On the Solvability of the Six Degrees of Kevin Bacon Game

A Faster Graph Diameter and Radius Computation Method

Michele Borassi[1], Pierluigi Crescenzi[2], Michel Habib[3], Walter Kosters[4], Andrea Marino[5⋆], and Frank Takes[4]

[1] IMT Institute of Advanced Studies, Lucca
[2] Dipartimento di Sistemi e Informatica, Università di Firenze
[3] LIAFA, UMR 7089 CNRS & Université Paris Diderot - Paris 7
[4] Leiden Institute of Advanced Computer Science, Leiden University
[5] Dipartimento di Informatica, Università di Milano

Abstract. In this paper, we will propose a new algorithm that computes the radius and the diameter of a graph $G = (V, E)$, by finding bounds through heuristics and improving them until exact values can be guaranteed. Although the worst-case running time is $\mathcal{O}(|V| \cdot |E|)$, we will experimentally show that, in the case of real-world networks, it performs much better, finding the correct radius and diameter value after 10–100 BFSes instead of $|V|$ BFSes (independent of the value of $|V|$), and thus having running time $\mathcal{O}(|E|)$. Apart from efficiency, compared to other similar methods, the one proposed in this paper has three other advantages. It is more robust (even in the worst cases, the number of BFSes performed is not very high), it is able to simultaneously compute radius and diameter (halving the total running time whenever both values are needed), and it works both on directed and undirected graphs with very few modifications. As an application example, we use our new algorithm in order to determine the solvability over time of the "six degrees of Kevin Bacon" game.

1 Introduction

The six degrees of separation game is a trivia game which has been inspired by the well-known social experiment of Stanley Milgram [11], which was in turn a continuation of the empirical study of the structure of social networks by Michael Gurevich [7]. Indeed, the notion of six degrees of separation has been formulated for the first time by Frigyes Karinthy in 1929, who conjectured that any two individuals can be connected through at most five acquaintances. This conjecture has somehow been experimentally verified by Milgram and extremely popularized by a theater play of John Guare, successively adapted to the cinema by Fred Schepisi. The corresponding game refers to a social network, such as the

⋆ The fifth author was supported by the EU-FET grant NADINE (GA 288956)

(movie) actor collaboration network, and can be played according to two main different variants. In the first variant, given two vertices x and y of the network, the player is asked to find a path of length at most six between x and y: for instance, in the case of the actor collaboration network, the player is asked to list at most five actors $x_1, \ldots, x_5$ and at most six movies $m_1, \ldots, m_6$ such that x and x_1 played in m_1, x_5 and y played in m_6, and x_i and x_{i+1} played in m_{i+1}, for $1 \leq i \leq 4$. In the second variant of the game, the vertex x is fixed and only the target vertex y is chosen during the game: for instance, in the case of the actor collaboration network, one very popular instance of this variant is the so-called "six degrees of Kevin Bacon" game, where the vertex x is the actor Kevin Bacon, who is considered one of the centers of the Hollywood universe [12]. Many other examples of both variants of the six degrees of separation game are now available on the web: one of the most popular games is the so-called "six degrees of Wikipedia" [4], in which the vertices of the network are the Wikipedia articles and the edges are the links between these articles (here, the network is directed).

In this paper we address the following question: is a given instance of a six degrees of separation game solvable? More generally, is a given instance of a k degrees of separation game solvable? In the case of the second variant of the game, an additional question is the following: which is the choice of vertex x that makes the game solvable? In particular, we will analyze the actor collaboration network, in order to answer these questions, and we will consider the evolution of this network over time, from 1940 to 2014. It will turn out that neither variant of the six degrees of separation game has ever been solvable, since there have always been actors at distance 13 (that is, in order to be solvable the first variant of the game has to choose $k = 13$) and no actor ever existed who could reach all other vertices in less than 7 steps. Moreover, it will turn out that, for the vast majority of the analyzed period, Kevin Bacon has never been the right choice of vertex x (indeed, this happened only in the last two/three years).

