Parallel Execution of Schedules with Random Dependency Graph

Bogumił Kamiński[*] Tomasz Olczak[†] Paweł Prałat[‡]

April 4, 2019

Abstract

We consider schedules of unit-duration tasks with random dependencies. Formally the schedules are obtained by imposing a random partial order of vertices in an Erdős-Rényi graph. For such schedules, we provide asymptotic formulas for the parallel execution time and for the required number of processors in a greedy allocation scheme as a function of connection probability. We also derive asymptotic bounds for the number of required processors in an optimal allocation scheme. We test our results for small schedule sizes using simulation and conclude that the convergence to asymptotic results is achieved relatively fast in practice. In our simulation experiment, we define and compare the efficiency of Mixed Integer Programming and Constraint Programming approaches to find the exact solution of the optimal allocation scheme and conclude that Constraint Programming is more efficient in our setting. In the last part of the paper, we provide preliminary results of similar analysis for random schedules generated from arbitrary graphs.

1 Introduction

In this text we study the limits on parallelization of schedules with random precedence constraints. This generic resource-constrained scheduling problem has important applications in many domains, e.g. project management [15, 36], simulation scheduling [13], analysis of citation networks or food webs [25], operations research [5, 10, 11, 34] and in computer science for study of parallel algorithms [12, 29, 38, 41].

While finding minimum parallel execution time for the considered scheduling problem is relatively simple, identification of an optimal schedule for a limited number of workers is known to be NP-hard [24, 28, 38]. Therefore many scheduling heuristics were proposed in the literature, aimed at narrowing a performance

[*]SGH Warsaw School of Economics, Poland, e-mail: bkamins@sgh.waw.pl
[†]SGH Warsaw School of Economics, Poland and Ryerson University, Canada
[‡]Ryerson University, Canada, e-mail: pralat@ryerson.ca

gap between optimal and heuristic scheduling policies [5, 10, 20, 34]. Much less attention was given to theoretical analysis of gains from parallelization of different classes of processes with precedence constraints, even though this problem has been growing in importance with proliferation of massively parallel computing platforms whose effective utilization requires fine-grained parallelization of applications [14, 26].

In the literature acyclic nature of schedules is often assumed to arise from ordering vertices of a graph and we follow the same approach in this paper [25]. Formally, we assume that a schedule is produced by generating Erdős-Rényi random graph and then randomly ordering its vertices. Such a setting, known as *random graph order*, has already received theoretical treatment by [4] and [7] for the size of a largest set of strongly independent vertices, [2] for the number of linear extensions of partial orders, [8] for the dimension of random graph orders, [37] for the distribution of transitive closure, and [30] for the convergence of first order properties on transitive closure of such posets.

In order to further motivate the proposed approach we present an example taken from agent-based modeling in Section 2 below. Agent-based models are computationally intensive and, at the same time, characterized by high degree of inherent parallelism. This makes them good candidates for running on parallel processors. We answer the question what speed-up one could expect from parallelization of such applications and what is the optimal number of parallel workers to be used for computation. Specifically, we are interested in the following three factors (they are formally defined in Section 3):

1) shortest time required to finish parallel execution of a schedule, denoted as Y;
2) number of processors required when greedy allocation scheme is applied, denoted as U;
3) number of processors required when optimal allocation scheme is applied, denoted as Q.

The difference between greedy and optimal allocation schemes is that in *greedy allocation* we start executing tasks as soon as it is possible (implicitly assuming that we have an infinite number of processors available at no cost) and in *optimal allocation* we ask for the minimum number of processors required to finish parallel execution of a schedule in time Y (where finding such optimal allocation is, of course, an independent and difficult task).

