

# Canadians should use memory

Pierre Bergé<sup>\*,1</sup>, Julien Hemery<sup>2</sup>, Arpad Rimmel<sup>2</sup>, and Joanna Tomasik<sup>2</sup>

<sup>1</sup>LRI, Université Paris-Sud, Université Paris-Saclay, 91405 Orsay Cedex, France

<sup>2</sup>LRI, CentraleSupélec, Université Paris-Saclay, 91405 Orsay Cedex, France

## Abstract

We establish that the competitive ratio of any randomized memoryless strategy is not less than  $2k + 1$ . Only randomized strategies using memory potentially overpass this ratio now.

## 1 Definitions

We start by introducing the notation. For any graph  $G = (V, E, \omega)$ , let  $G \setminus E'$  denotes its subgraph  $(V, E \setminus E', \omega)$ . If  $P$  is a path, we note its cost as  $\omega(P) = \sum_{e \in P} \omega(e)$ .

### 1.1 Memoryless Strategies for the $k$ -CTP

We remind the definition of CTP. Let  $G = (V, E, \omega)$  be an undirected graph with positive weights. The objective is to make a traveller traverse the graph from a source node  $s$  to a target one  $t$ , with  $s, t \in V$ . There is a set  $E_* \subsetneq E$  of blocked edges. The traveller does not know a priori which edges are blocked. He discovers a blocked edge only when arriving to one of its endpoints. For example, if  $(v, w)$  is a blockage he will discover it when arriving to  $v$  (or  $w$ ). The goal is to design the strategy  $A$  with the minimal competitive ratio.

We focus on *memoryless strategies* (MS). Concretely, we suppose that the traveller remembers the blocked edges he has discovered but forgets the nodes which he has already visited. In other words, a decision of an MS is independent of the nodes already visited. In the literature, the term *memoryless* was used in the context of online algorithms (e.g. PAGING PROBLEM [?], LIST UPDATE PROBLEM [?]) which take decisions according to the current state, ignoring past events. An MS can be either deterministic or randomized.

**Definition 1 (Memoryless Strategies for the  $k$ -CTP)** *A deterministic strategy  $A$  is an MS if and only if (iff) the next node  $w$  the traveller visits depends on graph  $G$  deprived of blocked edges already discovered  $E'_*$  and the current traveller position  $v$ :  $w = A(G \setminus E'_*, v)$ . Similarly, a randomized strategy  $A$  is an MS iff node  $w$  is the realization of a discrete random variable  $X = A(G \setminus E'_*, v)$ .*

MSes are easy to be implemented because they do not use past moves to take a decision. For the same reason, they do not need any extra memory either.

For example, the GREEDY strategy [?] is an MS. It consists in choosing at each step the first edge of the shortest path between the current node  $v$  and the target  $t$ . This strategy, illustrated

---

\*Corresponding author: [Pierre.Berge@lri.fr](mailto:Pierre.Berge@lri.fr)

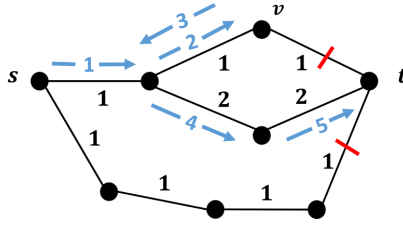


Figure 1: Illustration of the GREEDY strategy on a graph with 2 blocked edges

in Figure 1, does not refer to anterior moves. In contrast, the REPOSITION strategy [?] is not an MS as any decision refers to the past moves of the traveller.

We propose the following process to identify whether a strategy  $A$  is a deterministic MS. Let us suppose that a traveller  $T_1$  executes strategy  $A$ : he has already visited certain nodes of the graph, he is currently at node  $v$  but he has not reached target  $t$  yet. Let us imagine a second traveller  $T_2$  who is airdropped on node  $v$  and starts applying strategy  $A$ . If the traveller  $T_2$  always follows the same path as  $T_1$  until reaching  $t$ ,  $A$  is a deterministic MS. If  $T_1$  and  $T_2$  may follow different paths, then  $A$  is not an MS. Formally, proving that a strategy is an MS consists in finding the function which transforms the pair  $(G \setminus E_*, v)$  into node  $w = A(G \setminus E'_*, v)$ .

## 1.2 Competitive ratio

Let  $(G, E_*)$  be a *road map*, i.e. a pair with graph  $G = (V, E, \omega)$  and blocked edges  $E_* \subsetneq E$ , such that there is an  $(s, t)$ -path in graph  $G \setminus E_*$  (nodes  $s$  and  $t$  remain in the same connected component when all blocked edges are discovered). We note  $\omega_A(G, E_*)$  the distance traversed by the traveller reaching  $t$  with strategy  $A$  on graph  $G$  with blocked edges  $E_*$  and  $\omega_{\min}(G, E_*)$  the cost of the shortest  $(s, t)$ -path in graph  $G \setminus E_*$ .