Answering the above questions is equivalent to computing the diameter and the radius of a graph, where the diameter is the maximum distance between two connected vertices, and the radius is the distance from a center (that is, a vertex that minimizes the maximum distance to all other vertices) to the vertex farthest from it. Indeed, if the diameter (respectively, radius) of the network used by the game is equal to D (respectively, R), then the two variants of the game are always solvable if and only if $k \geq D$ (respectively, $k \geq R$). Actually, the diameter and the radius are relevant measures (whose meaning depends on the semantics of the network), which have been almost always considered while analyzing real-world networks such as biological, collaboration, communication, road, social, and web networks. Since the size of real-world networks has been increasing rapidly, in order to compute these values, we need algorithms that can handle a huge amount of data. Given a graph $G = (V, E)$, the simplest algorithm to compute the diameter and the radius performs a Breadth-First Search (in short, BFS) from each vertex: the total running time is $\mathcal{O}(|V| \cdot |E|)$ in the worst case, which is too expensive for networks with millions or billions of vertices (especially if we have to compute these values at several different instances in time). As a

consequence, much effort has been spent on improving performance at least in practical cases, by developing algorithms that still have worst-case running time $\mathcal{O}(|V|\cdot|E|)$, but that perform much better in most real-world networks and return the correct values after very few BFSes. In this paper we propose a new and more efficient algorithm to compute radius and diameter. Our algorithm relates the *sweep* approach (i.e. a new visit of the graph depends on the previous one, as in [5, 6, 9, 10]) with the techniques developed in [15, 16]. It is based on a new heuristic, named SumSweep, which is able to compute very efficiently lower bounds on the diameter and upper bounds on the radius of a given graph, and which can be adapted both to undirected and to directed graphs. We will combine the new SumSweep heuristic with the approach proposed in [15] in order to compute the exact values of the radius and of the diameter in the case of undirected graphs, and we will then adapt this combination to the case of directed graphs. We will experimentally verify that the new algorithm significantly reduces the number of required BFSes compared to previously proposed solutions.

Apart from efficiency, the new algorithm has many advantages over the existing ones. First of all, it is able to simultaneously compute the radius and the diameter, instead of making one computation for each of these two parameters. This way, if we are interested in both of them, the total time is halved. Moreover, the new method is much more robust than the previous ones: other algorithms are very fast on well-behaved graphs, but they obtain results which are far from the optimum on particular inputs. The new algorithm is almost equivalent to the existing ones on well-behaved graphs, and it drastically improves performance in the "difficult" cases.

Preliminary notations. In this paper, we address the problem of finding the radius and the diameter of a (strongly) connected (directed) graph. Given an *undirected* graph $G = (V, E)$, the eccentricity of a vertex v is $e(v) := \max_{w \in V} d(v, w)$, where the distance $d(x, y)$ between two vertices x and y is defined as the number of edges contained in a shortest path from x to y. The diameter of G is $\max_{v \in V} e(v)$ and the radius is $\min_{v \in V} e(v)$. Moreover, given a *directed* graph $G = (V, E)$, the forward eccentricity of a vertex v is $e^F(v) := \max_{w \in V} d(v, w)$, the backward eccentricity is $e^B(v) := \max_{w \in V} d(w, v)$. The diameter of G is $\max_{v \in V} e^F(v) = \max_{v \in V} e^B(v)$ and the radius is $\min_{v \in V} e^F(v)$ (in general it is different from $\min_{v \in V} e^B(v)$). Note that in all those definitions the strong connectivity of the graph plays a crucial role.

Structure of the paper. In the rest of this section, we will briefly review the existing methods used to compute the diameter and the radius. Then, in Section 2 we will explain in detail how the new SumSweep heuristic works. Section 3 will show how the eccentricities of all the vertices of a graph can be bounded by making use of a BFS starting from a given vertex. Section 4 will introduce the exact diameter and radius computation algorithm, and finally Section 5 will experimentally demonstrate the effectiveness of our approach. In Section 6, a case study on the actor collaboration network is provided, while Section 7 concludes the paper.

Related Work. Until now, several algorithms have been proposed to approximate or compute the diameter of big real-world graphs. A first possibility is using approximation algorithms with bounded error, like [3, 13]. Another possibility is using heuristics that perform BFSes from random vertices, in order to obtain an upper bound on the radius and a lower bound on the diameter (see for example [14]). This technique is highly biased, because the bounds obtained are rarely tight. More efficient heuristics have been proposed: the so-called 2Sweep picks one of the farthest vertices x from a vertex and returns the distance of the farthest vertex from x [9]; the 4Sweep picks the vertex in the middle of the longest path computed by a 2Sweep and performs another 2Sweep from that vertex [6]. Both methods work quite well and very often provide tight bounds. Adaptations of these methods to directed graphs have been proposed in [2, 5]. Even on directed graphs these techniques provide very good bounds.