With respect to quantities Y, U and Q, according to our present knowledge, only Y (shortest time required to finish parallel execution) has been studied in the literature. In [13], an analysis of Y using simulation for small sizes of schedules is provided. A survey of literature in other research fields provides the following formal results on asymptotic properties of Y. The authors of [1] analyze it for constant $p \geq 0.1$ and obtain the bound that a.a.s. $0.5654n < Y < 0.610n$ for $p = 1/2$. We extend these results for varying p, including the case when $p = p(n) \to 0$ as $n \to \infty$. Similarly, the authors of [9] studied Y but only for $p \to 0$ and $p \geq (1+\varepsilon)\pi^2/\log n$ for some $\epsilon > 0$. Hence, they concentrate on very dense graphs. In this text we consider also sparse graphs, that in fact are actually more interesting from applications point of view. Papers [32] and [33] also studied Y; however, the tools that are used in [32] for dense graphs (that is, when $pn = \Omega(\log n)$) are different than ours, and [33] concentrates only on the case $pn = c \in \mathbb{R}_+$. Our approach and tools are different and, more importantly, they provide some additional insight about the structure of the random directed graph we deal with (which may have implications in designing efficient algorithms in practice). In particular, our approach provides bounds for both U and Q—to the best of our knowledge, these values were not studied in detail in the existing literature. Finally, we close the missing (but important from the practical point of view) range of $p = p(n)$ when $pn \gg 1$ but $pn \ll \log n$ in evaluation of Y.

The rest of the text is organized as follows. In Section 2, as already mentioned, we discuss the motivating example for our study taken from agent-based simulation. Next, in Section 3, we provide analysis of Y, U and Q for random orders. Finally, in Section 4 we provide some preliminary results for schedules generated by random ordering of vertices in arbitrary graphs.

2 Motivating example

Agent-based simulations are becoming one of the standard tools in research and applications as they allow to investigate models of complex systems which would otherwise be intractable [21, 39, 40]. A basic assumption of *Agent-Based Model* (ABM) is that systems being modeled are composed of autonomous decision makers called agents who interact with each other. A typical execution pattern is that agents repeatedly perform actions which influence themselves and other agents in their neighborhood. Formally, let V be a set of agents. Action of agent $a \in V$ influences some subset of agents $E_a \subseteq V$. Usually $|E_a|$ is much smaller than $|V|$, i.e. an agent's action has only *local* effect on the whole system. Globally, the structure of a population is then naturally described by a dependency graph $G = (V, E)$ in which set of vertices V represents agents and set of edges $E = \{(a, b) : a \in V, b \in E_a)\}$ represents their interrelationships.

Large-scale, agent-based simulations are computationally intensive and it is desirable to run them in parallel environments to speed up execution through collaboration of processing units. This requires an application to be divided into subtasks to distribute computational load among processors. Due to their collective nature ABMs decompose naturally into subtasks defined as individual actions taken by agents. Hence in ABM the degree of available parallelism depends on two factors: (a) number of agents and (b) structure and nature of their interactions. In principle, the larger the set of agents the more parallelism is possible as more agents may take actions concurrently. However, if time dependency of agents' actions is not carefully treated, then a synchronization failure may occur and time consistency of a simulation may be violated resulting in *causality errors* [18, 27]. This problem is greatly amplified on massively parallel processors with thousands of processing cores.[1] To protect against causality er-

[1] such as programmable GPUs or clusters of many-core CPUs

rors a parallel simulation engine must maintain and observe a *precedence graph* which defines a partial order of agents' tasks as induced by the dependency graph G and their activation sequence. Adhering to this order is essential for the correct execution of the application [38].

Formally, a task precedence graph is a directed acyclic graph (DAG) implementing topological ordering of agents' actions. In agent-based simulations agents are usually activated asynchronously, so the order of their actions can be described by a random permutation of V. We will call such a permutation an execution order $\sigma\colon V \to \{1, 2, \dots, |V|\}$ that is a bijective mapping. In a single threaded execution, agents are simply activated in a sequence given by σ. In contrast, on parallel hardware it is possible to process up to m agents concurrently, where m is a number of available processors. But the degree of available concurrency is constrained by a dependency graph G. If $\sigma(b) > \sigma(a)$ and $b \in E_a$, then action of agent b must be executed later than action of agent a (in particular, they cannot be processed concurrently). Hence, for a given execution order σ, a dependency graph G induces a directed graph $G' = (V, D)$, such that $(a, b) \in D \iff \sigma(a) < \sigma(b) \wedge (a, b) \in E$, imposing a partial ordering of actions (in graph theory G' is known as *random graph order*).

