Real-time ATO reconfiguration for operational stability

A. Iliasov, I. Lopatkin, A. Mihut & A. Romanovsky
Newcastle University, UK

Abstract

This paper briefly describes a technique for improving capacity and operational stability of busy mainline junctions and stations. The technique, called the SafeCap advisory system, employs a mixture of existing train monitoring and control technologies to introduce an advisory control layer on top of fixed-block signalling. We present the main design principles and illustrate the advantages of the proposed solution with some experiments conducted in a train simulation toolkit.

1 Introduction

Static optimisation of junctions and stations critically depends on knowledge about train schedules and traffic mix. There is a danger of over-optimising to such a degree that stability is heavily compromised. To counter this, static optimisation should be somewhat defensive and leave some slack in train spacing. Real-time control can make the best use of this slack by assessing, in real time, the current state of trains around a station, acting to reconfigure signalling and, if applicable, ATO (Automated Train Operation) programmes.

Automated Train Operation-capable trains and moving-block signalling present the best opportunities for real-time control. The main function of the latter is an accurate projection of the current train configuration and control commands into the future. By projecting a set of alternative control scenarios, one can explore future configurations and choose the locally optimal control. Control software could alter acceleration/ braking curves, set individual train speed limits, adjust dwelling times, and choose platforms. It should allow a system to be quickly stabilised should there be a need to accommodate an extra train, recover from

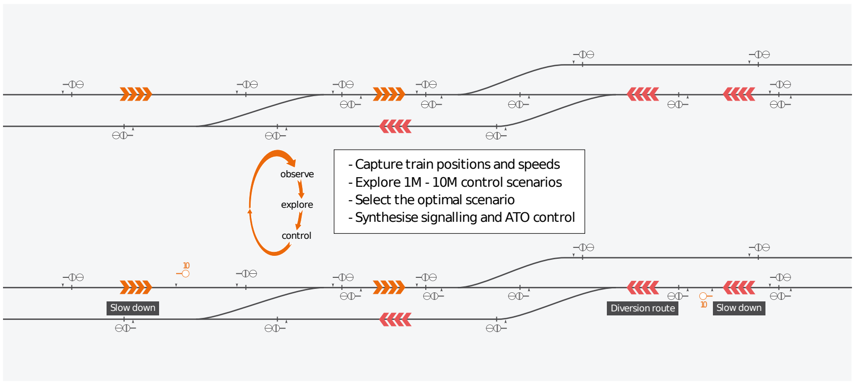

observe
explore
control
- Capture train positions and speeds
- Explore 1M - 10M control scenarios
- Select the optimal scenario
- Synthesise signalling and ATO control
Slow down
Diversion route
Slow down

Figure 1: A CUDA-based [1] simulation can explore many millions of control scenarios in just a few seconds, to predict and pre-empt delays and disruptions.

train delays, and, generally, sustain a high rate of train flow when there are limited opportunities for implementing stability margins in a schedule.

Albeit a simple idea, technologically realtime control is extremely challenging. To provide meaningful control logic for each particular station, and to be able to predict the time of arrival and the speed of approaching trains, one needs to consider a large context area. Furthermore, finding an optimum in such a setting is bound to rely on probabilistic and genetic algorithms that require a fairly large test base, in the order of tens of millions of control scenarios. Such scenarios would have to be generated and processed within a tiny time budget: one to ten seconds. Fortunately, the problem is amenable to massive parallelisation, and there are computer science techniques and specialised hardware to achieve this today.

Fig. 1 illustrates a schematic representation of a railway model, where each train's behaviour is pre-computed depending on the state of the route interlocking. Essentially, before commencing movement each train computes a number of scenarios in its near horizon, and selects to act upon the one which renders an optimal track capacity. Since at every time slice such a computation is carried by every train present in the simulation, the horizon might be increasingly more difficult to compute, being influenced directly proportional by the number of trains present. Hence, a parallel implementation is a good candidate for an optimisation which has potential for drastically increasing overall efficiency.