However, heuristics cannot guarantee the correctness of the results obtained. For this reason, a major further step in the diameter computation was the design of bound-refinement algorithms. Those methods apply a heuristic and try to validate the result found or improve it until they successfully validate it. Even if in the worst case their time complexity is $\mathcal{O}(|V| \cdot |E|)$, they turn out to be linear in practice. The main algorithms developed until now are BoundingDiameters [15] and iFub [6]. While the first works only on undirected graphs, the second is also able to deal with the directed case (the adaptation is called diFub [5]). For the radius computation, the current best algorithm is a modification of the BoundingDiameters algorithm [16]. It is also possible to use the method in [10], but this always requires the computation of all central vertices of the graph.

2 Bounding the Radius and Diameter using SumSweep

Undirected Case. The idea behind the SumSweep heuristic is finding "key vertices" in the computation of the radius and the diameter of a graph. It is based on the simple observation that the well-known *closeness* centrality measure [1] can be a good indicator for *eccentricity* when applied to the most and least *central* vertices of a network. Moreover, given vertices $v_1, \ldots, v_k$, the value $\sum_{i=1}^{k} d(v_i, w)$ can give an idea about the *closeness* centrality of a vertex w in a real-world network (hence, of its eccentricity). In particular, if the sum is big, the considered vertex is more likely to be peripheral, so it is a good candidate to be a vertex with maximum eccentricity. Conversely, if this sum is small, the vertex is probably central. These intuitions are formalized by the following propositions.

Proposition 1. *Let D be the diameter, let x and y be diametral vertices (that is, $d(x, y) = D$), and let $v_1, \ldots, v_k$ be other vertices. Then, $\sum_{i=1}^{k} d(x, v_i) \geq \frac{kD}{2}$ or $\sum_{i=1}^{k} d(v_i, y) \geq \frac{kD}{2}$.*

Proof. $kD = \sum_{i=1}^{k} d(x, y) \geq \sum_{i=1}^{k} [d(x, v_i) + d(v_i, y)] = \sum_{i=1}^{k} d(x, v_i) + \sum_{i=1}^{k} d(v_i, y)$. □

Proposition 2. *Let R be the radius and let $x \in V$ be such that $max_{y \in V} d(x, y) = R$, and let $v_1, \ldots, v_k$ be other vertices. Then $\sum_{i=1}^{k} d(x, v_i) \leq kR$.*

The previous intuition is the basis of the undirected SumSweep heuristic, that provides a lower bound for the diameter and an upper bound for the radius, by finding vertices $v_1, \ldots, v_k$ that are peripheral and well distributed within the graph. More formally, a k-SumSweep is the following procedure:

- Given a random vertex v_1 and setting $i = 1$, repeat k times the following:
  1. Perform a BFS from v_i and choose the vertex v_{i+1} as the vertex x maximizing $\sum_{j=1}^{i} d(v_j, x)$.
  2. Increment i.
- The maximum eccentricity found, i.e. $\max_{i=1,\ldots,k} e(v_i)$, is a lower bound for the diameter.
- Compute the eccentricity of w, the vertex minimizing $\sum_{i=1}^{k} d(w, v_i)$. The minimum eccentricity found, i.e. $\min\{\min_{i=1,\ldots,k} e(v_i), e(w)\}$, is an upper bound for the radius.

We can also impose that $v_i \neq v_j$ and $v_i \neq w$: indeed, if this is not the case, then we can simply choose v_j or w as the best vertex different from the previous ones.

Directed Case. The main ideas of the previous method can also be applied to strongly connected directed graphs. However, in such a context it is necessary to take into account that the distance $d(v, w)$ does not necessarily coincide with $d(w, v)$. Similarly to the previous case, we define two closeness centrality indicators, one for forward eccentricity and one for backward eccentricity: a vertex v is a *source* (respectively, *target*) if $d(v, w)$ (respectively, $d(w, v)$) is high on average. Note that there might be vertices that are both sources and targets. Analogously, Propositions 1 and 2 still hold.