Given G', a natural challenge is to process it most efficiently on parallel hardware. This problem, known as optimal scheduling, has been studied extensively in computer science, see e.g. [5, 10, 34]. While finding minimal execution time Y is relatively simple using topological sorting of G', optimal scheduling on parallel hardware is an NP-hard problem [24, 28, 38]. Therefore, in this text we are interested not only in Q (an optimal number of processors) but also in U (number of processors required in greedy allocation scheme), which does not require solving the optimal allocation problem. The greedy scheme is also called *list scheduling* in the literature [19].

3 Bounds on parallelization of random graph orders

3.1 Model of parallel graph traversal

Sticking with the ABM example from Section 2 let $G = (V, E)$ be a graph representing how actions of agents influence each other. In general, this is a directed graph but very often in practical applications it can be assumed to be an undirected graph. By $n = |V|$ we denote the number of agents. In this paper, we concentrate on two models for G leaving other types of graphs to be considered in the future:

1) binomial random graph $\mathbb{G}(n, p)$ (of our main interest, analyzed in this section);
2) arbitrary graph (preliminary results and open problems given in Section 4).

We consider the random graph $\mathbb{G}(n, p)$ model as a reference scenario. We have chosen to concentrate our analysis on it as it is possible to analytically show its properties and it is naturally related to random DAG literature. In practice other graph models might be more relevant therefore in Section 4 we provide preliminary results for general graphs.

Let us assume a bit more general setting than the one presented in Section 2 and presume that there are T actions in the simulation. The mapping $A\colon \{1,\dots,T\} \to V$ tells us in which sequence agents act. The following two models for A are of interest:

1) sample without replacement of size $T = n$; that is, A corresponds to a random permutation of V (σ considered in Section 2);
2) sample with replacement of arbitrary size T; that is, independently for each $i \in \{1,\dots,T\}$ and $v \in V$, $\mathbb{P}(A(i) = v) = 1/n$.

In the present paper we concentrate on the former scenario, random permutation. Again the choice was guided by analytical tracability and the fact that it gives a natural link to random DAG literature.

Note, however, that even this simple scenario is often encountered in agent based modeling practice. It is frequently assumed that all agents act exactly once in a random order in each turn (typically called *tick*) of the model. After simulating a single tick the whole simulation has to be synchronized (e.g. to calculate aggregate statistics of the modelled system). This situation corresponds exactly to a random permutation of V scenario we consider. Actually this kind of behaviour is a default one in one of the most popular agent-based modeling frameworks NetLogo, [42]. The latter scenario of sampling with replacement is left for future investigation.

Definition 3.1 (**Greedy scheme**)**.** Given a graph $G(V, E)$ and a mapping $A\colon \{1,\dots,T\} \to V$, we create a directed graph $S_0 = (V_0, E_0)$, where $V_0 = \{1,\dots,T\}$ and $E_0 \subseteq V_0^2$; there exists an edge $(i, j) \in E_0$ if and only if $i < j$ and $(A(i), A(j)) \in E$ or $A(i) = A(j)$. Now, we recursively create a sequence $(S_k)_{k\geq 0}$ of directed graphs. Given S_k for some $k \in \mathbb{N} \cup \{0\}$, let X_k be the set of vertices of S_k of in-degree 0. Finally, S_{k+1} is obtained from S_k by removing vertices from X_k. Clearly, as long as S_k contains at least one vertex, $|X_k| \geq 1$ (in particular, the vertex of the smallest label in S_k belongs to X_k). As a result, at some time-step $Y \leq n$ of the process S_Y is the null graph; that is, the graph with no vertex. In other words, let Y be the smallest natural number k for which the graph S_k is the null graph. Finally, let $U := \max\big\{|X_k| : k \in \mathbb{N} \cup \{0\}\big\}$.