2 Overview

Our aim is to improve capacity and operational stability of a railway network. By capacity we understand the ability of a network to accommodate specific traffic schedules. The stability property ensures that railway operation is able to recover

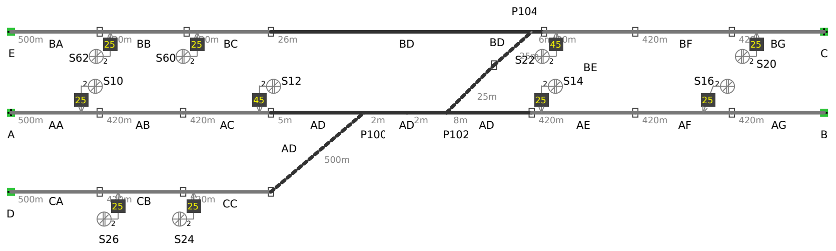


Figure 2: Sample layout.

(i.e., run to schedule) promptly after a disruption caused by a late or unscheduled train.

One common technique for assessing capacity is the UIC 406 “Capacity Calculation Method” [2] that measures how well a given layout and signalling are able to accommodate some specific schedule and whether there is enough capacity left to introduce extra traffic. Stability is achieved through a degree of redundancy in the spatial domain: time gaps between successive trains are used to mitigate knock-on effect of delayed trains and isolated ’service’ gaps help to schedule unplanned traffic.

The measures advocated by UIC 406 to increase stability achieve this at the expense of capacity: time gaps between trains mean fewer trains per unit of time. Time gaps may be smaller if the trains are slower (on average) but this again severely impacts capacity. Thus, for the critical sections of railway infrastructure, traditional schedule-based approach to capacity and stability is too restricting.

At a finer level of granularity, the available capacity of a network is distributed quite unevenly. A straight sections of track offer offers more capacity than, for instance, a curved one (assuming a speed limit is enforced on a curved section). Not much can be done to rectify this other constructing new tracks or using tilting trains. However, the major sources of capacity restriction are areas where tracks intersect and trains stop, i.e., junctions and stations. And, unlike curved tracks, we believe there is an opportunity for capacity improvement in junctions and stations by adding a *train advisory* layer on top of the existing signalling and interlocking. The guiding idea is to redistribute capacity consumption from areas of restricted capacity (junctions, stations) to areas where there may be an excess capacity (e.g., straight line). Our thesis is that this leads the better overall capacity utilisation.

Another important step is realising that assessing in static context, with a layout and schedule alone, necessarily leads to conservative solutions that try to minimize occurrence of significant service degradation at the cost of somewhat lower overall performance. In contrast, a reactive control system, operating in real-time, has far more opportunities for delivering higher capacity as extreme events may be ruled out with a high degree of confidence for the time window of control.

As an illustration, consider the layout in Fig. 2. Assume that running trains over a path entering the layout at A and leaving at B delivers capacity $cap(A\_B)$ when there is no traffic on other paths; let us denote the same metric for paths from C to D and from C to E as $cap(C\_D)$ and $cap(C\_E)$. Also let $cap(p_1, \ldots, p_n)$ be the overall capacity of simultaneous traffic on paths $p_1, \ldots, p_n$. For our example we have that

$$cap(A\_B) + cap(C\_E) = cap(A\_B, C\_E)$$
$$cap(A\_B) + cap(A\_B) \leq cap(A\_B, C\_D)$$

In other words, when some paths share an area of track the resulting interference may prevent attaining the full capacity of each path considered in isolation. Whereas paths do not interfere, like A_B and C_E in our example, their combined capacity is a simple summation of individual capacities. If we account for the interference between some two path p_i and p_j with the term $int(p_i, p_j) \geq 0$ then the overall capacity is sum of isolated path capacities minus the overall path interference:

$$cap(p_1, \ldots, p_n) = \sum_i cap(p_i) - \sum_{i \neq j} int(p_i, p_j) \tag{1}$$

Hence, to maximize capacity one needs to minimize term $\sum_{i \neq j} int(p_i, p_j)$.

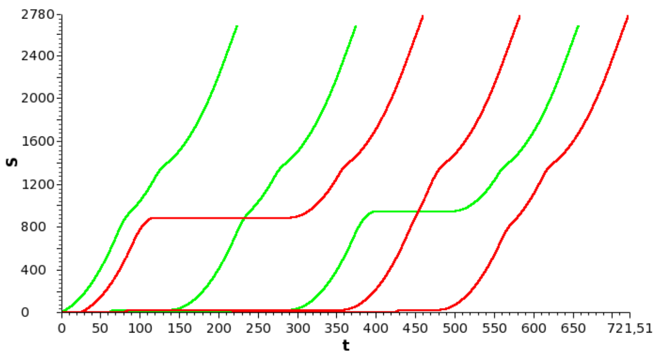
The inference of paths p_i and p_j is a consequence of interference of individual trains on these paths, i.e., the delays introduced on either path due traffic on another path. The ordering of trains on a path may be fixed by a route reservation policy, commonly defined in a train schedule. Our approach, however, requires a degree of flexibility in route reservation where the priority of reservation of routes with shared track elements is dynamically computed by the proposed advisory system.