The definition of directed SumSweep is very similar to the undirected case, with the difference that the BFSes are performed alternating their direction. More formally, we do the following:

- Given a random vertex s_1 and setting $i = 1$, repeat $k/2$ times the following.
  1. Perform a forward BFS from s_i and choose the vertex t_i as the vertex x maximizing $\sum_{j=1}^{i} d(s_j, x)$.
  2. Perform a backward BFS from t_i and choose the vertex s_{i+1} as the vertex x maximizing $\sum_{j=1}^{i} d(x, t_j)$.
  3. Increment i.
- The maximum eccentricity found, which is the maximum of the two values $\max_{i=1,\ldots,k/2} e^F(s_i)$ and $\max_{i=1,\ldots,k/2} e^B(t_i)$, is a lower bound for the diameter.
- Compute the eccentricity of w, the vertex minimizing $\sum_{i=1}^{k/2} d(w, t_i)$. The minimum eccentricity found, i.e. $\min\{\min_{i=1,\ldots,k/2} e^F(s_i), e^F(w)\}$, is an upper bound for the radius.

Once again, we impose $v_i \neq v_j$ and $v_i \neq w$.

3 Bounding the Eccentricities of the Vertices

This section aims to show some bounds on the eccentricity of the vertices. In particular, we will explain how to lower and upper bound the eccentricity of a vertex w, using a BFS from another vertex v.

Undirected Case. Suppose we have performed a BFS from a vertex v, forming the BFS tree T, and we want to use the resulting information to bound the eccentricity of all other vertices. The following observation can provide an upper bound, while we will use $L_v(w) := d(v, w)$ as a lower bound.

Lemma 1. *Let v' be the first vertex in T having more than one child. Let Φ be the set of vertices on the (only) path from v to v', let Ψ be the set of vertices in the subtree of T rooted at the first child of v', and let h be the maximum distance from v' to a vertex outside Ψ. Then, for each $w \in V$, $e(w) \leq U_v(w)$, where*

$$U_v(w) := \begin{cases} \max(d(v,w), e(v) - d(v,w)) & w \in \Phi \\ \max(d(v',w) + e(v') - 2, d(v',w) + h) & w \in \Psi \\ d(v',w) + e(v') & \textit{otherwise} \end{cases}$$

Proof. If $w \in \Phi$ or $w \notin \Phi \cup \Psi$, the conclusion follows easily by the triangle inequality. If $w \in \Psi$, let x be the farthest vertex from w: if $x \notin \Psi$, then $d(x, w) \leq d(x, v') + d(v', w) \leq h + d(v', w)$. If $x \in \Psi$ and r is the root of the subtree of T consisting of vertices in Ψ, $d(w, x) \leq d(w, r) + d(r, x) = d(w, v') + d(v', x) - 2 \leq d(w, v') + e(v') - 2$. □

Note that all values appearing in the definition of $L_v(w)$ and $U_v(w)$ can be computed in linear time by performing a BFS from v.

Directed Case. In this case, the previous bounds do not hold: we will use a weaker version, based on the following lemma, whose proof is straightforward.

Lemma 2. *Let $L_v^F(w) := d(w, v)$, $L_v^B(w) := d(v, w)$, $U_v^F(w) := d(w, v) + e^F(v)$ and $U_v^B(w) := d(v, w) + e^B(v)$. Then, for each $v, w \in V$,*

$$L_v^F(w) \leq e^F(w) \leq U_v^F(w) \qquad \textit{and} \qquad L_v^B(w) \leq e^B(w) \leq U_v^B(v).$$

Note that L_v^F (resp. L_v^B) can be computed through a backward (forward) visit from v, while to compute the upper bounds we need both a forward and a backward visit from v.

4 Computing Radius and Diameter

In order to exactly compute the radius and diameter, we apply the technique of BoundingDiameters algorithm, improved through the use of SumSweep and generalized to directed graphs. Generally speaking, our algorithm refines lower and upper bounds on the eccentricities of vertices, until the correct eccentricity is found.

Undirected Case. The algorithm maintains two vectors e_L and e_U of lower and upper bounds on the eccentricity of all vertices, and a vector S containing the sum of distances from the starting points of previous BFSes.