Before we move to the next definition, let us make a few comments:

1) An edge (i, j) indicates that action j cannot be performed before action i;
2) Since for $i \geq j$ we never create an edge from i to j, graph S_0 (and so also S_k for any k) is acyclic;
3) Set X_k represents a set of actions that can be performed, in parallel, at step k;
4) We may assume that the process ends at time-step Y as $S_k = S_Y$ (the null graph) for any $k \geq Y$;
5) Since G and A are created at random, Y is a random variable; however, as already pointed out in the above definition, deterministically $Y \leq T$;

6) Value $Y = T$ is achieved when, for example, G is a directed path $1 \to 2 \to \ldots \to n$, $T = n$, and mapping A is $A(i) = i$. In this scenario, S_0 is also a directed path;
7) U is the number of parallel processors needed in the greedy scheme.

The following observation, also simple but very useful, will be employed a few times and so it deserves special treatment and label.

Observation 3.2. It is clear that in order to minimize Y, the number of rounds the process lasts, one should remove all vertices in X_k. However, alternatively, one could select some proper subset of vertices of in-degree 0 to form X_k. For any possible rule, the corresponding sequence of directed graphs $\hat{S}_k = (\hat{V}_k, \hat{E}_k)$ will have the property that $V_k \subseteq \hat{V}_k$ and $\hat{Y} \geq Y$. Formally, the proof by induction of this fact uses the property that once a vertex is ready to be removed from S_k (that is, has in-degree 0), it will remain in such state till it is actually removed.

Why one would want do do it? There are at least two reasons for that. The first one is technical: in some proofs we provide in this paper it will be convenient to pick some specific subsets of vertices of in-degree 0 to get an upper bound for Y. The second one is more important as it has some practical implications. As already mentioned, $U = \max_k\{|X_k|\}$ corresponds to the number of processors needed to run the simulation in parallel. However, we usually want to keep the number of required processors as small as possible. Hence, very often one can gain more by selecting X_k sub-optimally (and so reducing the number of processors needed) in comparison to (possibly) some small increase of the number of rounds. This trade-off is not always simple to judge, but reducing the number of processors needed without increasing the number of rounds is always beneficial. This motivates the next definition.

Definition 3.3 (**Optimal scheme**)**.** Let $\mathcal{R}$ be the family of all rules with the property that the corresponding sequence of directed graphs $\hat{S}_k$ has the property that $\hat{Y} = Y$. (Clearly $\mathcal{R}$ is non-empty as the rule following the greedy scheme that yields Y belongs to this family.) For a given rule $R \in \mathcal{R}$, let $Q_R = \max_k\{|X_k|\}$, where (X_k) is the sequence of the corresponding subsets of in-degree 0 that are removed during the process following R. (In other words, Q_R is the number of processors needed to execute rule R.) Finally, let $Q = \min_{R \in \mathcal{R}} Q_R$.

The graph parameter Q is both interesting and important because, in general, one would like to use a minimum number of parallel processors to finish the task without compromising the running time. Clearly, the number of processors used in a greedy scheme, U, could be much larger than Q. Consider, as an example, a graph consisting of an n-element directed path and n isolated vertices (together with an ordering A consistent with the ordering of the directed path). In this example, $Y = n$, $U = n + 1$ (in the first phase we use $n + 1$ processors), but $Q = 2$ (in each phase we use one processor for a vertex from the path and one processor for an isolated vertex).

In summary, given some process (possibly random) that generates graph G, assuming that $T = n$ and A is a random permutation, we are interested in the following natural values: Y (the number of steps for the process to finish), U (number of processors needed in greedy scheme) and Q (number of processors needed in an optimal scheme). We are typically interested in finding upper and lower bounds that hold asymptotically almost surely for such variables (see below for a definition) or their expected values.