3 Control and optimisation

Interference $int(p_i, p_j)$ may be determined from the observation of train runs (using, for instance. a simulator tool) and comparing the measured capacity $cap(p_i, p_j)$ to $cap(p_i) + cap(p_j)$. Each of these terms is completely defined by a record of individual train runs - record of train position over time. A detailed train run is a costly to compute and manipulate for a railway junction of any significant size. This limitation becomes prohibitive in the context of real-time control. In lieu of a suitable explicit form of term $int(p_i, p_j)$ that gives an accurate quantitative measure of interference, it is necessary to devise a simpler, proxy metric that only correlates with actual interference value to provide a qualitative measure of interference.

As such metric we propose to employ a record of times when circuits containing points are locked and freed by trains. Such a record T, defines, for each point, a sequence which elements are triples of path, circuit entry time and circuit occupation time:

$$T = \begin{cases} \mathrm{R}_j & : \quad \ldots, (p^j_i, \tau^j_i, \Delta^j_i), (p^j_{i+1}, \tau^j_{i+1}, \Delta^j_{i+1}), \ldots, (p^j_{i_k}, \tau^j_{i_k}, \Delta^j_{i_k}), \ldots \\ \ldots & : \quad \ldots \end{cases}$$



$$AD \; : \; (AB, 126.9, 8.9), (AB, 278.9, 9.1), (CD, 353.6, 11.6), (CD, 476.7, 11.7), \ldots$$
$$BD \; : \; (CD, 351.7, 5.6), (CD, 474.35, 4.94), (CD, 403.6, 5.1)$$

Figure 3: Train run plot and shared circuit traversal histories. In green (light gray) are trains on paths between nodes A, B; in red (dark gray) are trains between nodes C, D. All trains are the same.

In the above, R_j is the name of a train detection circuit (e.g, AD in Fig. 2) which contains point(s) or diamond crossing(s); p_i^j are path identifiers; τ_i^j, Δ_i^j are positive real values that define the moment and extent of time when circuit R_j was occupied by a train from the respective path.

The example in Fig. 3 shows simulated run over the layout from Fig. 2 obtained with the SafeCap modelling platform [3]. Six trains, divided evenly between paths AB and CD are scheduled to enter the layout in 25 second intervals and trains on AB have priority in route reservation. The time vs. distance plot in 3 shows the first train on CD being delaying on signal $S22$. Other CD trains are also slightly delayed by traffic on AB. Slightly wobble of the AB distance plot is the effect of sub-optimal two aspect signalling. Fig. 3 also gives an excerpt of shared circuit traversal histories. One point of interest is that the second train on line CD occupies AD for a somewhat shorter duration of time, i.e., it enters circuit boundary with a higher speed.

By changing train schedule and introducing extra delays (i.e., an extra stop or longer dwelling time), one may observe a differing record T'. It is useful to identify the ways some two records T and T' may differ. One possibility is that a train arrives at the boundary of a circuit slightly later or earlier and occupies it for a different duration of time. We record such a change with a transformation rule shift:

$$\mathrm{shift}(T, j, i, t_0, t_1) = \left\{ \begin{array}{lcl} \mathrm{R}_j & : & \ldots, (p_i^j, \tau_i^j + t_0, \Delta_i^j + t_1), \ldots \\ \ldots & : & \ldots \end{array} \right.$$

One may also observe, due to a differing schedule or a change in route reservation policy, a different order of paths in the history record of a circuit. Rule swap defines a transformation where two previously adjacent paths $p^j_{i_s}$ and $p^j_{i_{s+1}}$ are swapped:

$$\text{swap}(T, j, i) = \begin{cases} \text{R}_j & : \quad \ldots, (p^j_{i+1}, \tau^j_{i+1}, \Delta^j_{i+1}), (p^j_{i_i}, \tau^j_{i_i}, \Delta^j_{i_i}), \ldots \\ \ldots & : \quad \ldots \end{cases}$$

If the traffic pattern remains invariant (i.e., the same number trains travel over same paths), any possible record T' may be obtained from any other record T by applying a sequence S of transformation rules shift and swap: $T' = S(T)$. As a notation short-cut, we treat S as a composite rule formed by a successive application of individual rules it is comprised from.