Every time a BFS is performed from a vertex u, for each $v \in V$ $e_L[v]$ (resp. $e_U[v]$) is updated with $\max(e_L[v], L_u(v))$ (resp. $\max(e_U[v], U_u(v))$), and $S[v]$ is updated with $S[v] + d(u, v)$. Let us denote by X the set of vertices v such that $e_L[v] < e_U[v]$ and by Y the set $V - X$. It is worth observing that for any $v \in Y$ we have $e(v) = e_L[v] = e_U[v]$.

At the beginning, for each v, $e_L[v] = 0$ and $e_U[v] = +\infty$. The algorithm starts by performing k iterations of SumSweep (according to our preliminary experiments, $k = 3$ or $k = 4$ is the best), updating e_L and e_U after each BFS. Then, at each step, a vertex u is *selected* from the set X and a BFS starting from u is performed, updating lower and upper bounds.

Termination. The radius is found when $\min_{y \in Y}(e(y)) \leq \min_{x \in X}(e_L[x])$, and the value is $\min_{y \in Y}(e(y))$. Analogously, the diameter is found when $\max_{y \in Y}(e(y)) \geq \max_{x \in X}(e_U[x])$, and its value is $\max_{y \in Y}(e(y))$.

The selection of vertex u is crucial to speed up the computation. At each step, we alternate the following two choices:

1. choose a vertex $u \in X$ minimizing $e_L[u]$;
2. choose a vertex $u \in X$ maximizing $e_U[u]$.

In order to break ties (which occur very often), we use the vector S: in the first case, we minimize $S[v]$ and in the second case we maximize it. Intuitively, the first choice should improve upper bounds, while the second choice should improve lower bounds.

Although the algorithm could perform $\mathcal{O}(|V|)$ BFSes in the worst case, we will show that in practice it needs just $\mathcal{O}(1)$ BFSes.

Directed Case. In the directed case, we need to maintain two vectors (e_L^F and e_L^B) containing lower bounds on forward and backward eccentricity, respectively, and other two vectors (e_U^F and e_U^B) containing upper bounds. Moreover, we need to keep two vectors S^F and S^B containing the sum of forward and backward distances from the starting points of previous BFSes.

Every time a forward visit is performed from a vertex u, $e_L^B[v]$ is updated with $\max(e_L^B[v], L_u^B(v))$ and $S^B(v)$ is updated with $S^B(v) + d(u, v)$ (the backward case is analogous). In order to update upper bounds, we need to perform both a forward and a backward visit from a vertex u, and in that case the new value of $e_U^F[v]$ is $\min(e_U^F[v], U_v^F(w))$ and the new value of $e_U^B[v]$ is $\min(e_U^B[v], U_v^B(w))$. Let us denote by X^F (resp. X^B) the set of vertices v such that $e_L^F[v] < e_U^F[v]$ (resp. $e_L^B[v] < e_U^B[v]$), by Y^F (resp. Y^B) the set $V - X^F$ (resp. $V - X^B$). Observe that for any $v \in Y^F$ (resp. Y^B) we have $e^F(v) = e_U^F[v] = e_L^F[v]$ (resp. $e^B(v) = e_U^B[v] = e_L^B[v]$).

At the beginning, for each v, all lower bounds are set to 0, all upper bounds are set to $+\infty$, S^F and S^B are set to 0. The algorithm starts by performing k iterations of SumSweep (according to our preliminary experiments, $k = 6$ is the best), updating lower and upper bounds after each BFS. Then, at each step,

a vertex u is selected and a BFS starting from u is performed, updating lower and upper bounds.
Termination. The radius is found when $\min_{y \in Y^F}(e^F(y)) \leq \min_{x \in X}(e_L^F[x])$, and the value is $\min_{y \in Y^F}(e^F(y))$. Analogously, the diameter is found when

$$\max(\max_{y \in Y^B}(e^B(y)), \max_{y \in Y^F}(e^F(y))) \geq \min(\max_{x \in X^F}(e_U^F[x]), \max_{x \in X^B}(e_U^B[x])).$$

The diameter value is then the left side of this inequality.

Once again, the selection of the vertex for the next visit is crucial for the efficiency of the algorithm. The choices are made alternating the following strategies (in the order in which they appear).