3.2 Theoretical bounds for binomial random graphs

3.2.1 Preliminary results and notation

Let $0 \leq p \leq 1$ and let Ω be the family of all graphs on n vertices. To every graph $G \in \Omega$ we assign a probability

$$\mathbb{P}(\{G\}) = p^{|E(G)|}(1-p)^{\binom{n}{2}-|E(G)|}.$$

We denote this probability space by $\mathbb{G}(n,p)$. The space $\mathbb{G}(n,p)$ is often referred to as the **binomial random graph** or **Erdős-Rényi random graph**. Note also that this probability space can informally be viewed as a result of $\binom{n}{2}$ independent coin flips, one for each pair of vertices u, v, where the probability of success (that is, adding an edge uv) is equal to p.

As typical in random graph theory, we shall consider only asymptotic properties of the model as $n \to \infty$ and $T \to \infty$. In particular, when writing inequalities in the proofs we require that they hold asymptotically. Moreover, $p = p(n)$ and $T = T(n)$ may and usually do depend on n. We emphasize that the notations $o(\cdot)$ and $O(\cdot)$ refer to functions of n, not necessarily positive, whose growth is bounded. We use the notations $f \ll g$ for $f = o(g)$ and $f \gg g$ for $g = o(f)$. We also write $f(n) \sim g(n)$ if $f(n)/g(n) \to 1$ as $n \to \infty$ (that is, when $f(n) = (1+o(1))g(n)$).

We say that an event in a probability space holds **asymptotically almost surely** (**a.a.s.**) if its probability tends to one as n goes to infinity. For mode details see, for example, the two classic books [6, 23] or more recent monograph [17].

In the proofs we will use a well-known bound on tail probabilities.

Theorem 3.4 (**Chernoff Bound** (see [23], Theorem 2.1))**.** *Let $X \in \mathrm{Bin}(n,p)$ be distributed as a binomial random variable with n trials and success probability p, so $\mathbb{E}[X] = \mu = pn$. If $0 < \delta < 1$, then*

$$\mathbb{P}[X < (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2}\right). \tag{1}$$

If $\delta > 0$, then

$$\mathbb{P}[X > (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right). \tag{2}$$

In addition, all of the above bounds hold for the general case in which $X = \sum_{i=1}^{n} X_i$ and X_i is the Bernoulli random variable with parameter p_i with (possibly) different p_i's.

Finally, let us make an observation that will simplify the notation. Since one can generate random permutation A before generating random graph $G \in \mathbb{G}(n,p)$, without loss of generality, we may assume that A is the identity permutation; that is, $A(i) = i$ for all i.

3.2.2 Results

Let $Y_{\mathbb{G}}(n,p)$ and $U_{\mathbb{G}}(n,p)$ be the corresponding values of Y and U for $\mathbb{G}(n,p)$, respectively. Below we collect our results for $Y_{\mathbb{G}}(n,p)$ into a single theorem and then we deal with upper and lower bounds independently (that, in fact, are often stronger than what is claimed here). Part (a) follows from Theorems 3.10 and 3.11. Part (b) follows from Theorems 3.10 and 3.12.

Theorem 3.5. *Let $\epsilon > 0$ be any constant. The following hold a.a.s.*
(a) If $pn \ge \epsilon \log n$, then $Y_{\mathbb{G}}(n,p) = \Theta(pn)$;
(b) If $pn = o(\log n)$ and $pn \ge \epsilon$, then $Y_{\mathbb{G}}(n,p) = \Theta(\beta pn)$, where

$$\beta = \beta(n,p) = \frac{\log n/(pn)}{\log(\log n/(pn))}.$$

In particular $Y_{\mathbb{G}}(n,p) = \Theta(\log n/\log\log n)$ for $pn = \Theta(1)$.

(Note our convention that $\log n/a = (\log n)/a$, here and throughout the rest of the paper.)

The results for $U_{\mathbb{G}}(n,p)$ are collected below. In fact, more can be said for $pn > 0.49 \log n$—see Theorem 3.11.