The notion of transformation rules may be used to define a measure of how close two records T and T' are. Let S^* be the set of all rule sequences such that for any $s \in S^*$ it holds that $T' = s(T)$. Then the *distance* $\|T - T'\|$ between records T and T' is defined as follows:

$$\|T - T'\| = \sum_i \|s_i\| \qquad \text{where } s = \arg\min_s \text{card}(s),$$

s is the shortest rule sequence, s_i is the i-th transformation rule in s and $\|s_i\|$ is a transformation rule distance metric, defined by the following (empirical) formulae:

$$\|\text{shift}(T, j, i, t_0, t_1)\| = |t_1| + \sqrt{|t_0|} \qquad \|\text{swap}(T, j, i)\| = |\tau^j_{i+1} - \tau^j_{i_i}|.$$

Here $|x|$ signifies the absolute value of x.

The essence of our technique is based on the following conjecture: *small (large) variations in record history distance* $\|T - T'\|$ *correspond to small (large) variations in interference value* $int(p_i, \ldots, p_k)$. In other words, metric $\|T - T'\|$ provides a qualitative characterisation of capacity change.

3.1 Real-time control

The SafeCap [4] advisory system implements a real-time control loop for the supervision of train movement at the sub-signalling level with the aim of increasing overall system capacity or meeting specific operation goals. It is assumed that the system has a complete view of a controlled area including position and velocity of each train and the state of movable track equipment. This view cumulatively defines the sensors of the control system. Its sole actuator - the way it influences train behaviour - is the command interface to an automated train operation subsystem (ATO), assumed to be already in place. The control system issues commands to ATO to slow down or speed up specific trains. The core of the control is an optimisation procedure based on the notion of shared circuit traversal history, presented above. An optimal control is identified by finding a sequence of transformation rules s leading to a significant and positive change in capacity.

The diagram in Fig. 4 presents the overall architecture of the developed control system. The first step, *state snapshot*, polls railway infrastructure and sensing

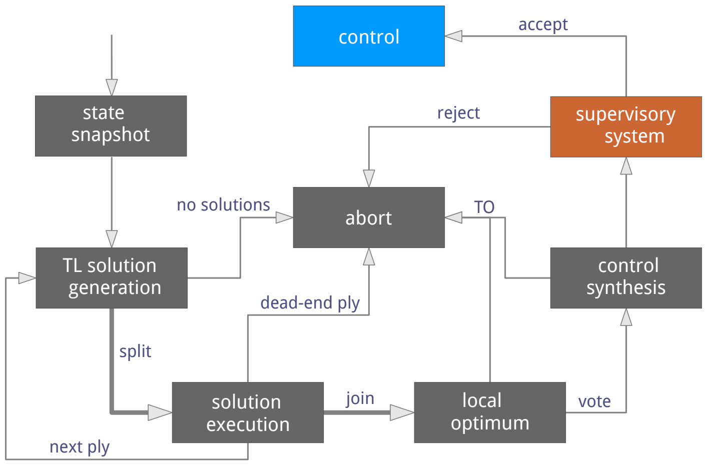


Figure 4: SafeCap real-time control system.

equipment to acquire train and equipment state. A simulator tool is instantiated with the captured state and its simulation run provides the initial record history T.

The *solution generation* step employs a probabilistic procedure to generate a set of transformation sequences $S^{(1)}$ such that, at this stage, each sequence has just one transformation rule. The procedure is guided by the $\|T - s(T)\|, s \in S^{(1)}$ metric to filter away all transformations s where $\|T - s(T)\|$ is either too small (i.e., likely to give insignificant change in capacity) or too large (unlikely to correspond to any actual train runs). Early filtering is essential as the number of potential solutions is very large.