1. Choose a vertex $u \in X^F$ which minimizes $e_L^F[u]$ and perform a forward BFS.
2. Choose a vertex $u \in X^F \cap X^B$ minimizing $e_L^F[u] + e_L^B[u]$, and perform a forward and backward BFS.
3. Choose a vertex $u \in X^B$ maximizing $e_U^B[u]$ and perform a backward BFS.
4. Choose a vertex $u \in X^F$ maximizing $e_U^F[u]$ and perform a forward BFS.
5. Repeat Item 2.

In Items 1, 2 and 5 we break ties by choosing u minimizing $S^F[u]$, in Item 3 and 4 by maximizing $S^B[u]$ and $S^F[u]$, respectively. Intuitively, Item 1 aims to improve the forward upper bound of u (in order to find the radius), Items 2 and 5 aim to improve upper bounds on all the vertices; both Item 3 and Item 4 aim to improve lower bounds: in particular, Item 3 improves the forward eccentricity, while Item 4 improves the backward eccentricity.

5 Experimental Results

In order to compare the different methods, we analyzed a dataset of 34 undirected graphs and 29 directed graphs, taken from the Stanford Large Network Dataset Collection. This dataset is well-known and covers a large set of network types (see [14] for more details). These experiments aim to show that the SumSweep method improves the time bounds, the robustness, and the generality of all the existing methods, since they are outperformed for both radius and diameter computation, both in the directed and in the undirected case.

More detailed results about the comparison, together with the code used, are available at `amici.dsi.unifi.it/lasagne`.

Undirected Case. In the undirected case, we compared our method with the state of the art: the iFub algorithm for the diameter and the BoundingDiameters (BD) algorithm both for the radius and for the diameter.

Indeed, this latter algorithm, used in [16] just to compute the diameter, can be easily adjusted to also compute the radius, using the same vertex selection strategy and updating rules for the eccentricity bounds. In particular, it bounds the eccentricity of vertices similarly to our method, by using the fact that, after a visit from a vertex v is performed, $d(v, w) \leq e(w) \leq d(v, w) + e(v)$ (it is a

Table 1. The average performance ratio p, percentage of the number of BFSes used by the different methods, with respect to the number of vertices (number of visits in the worst-case)

Method	p	Std Error
SumSweep	0.023 %	5.49E-5
BD	0.030 %	9.62E-5

(a) Radius in Undirected Graphs

Method	p	Std Error
SumSweep	0.084 %	2.73E-4
BD	0.538 %	2.77E-3
iFubHd	>0.677 %	3.34E-3
iFub4S	>1.483 %	7.72E-3

(b) Diameter in Undirected Graphs

Method	p	Std Error
SumSweep	0.27 %	8.02E-4
HR	>3.20 %	8.51E-3

(c) Radius in Directed Graphs

Method	p	Std Error
SumSweep	0.39 %	1.23E-3
diFubHdIn	3.06 %	1.47E-2
diFubHdOut	2.37 %	1.03E-2
diFub2In	1.12 %	4.68E-3
diFub2Out	1.02 %	4.45E-2

(d) Diameter in Directed Graphs

weaker version of Lemma 1). It does not perform the initial SumSweep and simply alternates between vertices v with the largest eccentricity upper bound and the smallest eccentricity lower bound.

For the diameter computation, we compared SumSweep not only with BD, but also with two variations of iFub: iFubHd, starting from the vertex of highest degree, and iFub4S, starting by performing a 4Sweep and choosing the central vertex of the second iteration (see [6] or the section on related work for more details about 2Sweep and 4Sweep).

The results of the comparison are summarized in Table 1: for each method and for each graph in our dataset, we have computed the corresponding *performance ratio*, that is the percentage of the number of visits performed by the method with respect to the number of vertices of the network (i.e. the number of visits in the worst case). In Table 1 we report the average of these values together with the corresponding standard error.

In the radius computation, the SumSweep method is slightly more effective than the BD algorithm. It is also more robust: in our dataset, it never needs more than 18 BFSes, while the BD algorithm needs at most 29 BFSes. Moreover, there are only 3 graphs where the BD algorithm beats the SumSweep algorithm by more than one BFS.

In the diameter computation, the improvement is even more evident in Table 1 (b). Again, we see that the new method is much more robust than the previous ones: the computation of the diameter for SumSweep always ends in less than 500 BFSes, while the old methods need up to 5000 BFSes.