Theorem 3.6. *For any $0 < p = p(n) \le 1$,*

$$\max\Big\{\mathbb{E}\big[|X_k|\big] : k \in \mathbb{N}\Big\} \le \mathbb{E}\big[|X_0|\big] \le p^{-1}.$$

Moreover, the following holds a.a.s.
(a) If $pn \gg 1$ and $pn = o(n/\log n)$, then $U_{\mathbb{G}}(n,p) \sim |X_0| \sim p^{-1}$.
(b) If $pn \gg 1$ and $pn \le 0.49 \log n$, then $U_{\mathbb{G}}(n,p) = |X_0| \sim p^{-1}$.
(c) If $pn = c$ for some $c \in \mathbb{R}_+$, then $U_{\mathbb{G}}(n,p) = |X_0| \sim n(1-e^{-c})/c$.

Knowing some properties of $Y_{\mathbb{G}}(n,p)$ and $U_{\mathbb{G}}(n,p)$, we can reason about $Q_{\mathbb{Q}}(n,p)$ using the following natural bounds that hold for any graph generating process:

Observation 3.7. *We have that $n/Y \le Q \le U$.*

Therefore, using this observation the following corollary follows immediately from Theorems 3.5 and 3.6:

Corollary 3.8. *Let $\epsilon > 0$ be any constant. The following hold a.a.s.*
(a) if $pn > \epsilon \log n$ and $pn = o(n/\log n)$, then

$$\Theta(p^{-1}) = n/Y_{\mathbb{G}}(n,p) \le Q_{\mathbb{G}}(n,p) \le U_{\mathbb{G}}(n,p) \sim p^{-1};$$

(b) if $pn \gg 1$ and $pn = o(\log n)$, then

$$\Theta(\beta^{-1}p^{-1}) = n/Y_{\mathbb{G}}(n,p) \le Q_{\mathbb{G}}(n,p) \le U_{\mathbb{G}}(n,p) \sim p^{-1},$$

where

$$\beta = \beta(n,p) = \frac{\log n/(pn)}{\log(\log n/(pn))}.$$

(c) if $pn \to c$ for some $c \in \mathbb{R}_+$, then

$$\Theta(n \log\log n/\log n) = n/Y_{\mathbb{G}}(n,p) \le Q_{\mathbb{G}}(n,p) \le U_{\mathbb{G}}(n,p) \sim n(1-e^{-c})/c.$$

In Section 3.3 we provide simulation analysis of $Q_{\mathbb{G}}(n,p)$ and find that it actually tends (very fast) to the lower bound established in Corollary 3.8 for sparse graphs (for such graphs there is the widest gap between lower and upper bound on $Q_{\mathbb{G}}(n,p)$). A rigorous proof of this fact is left as an open problem and we plan to investigate it in a future work.

3.2.3 Upper bound

Let us start with an upper bound. First we present a weaker bound but one that has some practical implications. In particular, from the proof it follows that one can partition a permutation A into blocks of order pn and deal with each block one by one, without loosing too much on the performance.

Theorem 3.9. *Let $\epsilon > 0$ be any constant. A.a.s. $Y_{\mathbb{G}}(n,p) = O(pn \log n)$, provided that $pn \ge \epsilon$ and $p = o(1)$.*

Proof. Let us partition a permutation A into $\lceil 2pn \rceil$ blocks, each of length at most $\lceil 1/2p \rceil$. (Recall that A is assumed to be the identity permutation.) We will use the following sub-optimal strategy (see the discussion in Observation 3.2) that will yield sets $\hat{X}_k$ in the corresponding directed graphs $\hat{S}_k$. During the first phase, we keep removing all vertices of in-degree zero, but only those that belong to the first block. Once all vertices from the first block are deleted, we move to the second phase during which only vertices from the second block are considered. We continue similarly with the remaining blocks until all vertices are removed and the process ends. Clearly, this is a suboptimal strategy but will be relatively easy to analyze. Our goal is to show that a.a.s. each phase takes $O(\log n)$ rounds to be finished. This immediately implies the desired upper bound for Y.

