of the graph. In all the cases we used 31 discretization points in space and set $h := 10^{-2}$ and $k := 10$ for the discretization of the heat equation. To compute the matrix exponential we used the library Expokit [36]. The temperatures along the arcs were chosen randomly between 20 and 200. The constant starting temperature of the object $\Omega := [0, 1]$ was set to 110. The computations were carried out on a 1.0Ghz Intel-Pentium III with two cores and 2GByte RAM, and the single core version of the MILP solver CPLEX11.

With the first model (32) we cannot solve instances with 10 vertices, because the lower bounds that are produced by the relaxations are extremely bad. With the second model (35), we can treat examples of up to 70 vertices in reasonable time. The combinatorial problems then start to take over. From Figure 4 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.

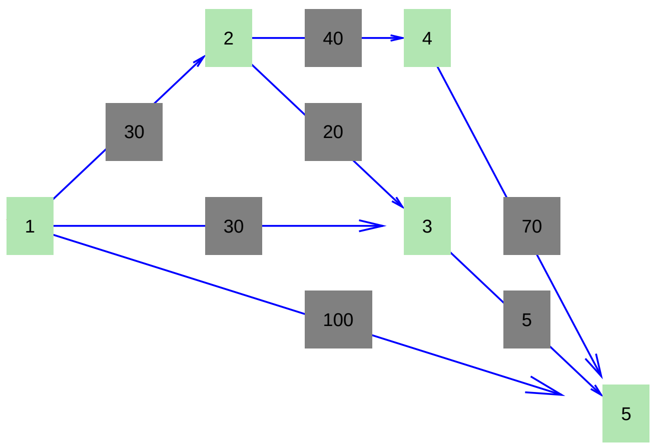
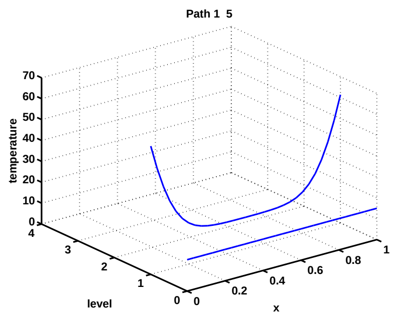
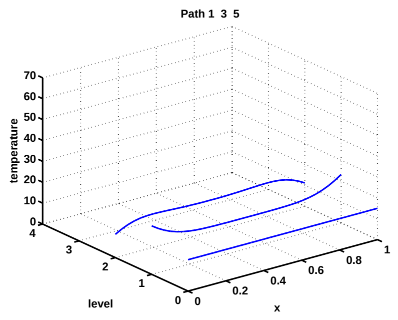


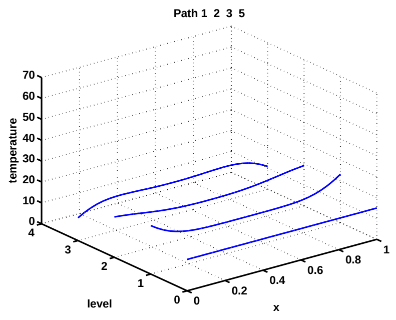
FIGURE 2. Example network.



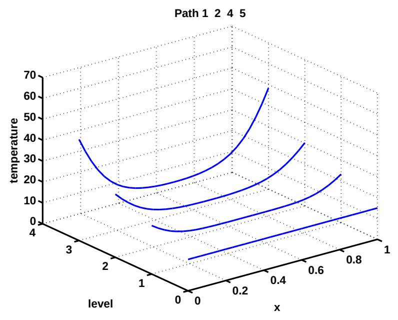
(a) Path 1 $\longrightarrow$ 5.



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



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



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

FIGURE 3. Temperature distributions in the nodes of the four paths of the example network, form the first node (level 0) to the final node.