Directed Case. In the directed case, the only efficient known method to compute the radius is explained in [10], which we will refer to as HR. Basically, it works as follows: given the farthest pair of vertices x and y found by the directed version of 2Sweep, order the vertices v according to $g(v) = \max\{d(v,x), d(v,y)\}$; scan the eccentricities of the vertices in this order and stop when the next vertex w has a value of $g(w)$ which is greater than the minimum eccentricity found. It is easy to see that all the vertices with minimum eccentricity must always be scanned (which is not necessary for our algorithm).

Since this method is the only algorithm to compute the radius, we compared our method just with this one. The results are shown in Table 1(c).

For the diameter computation, we compared our results to the four variations of the diFub method:

diFubHdIn: starts from the vertex with highest in-degree;
diFubHdOut: starts from the vertex with highest out-degree;
diFub2In: starts from the central vertex of a 2Sweep performed from the vertex with highest in-degree;
diFub2Out: starts from the central vertex of a 2Sweep performed from the vertex with highest out-degree.

The results are shown in Table 1(d).

In the radius computation, the SumSweep algorithm performs about 12 times better than the old method. We also remark that the robustness of SumSweep applies also to the directed case: at most 40 BFSes are needed to find the radius of any graph of our dataset.

In the diameter computation, the best previous method is diFub2Out: the new SumSweep method performs about 2.5 times better. We note again the robustness: the maximum number of BFSes is 93, against the maximum number for diFub which is 482.

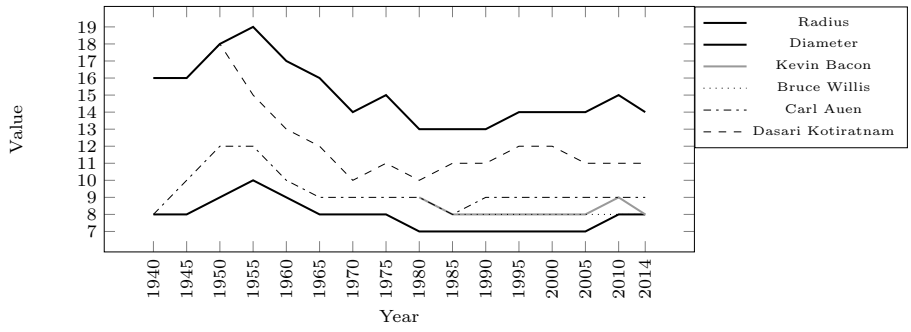
Overall, we conclude that the new method is more general (it is the only one which is able to deal with both directed and undirected cases, both in the radius and in the diameter computation), more robust, and more efficient than the best previous methods.

Finally, we observe that, for each of the algorithms considered, the number of BFSes for computing the radius or diameter is very low (often no more than 5) when $D \approx 2R$. When $D < 2R$, then there are two other factors that appear to influence performance. First, the relation $D \,/\, 2R$ between the diameter and radius appears to be of influence: the closer this value is to 1, the faster the computation, in most cases. Second, the actual value of the diameter itself plays a role: the diameter of graphs with a very small diameter is often harder to compute, as there is little diversity in the eccentricity values and therefore little opportunity for vertices to effectively influence the lower and upper eccentricity bounds of neighboring vertices.

6 Internet Movies Database Case Study

This section applies the SumSweep algorithm to the Internet Movies Database, in particular to the so-called actor graph, in which two actors are linked if they played together in a movie (we ignore TV-series in this work). All data have been taken from the website `http://www.imdb.com`. According to [12], we decided to exclude some genres from our database: awards-shows, documentaries, game-shows, news, realities and talk-shows. We analyzed snapshots of the actor graph, taken every 5 years from 1940 to 2010, and 2014.

Fig. 1. Actor graph evolution in terms of radius, diameter, and actor eccentricity.



Running Time Analysis. First, we compared the performances of our algorithm to the BoundingDiameters method. Similarly to the previous experiments, we found that the new method improves the previous one in the diameter computation, and it has similar results in the radius computation. However, in this latter case, the new method needed a smaller number of BFSes for big actor graphs (the most recent ones), where a BFS is more expensive in terms of computation time.