In general, we are looking for solutions that are likely to yield significantly different performance. Many of these solutions will offer a lower capacity than the original situation; some proportion may also be not *realizable*. Realizability of transformation rules $\{s_1, \dots, s_k\}$ requires that every train, starting from the captured state, is able, in a simulator tool, to meet the constraints of every $T' = s_i(T), i \in 1..k$. To remove solutions which unrealisable or offer lower performance, we employ an event-based train simulator to quickly confirm the capacity utilisation and the feasibility of the imposed constraints. The capacity utilisation metric may be a general purpose one (i.e., track utilisation) or specific to an area and situation (e.g., station call punctuality).

The result is a set $S_f^{(1)} \subseteq S^{(1)}$ of *viable* one-step transformation sequences. We call such set the first-ply solution and, more generally, refer to $S_f^{(n)}$ as n-th ply solution. The ply index describes how far, in the terms of shared circuits, the optimisation procedure has advanced. The first ply solution considers only the initial points and crossings on each paths, the second ply adds all the second ones, and so on. Each ply adds a new dimension to the solution space: each

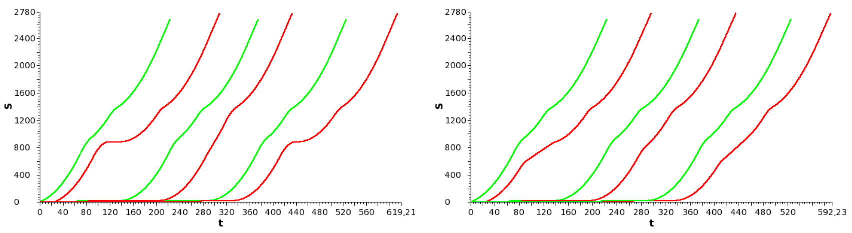


Figure 5: Train runs of optimal solution; the left plot is an optimal train ordering by the swap heuristic; the right also plot adds the speed control of shift heuristic.

transformation sequence s from ply i is expanded to a set of sequences q such that s is a prefix for every sequence in q.

The set of first-ply solutions is fed back into the solution generation procedure to obtain the candidate set $S^{(2)}$ of second-ply solutions. –These are again filtered by realizability and capacity estimates and provide the basis for the next ply. The depth of the solution generation procedure is limited by the number of circuits with movable elements that occur on a path. The largest such number (i.e., a path with three points, all on separate circuits) defines the index of the last ply.

Once all solutions are generated and filtered, those remaining are ranked to identify one optimal solution. The ranking criteria is specific to the current operational goals and the area controlled.

The control loop executed every 5 to 10 seconds. This time window is small enough to maintain a useful degree of feedback in the control loop and large enough to be able to consider a large number of potential solutions.

We apply certain heuristics to generate a set of candidate solutions (transformation sequences). This is set $S^{(i)}$ before filtering with $\|T - s(T)\|$ metric. For each transformation rule there is a separate heuristics. We first apply the swap heuristic to obtain a reordering of shared circuit traversal and afterwards use the shift heuristic to attempt to increase performance of each specific reordering.

3.2 The swap heuristic

The aim of the swap heuristic is to generate a relatively small number of promising candidate solutions by changing order trains over shared circuits. The key idea is that the that the likelihood of swapping some two paths p^j_{i+1} and $p^j_{i_i}$ decreases with the time distance between $|\tau^j_{i+1} - \tau^j_{i_i}|$ between these instances of traversal. Thus, if some two trains travel over a point one right after another, the heuristic is likely to generate a candidate solution where the order of the trains are swapped for this point. If the trains are separated by a considerable time gap, the swap becomes less likely.

The following recursive function describes the major steps of swap heuristics. It is variant of a simulated annealing algorithm.

```
Data: history $T$, per-path energy $E$ and termination tolerance $\epsilon$
Result: set $S$ of transformation sequences
begin
    $S \leftarrow \emptyset$
    while $\sum_i E_i > \epsilon$ do
        $t \leftarrow |\tau^j_{i+1} - \tau^j_{i_i}|$
        $r \leftarrow \mathbf{random}(0, 1)$
        if $e^{-t/(E_i+E_j)} > r)$ then
            $E_i \leftarrow E_i - t, E_j \leftarrow E_j - t$
            $S \leftarrow S \cup \text{genswap}(\text{swap}(T, i, j), E, \epsilon)$
        end
    end
    return $S$
end
```

Function genswap(T, E, ϵ)

Fig. 5 demonstrates an application of the heuristic to the example from Fig. 2. An optimal solution here is the alternating order of trains. This heuristic determined between 6 or 7 (depending on random seed) candidate solutions out of 720 possible combinations. The best of the solutions (Fig. 5, left) delivers 21% increase in track utilisation having considered. The worst of these solutions delivers 17% capacity decrease.