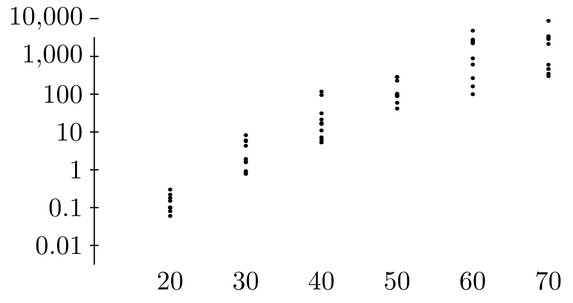


FIGURE 4. Running time for random samples

TABLE 1. Parameters of the random graphs

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

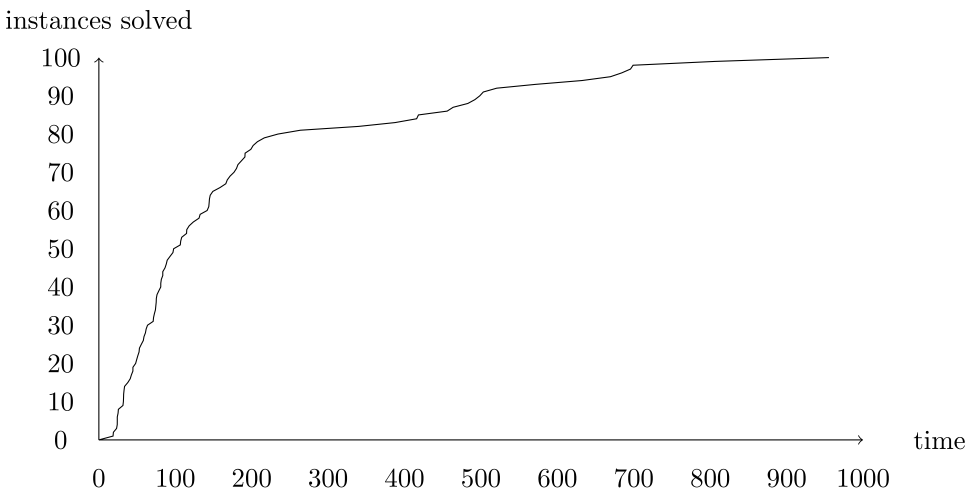


FIGURE 5. Comparison of running times for networks with 50 vertices and 500 edges (time given in seconds)

In Figure 5 we show a detailed comparison 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 example shown in Figure 6 needed about four weeks of computation on a 2.4Ghz AMD Opteron computer with 32GByte RAM and 8 available CPU cores for the parallel version of the MILP solver CPLEX11. The branch-and-bound tree consists 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 time was 2.2 mill. seconds (25 days). The solid lines in Figure 6 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.

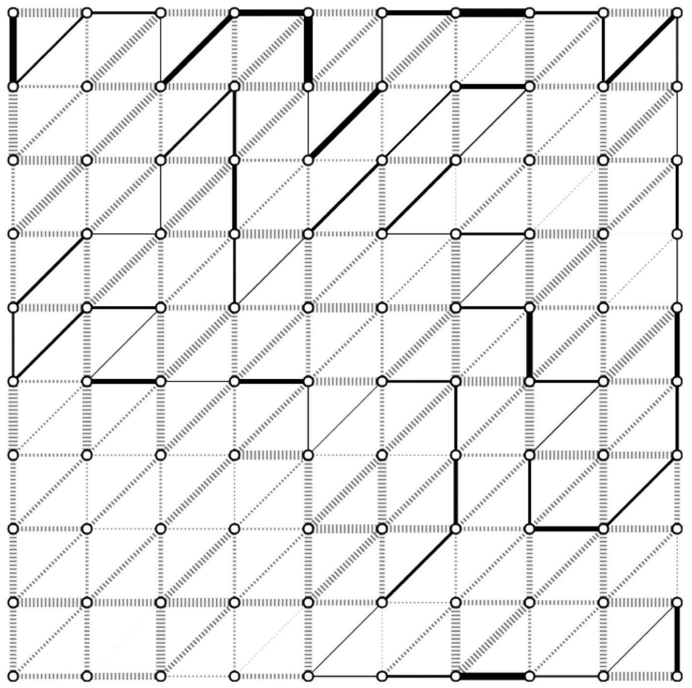


FIGURE 6. A 10×10 network with random temperatures, and a coolest path.

Informally speaking, one can see the following overall "behavior" 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 temperatures 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 6). Moreover one can observe that the coolest path might well include some "hot" arcs, as long as 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 7 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.

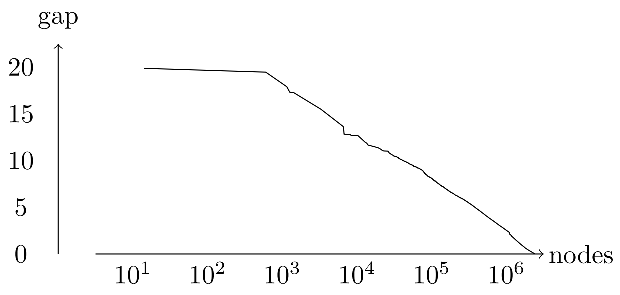


FIGURE 7. Computation of the 10×10 network: remaining gap (in percent) after n nodes

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.

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 Lagrangian 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 $(|V|^2 + |V| \cdot |E|)$. The master problem of the relaxation can then be solved using a bundle or a subgradient method. Although the linear relaxations are expected to be solved faster, the combinatorial structure of the problem will always be the bottleneck for larger instances.

Acknowledgments. 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] K. Aardal and A. K. Lenstra, *Hard equality constrained knapsack problems*, Mathematics of Operations Research, **29** (2004), 724–738.

[2] T. Achterberg, "SCIP - A Framework To Integrate Constraint And Mixed Integer Programming," Technical Report ZR-04-19, ZIB, 2004.

[3] F. Ahourai, M. Tabandeh, M. Jahed and S. Moradi, *A thermal-aware shortest hop routing algorithm for in vivo biomedical sensor networks*, IEEE 2009 Sixth International Conference on Information Technology: New Generations, Las Vegas, Nevada, April 27–April 29 2009, 1612–1613.

[4] E. Balas, S. Ceria and G. Cornuéjols, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, **58** (1993), 295–324.

[5] E. Balas, S. Ceria and G. Cornuéjols, *Mixed 0-1 programming by lift-and-project in a branch-and-cut framework*, Management Science, **42** (1996), 1229–1246.