Analysis of the Actor Graph. Figure 1 shows the evolution of the diameter, the radius and the eccentricity of some actors. It shows that the radius and diameter of the graph increased between 1940 and 1955, then they started decreasing, as also observed in [8] as a property of large evolving graphs. The first increase might be explained by the fact that the years between the forties and the sixties are known as the golden age for Asian cinema, especially Indian and Japanese. This trend is also confirmed by the names of the central actors during that period. In 1940, they are almost all western, usually German (like for instance Carl Auen). By 1955, we find both western and eastern actors. Later, in the sixties, the increase of independent producers and production companies led to an increase of power of individual actors. This can explain the decreasing size of the graph during those years: the number of contacts between actors from different countries increased (consider for instance the first James Bond movie, *Dr. No*). For further historical film information we refer the reader to [17]. The decreasing of the graph diameter and radius halted in the eighties, and there were little changes until the present. Now it seems that the radius is slightly increasing again, but the number of central actors is increasing as well.

Almost all actors seem to have decreasing eccentricity over time, even actors that are no longer active (like Dasari Kotiratnam and Carl Auen). Instead, the periphery is usually made by recent actors. We finally remark that Kevin Bacon has not minimum eccentricity until the present, and he never gets eccentricity 6, as required by the game "Six Degrees of Kevin Bacon". Hence not all the actors can be linked to Kevin Bacon by using at most 6 edges.

7 Conclusion

In this paper, we proposed a new heuristic to upper and lower bound respectively the radius and diameter of large graphs and a new algorithm for computing their exact value. We performed experiments on a large number of graphs, including the IMDb actor graph of which we analyzed the radius, diameter and actor eccentricity over time in order to verify the hypothesis of six degrees of separation.

In future work we would like to investigate theoretically how the observations from the experiments regarding the link between the diameter, radius and the number of BFSes can be exploited in the diameter and radius computation itself.

References

1. Bavelas, A.: Communication Patterns in Task-Oriented Groups. The Journal of the Acoustical Society of America 22(6), 725–730 (Nov 1950)
2. Broder, A.Z., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.L.: Graph Structure in the Web. Computer Networks 33(1-6), 309–320 (2000)
3. Chechik, S., Larkin, D., Roditty, L., Schoenebeck, G., Tarjan, R.E., Williams, V.V.: Better approximation algorithms for the graph diameter. In: SODA. pp. 1041–1052 (2014)
4. Clemesha, A.: The Wiki Game. `http://thewikigame.com` (2013)
5. Crescenzi, P., Grossi, R., Lanzi, L., Marino, A.: On Computing the Diameter of Real-World Directed (Weighted) Graphs. In: SEA. pp. 99–110 (2012)
6. Crescenzi, P., Grossi, R., Habib, M., Lanzi, L., Marino, A.: On Computing the Diameter of Real-World Undirected Graphs. Theor. Comput. Sci. 514, 84–95 (2013)
7. Gurevich, M.: The Social Structure of Acquaintanceship Networks (1961), PhD Thesis
8. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graph Evolution: Densification and Shrinking Diameters. TKDD 1(1) (2007)
9. Magnien, C., Latapy, M., Habib, M.: Fast Computation of Empirically Tight Bounds for the Diameter of Massive Graphs. Journal of Experimental Algorithmics 13 (2009)
10. Marino, A.: Algorithms for Biological Graphs: Analysis and Enumeration (2013), PhD Thesis, Dipartimento di Sistemi e Informatica, University of Florence
11. Milgram, S.: The Small World Problem. Psychology Today 2, 60–67 (1967)
12. Reynolds, P.: The Oracle of Kevin Bacon. `http://oracleofbacon.org` (2013)
13. Roditty, L., Williams, V.V.: Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs. In: STOC. pp. 515–524 (2013)
14. SNAP: Stanford Network Analysis Package (SNAP) Website. `http://snap.stanford.edu` (2009)
15. Takes, F.W., Kosters, W.A.: Determining the Diameter of Small World Networks. In: CIKM. pp. 1191–1196 (2011)
16. Takes, F.W., Kosters, W.A.: Computing the Eccentricity Distribution of Large Graphs. Algorithms 6(1), 100–118 (2013)
17. Thompson, K., Bordwell, D.: Film History: an Introduction. McGraw-Hill Higher Education (2009)