3.3 The shift heuristic

The aim of this heuristics is to adjust the way a train approaches the boundary of a route containing a shared train circuit. With conventional signalling, a train slowing down to a red signal has no knowledge of the cause of the signal being red nor how soon the cause will cease to be and let the signal show a permissive aspect.

We are going to override this behaviour and guide the train to approach a red signal at speed. Knowing the position and capabilities of each train, it is possible to predict exactly the point of time when the route containing a shared circuits becomes available to any given train. Thus, we can command a train to arrive at the route boundary right at the moment the route is available for the train.

No matter how long a train had to wait for a point and its containing route to be allocated to the train, the moment this happens the train should be entering the route with the maximum feasible speed. Given this rule is maintained for all the trains, the occupation of the critical sections of railways (those containing points and switching crossing) is minimized (within the limitation of a current solution).

Although such behaviour does not violate route-based signalling principles (train should never pass a red signal), the risk of passing a red signal increases as a train approaches it at speed. Two principal safety violation scenarios are having train on a circuit of route and the fault in a movable track equipment. The former case may be largely excluded with a small (e.g., 5 sec.) control window: the advisory system would be able to detect stationary or unusually slow train and direct other trains accordingly. The latter case when, for instance, a point lock is not acquired, can be managed by ascertaining that at all times there is a sufficient braking distance for ATP to stop a train in front of a faulty point.

The righ-hand side plot in Fig. 5 shows a combination of the optimal solution found by *swap* and approach speed control computed by the *shift* heuristic. As evident from the plots, the heuristic prevents two of the trains from stopping and thus making them traverse the shared area quicker. This deliver approx. 8% increase in capacity.

3.4 Synthesising ATO control

The selected optimal solution (in the form of transformation sequence s_j) must be transformed into a set of train control rules.

The first step consists in overriding of the existing route reservation and point locking schedule in order to realise swap rules of the optimal solution s_j. Since from T' we order the relative order of traversal of every shared circuit (and thus point and diamond crossing), it is not straightforward to compute such new route reservation schedule.

The second kind of rules, the shift rule, speeds up or slows down a train. The effect of these rules is synthesised using a dedicated simulation procedure where train time of traversal of shared circuits is reflected in *waypoints* on a train path.

A waypoint $W = (d, t)$ is defined by its offset d within the train path and the arrival time t. If every train respects the constraints set by waypoints, one would observer history record $T' = s_j(T)$ (record T is a project of state snapshot based on signalling and scheduling rules alone).

The set of waypoints are handed over to a third-party ATO subsystem. This subsystem is assumed to analyse waypoints for feasibility and safety and generate corresponding driving profiles. It the obligation of the ATO subsystem to reject waypoints that may violate signalling constraints. If any single waypoint is rejected, all waypoints are cleared and the system falls back, until the next control window, into it original state.

4 Conclusion

We have briefly presented out findings on real-time advisory control system. The system attempts to stabilise performance in presence of disturbances by issuing ATO control commands computed by a relatively simple optimisation engine. We are working on applying the technique to a large-scale, real world railway operation spanning busy stations.

The system illustrated in the present paper is aimed at computing a myriad of scenarios for each entity in order to determine the optimal solution in terms of track capacity at any given time-step and route region. By taking advantage of the multi-core graphics processor, the simulation could compute scenarios into the future for each train entity in parallel, simultaneously, and return the optimal scenario, therefore increasing the computation speed that the CPU would have to carry. In addition to that, a very large number of train entities could be simulated in parallel using the appropriate hardware and memory management paradigms.

References

[1] Nickolls, J., Buck, I., Garland, M. & Skadron, K., Scalable parallel programming with CUDA. *Queue*, **6(2)**, 2008.
[2] UIC, Kodex 406, 2004.
[3] Iliasov, A., Lopatkin, I. & Romanovsky, A., The safecap platform for modelling railway safety and capacity. *SAFECOMP*, 2013.
[4] SafeCap Project, SafeCap Platform website, 2012. Available at http://sf.net/projects/safecap.