[6] E. Balas, S. Ceria, G. Cornuéjols and N. Natraj, *Gomory cuts revisited*, Operations Research Letters, **19** (1996), 1–9.

[7] R. E. Bellman, *On a routing problem*, Quarterly of Applied Mathematics, **16** (1958), 87–90.

[8] R. Bixby, *Solving real-world linear programs: A decade and more of progress*, Operations Research, **50** (2002), 3–15.

[9] R. Borndörfer, “Aspects of Set Packing, Partitioning, and Covering,” Shaker Verlag, Aachen, 1998.

[10] S. Ceria, C. Cordier, H. Marchand and L. A. Wolsey, *Cutting planes for integer programs with general integer variables*, Mathematical Programming, **81** (1998), 201–214.

[11] G. B. Dantzig, “Linear Programming And Extensions,” Princeton University Press, 1963.

[12] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik, **1** (1959), 269–271.

[13] L. C. Evans, “Partial Differential Equations,” AMS, 1999.

[14] L. R. Ford, “Network Flow Theory,” Technical Report P-923, The Rand Corporation, Santa Monica, 1956.

[15] J. Forrest and R. Lougee-Heimer, “CBC User Guide,” Online available at URL `http://www.coin-or.org/Cbc`, 2005.

[16] M. R. Garey and D. S. Johnson, “Computers And Intractability: A Guide To The Theory Of NP-Completeness,” W. H. Freeman, 1979.

[17] R. E. Gomory, *An algorithm for the mixed integer problem*, Technical Report RM- 2597, The RAND Cooperation, 1960.

[18] M. Guilbert, P.-B. Wieber and L. Joly, *Optimization of complex robot applications under thermal constraints*, INRIA Technical Report 6074, 2006.

[19] N. Karmarkar, *A new polynomial time algorithm for linear programming*, Combinatorica, **4** (1984), 373–395.

[20] A. H. Land and A. G. Doig, *An automatic method of solving discrete programming problems*, Econometrica, **28** (1960), 497–520.

[21] A. Land and S. Powell, *Computer codes for problems of integer programming*, Annals of Discrete Mathematics, **5** (1979), 221–269.

[22] J. T. Linderoth and M. W. P. Savelsbergh, *A computational study of search strategies for mixed integer programming*, INFORMS Journal on Computing, **11** (1999), 173–187.

[23] H. Marchand, “A Polyhedral Study Of The Mixed Knapsack Set And Its Use To Solve Mixed Integer Programs,” PhD thesis, Université Catholique de Louvain, Louvain- la-Neuve, Belgium, 1998.

[24] H. Marchand and L. A. Wolsey, *Aggregation and mixed integer rounding to solve MIPs*, Operations Research, **49** (2001), 363–371.

[25] F. Margot, *BAC: A BCP based branch-and-cut example*, IBM Research Report RC22799 (W0305-064), 2003, revised 2008.

[26] E. F. Moore, *The shortest path through a maze*, Procedings of the International Symposium on the Theory of Switching, Havard University Press, 285–292, 1959.

[27] G. L. Nemhauser and L. A. Wolsey, *A recursive procedure to generate all cuts for 0-1 mixed integer programs*, Mathematical Programming, **46** (1990), 379–390.

[28] G. L. Nemhauser and L. A. Wolsey, “Integer And Combinatorial Optimization,” Wiley-Interscience, 1999.

[29] M. W. Padberg, *Classical cuts for mixed-integer programming and branch-and-cut*, Mathematical Methods of OR, **53** (2001), 173–203.

[30] M. W. Padberg and G. Rinaldi, *Optimization of a 532-city symmetric travelling salesman problem by branch and cut*, Operations Research Letters, **6** (1987), 1–7.

[31] M. W. Padberg, T. J. Van Roy and L. A. Wolsey, *Valid inequalities for fixed charge problems*, Operations Research, **33** (1985), 842–861.

[32] A. Pazy, “Semigroups Of Linear Operators And Applications To Partial Differential Equations,” Springer, 1983.

[33] M. Renardy and R. C. Rogers, “An Introduction To Partial Differential Equations,” Springer, 1996.

[34] A. Schrijver, “Theory Of Linear And Integer Programming,” Wiley, 1998.

[35] A. Schrijver, *On the history of combinatorial optimization (till 1960)*, in “Handbook of Discrete Optimization” (eds. K. Aardal, G. L. Nemhauser, R. Weismantel), Elsevier, Amsterdam, 1–68, 2005.

[36] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Transactions on Mathematical Software, **24** (1998), 130–156.

[37] R. Stephan, *Facets of the (s,t)–p-path polytope*, Online available at URL `arXiv:math/0606308v1`.

[38] R. Stoer and D. Burlisch, "Numerische Mathematik," Springer, 2000.

Received March 2009; revised December 2009.

E-mail address: frank@mathcces.rwth-aachen.de
E-mail address: fuegenschuh@zib.de
E-mail address: herty@mathc.rwth-aachen.de
E-mail address: schewe@mathematik.tu-darmstadt.de