The ratio  $\omega_A(G, E_*) / \omega_{\min}(G, E_*)$  is abbreviated as  $c_A(G, E_*)$ . A strategy  $A$  is  $c_A$ -competitive [?, ?] iff for any  $(G, E_*)$ ,  $\omega_A(G, E_*) \leq c_A \omega_{\min}(G, E_*)$ . Otherwise stated, for any  $(G, E_*)$ ,  $c_A(G, E_*) \leq c_A$ . If strategy  $A$  is randomized,  $\omega_A(G, E_*)$  is replaced by  $\mathbb{E}(\omega_A(G, E_*))$  which is the expected distance traversed by the traveller to reach  $t$  with strategy  $A$ . The competitive ratio can also be evaluated on a family  $\mathcal{R}$  of road maps, put formally,

$$c_{A, \mathcal{R}} = \max_{(G, E_*) \in \mathcal{R}} c_A(G, E_*) \quad (1)$$

This “local” competitive ratio fulfils  $c_{A, \mathcal{R}} \leq c_A$ . The competitive ratio can also be extended to families of strategies. We note  $c_{\text{MS}}$  the competitive ratio of MSes, which is the minimum over competitive ratios of any MSes:  $c_{\text{MS}} = \min_{A \text{ MS}} c_A$ . In the remainder, we identify a set of road maps  $\mathcal{R}$  such that the competitive ratio of MSes on it is  $2k + 1$ , i.e.  $c_{\text{MS}, \mathcal{R}} \geq 2k + 1$ .

## 1.3 Graphs $G_k$

We define recursively a sequence of graphs  $G_i$  for  $i \geq 1$  with weights from  $\{1, \varepsilon\}$ ,  $0 < \varepsilon \ll 1$ . Graphs  $G_1$  and  $G_{i+1}$  are represented in Figures 2a and 2b, respectively. Edges with weight 1 are thicker than edges with weight  $\varepsilon$ . For every graph  $G_k$ , we will only consider the road maps where the blocked edges are at the right of the graph, the left edges affecting negligibly the total cost of a traveler. We note  $E_{\text{right}}$  the set of all the right edges.

When the traveller discover blocked edges in  $G_k$ , parts of the graph become *dead ends*. If a traveller visits a dead end, the only chance for him to reach  $t$  is to leave it anyway. Dead ends

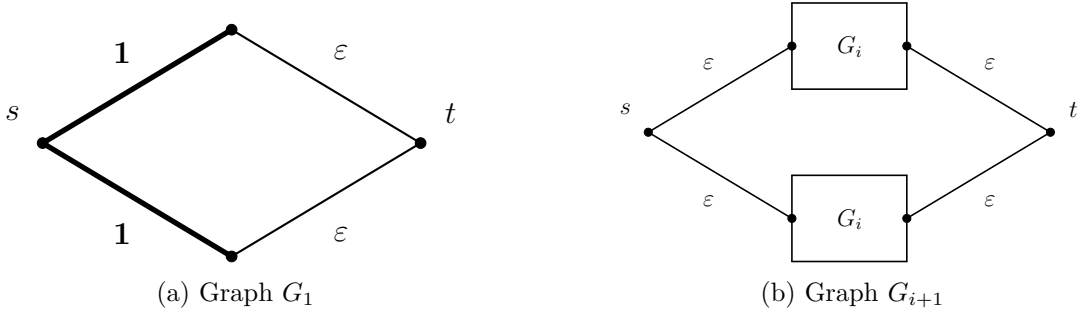


Figure 2: Recursive construction of graphs  $G_i$

only provide to the traveller additional costs which make the competitive ratio increase. As a consequence, the most competitive strategies will not visit dead ends. From now on, as soon as a dead end appear when the traveller discovers a blockage, we make it disappear from the graph: graph  $G \setminus E'$  becomes graph  $G$  deprived of both edges  $E'$  and dead ends appeared when edges  $E'$  are deleted. Let  $\mathcal{G}_k$  denote the set of all sub-graphs of  $G_k$  where at most  $k$  edges have been removed. Formally,  $\mathcal{G}_k = \{G_k \setminus E' : |E'| \leq k\}$ .

#### 1.4 Equivalent binary trees

We define for each graph of  $\mathcal{G}_k$  an equivalent binary tree, which will represent all changes of direction from  $t$ . We note  $T_\emptyset$  the empty tree, and  $(v, T_a, T_b)$  a non empty tree with  $v \in V$ ,  $T_a$  the binary tree above and  $T_b$  the binary tree below.

We obtain the equivalent binary tree by applying the BIN-TREE algorithm. This algorithm take as an entry a directed tree obtained by first removing the left part of the graph and then directed all edges from  $t$ . Then, the BIN-TREE algorithm will construct a binary tree according to the outdegree of each node.

---

**Algorithm 1:** BIN-TREE algorithm

---

**Data:** a directed tree  $T$  and a node  $v$

**Result:** binary tree

**if**  $\deg^-(v) = 0$  **then return**  $T_\emptyset$  ;

**if**  $\deg^-(v) = 1$  **then return** BIN-TREE( $T, v_{\text{next}}$ ) ;

**if**  $\deg^-(v) = 2$  **then return** ( $v$ , BIN-TREE( $T, v_{\text{up}}$ ), BIN-TREE( $T, v_{\text{down}}$ ) ;

---

Given an edge in a binary tree, we will note  $A(e)$  the child edge from above and  $B(e)$  the child edge from below. We also note  $C(e)$  the set of all the descendants of  $e$ . Formally,  $C(e) = C(A(e)) \cup C(B(e)) \cup \{A(e), B(e)\}$ . Finally, we note  $P(e)$  the parent edge of  $e$  :  $P(e) = e' \Leftrightarrow e = A(e')$  or  $e = B(e')$ .

When an edge is cut from the original graph, the equivalent tree can be easily obtain by replacing the tree  $(v, T_a, T_b)$  by  $T_a$  (resp.  $T_b$ ) if the cut edge is before  $T_b$  (resp.  $T_a$ )

**Lemma 1** *Given  $G_k \setminus E' \in \mathcal{G}_k$ ,  $(k - |E'|) \leq \min_{v \in V} h(v)$*