Note that our sub-optimal process on $\mathbb{G}(n,p)$ reduces to analyzing $\lceil 2pn \rceil$ original processes on independent copies of $\mathbb{G}(\lceil 1/2p \rceil, p)$. Since the expected degree in $\mathbb{G}(\lceil 1/2p \rceil, p)$ is asymptotic to $1/2 < 1$ it is well known that a.a.s. the largest component has size $O(\log n)$. For our purpose, however, we need to show that a.a.s. it is true for *all* copies. Since the argument is easy and standard we omit some details.

Take any vertex v and consider the **breadth-first-search** process starting from v. Put v into the **queue** Q (first-in first-out list), call v **saturated**, and then do the following as long as Q is not empty: remove w from Q, expose all edges from w to non-saturated vertices, put all new neighbours of w into Q and call them saturated. Note that Z_i, random variable counting the number of vertices added into Q at ith step of this process has binomial distribution $\mathrm{Bin}(\lceil 1/2p\rceil - m, p)$, where m is the number of saturated vertices at that step of the process (m is affected by the process but clearly it is always non-negative). Hence, it can be stochastically upper bounded by $\bar{Z}_i$, an independent copy of $\mathrm{Bin}(\lceil 1/2p\rceil, p)$. As a result, the BFS process resembles very much the branching process with parameter $1/2$ that is known to die out with probability 1.

Note that the probability that v belongs to a component of size at least $k = k(n)$ is bounded from above by the probability that $\sum_{i=1}^{k} Z_i \geq k-1$. Note also that $\bar{Z} := \sum_{i=1}^{k} \bar{Z}_i \sim \mathrm{Bin}(k\lceil 1/2p\rceil, p)$ and so $\mathbb{E}[\bar{Z}] \geq k/2$. Hence, using the union bound over all vertices, the probability that some copy of $\mathbb{G}(n,p)$ contains a component of size k is at most

$$\begin{aligned} n\ \mathbb{P}\left(\sum_{i=1}^{k} Z_i \geq k-1\right) &\leq n\ \mathbb{P}\left(\sum_{i=1}^{k} \bar{Z}_i \geq k-1\right) \leq n\ \mathbb{P}\Big(\bar{Z} \geq 1.9\ \mathbb{E}[\bar{Z}]\Big) \\ &\leq n\ \exp\left(-\frac{0.9^2\ \mathbb{E}[\bar{Z}]}{2+0.9}\right) \leq n\ \exp\Big(-0.27(k/2)\Big) \\ &= o(1), \end{aligned}$$

provided that $k = \lceil 8 \log n\rceil$. The proof is finished. □

Here is another upper bound that determines an order of Y. However, it provides less insight into the problem.

Before we state the result, we need to define one important constant. For $0 < p = p(n) < 1$, let $c = c(p)$ be a positive constant that is the solution of the following equation:

$$xp\log(x) + (1-xp)\log(1-xp) = 0. \tag{3}$$

It is easy to see that $c \to e$ if $p \to 0$ and $c \to 1$ if $p \to 1$. Moreover, cp is growing with p but $cp < 1$, provided p is bounded away from 1, see Figure 1.

Theorem 3.10. *The following hold a.a.s.*

(a) $Y_{\mathbb{G}}(n,p) \leq (1+o(1))cpn$, *provided that* $p = \Theta(1)$.

(b) $Y_{\mathbb{G}}(n,p) \leq (1+o(1))epn$, *provided that* $p = o(1)$ *and* $pn \gg \log n$.

(c) $Y_{\mathbb{G}}(n,p) \leq xpn = O(pn)$, *provided that* $pn = \Theta(\log n)$, *where* x *is a constant such that*

$$x\log(x) - x = \log n/(pn).$$

(d) $Y_{\mathbb{G}}(n,p) \leq (1+o(1))\beta pn$, *provided that* $pn = o(\log n)$ *and* $pn = \Omega(1)$, *where*

$$\beta = \beta(n,p) = \frac{\log n/(pn)}{\log(\log n/(pn))}.$$

Note that $\beta \to \infty$. *In particular,* $\beta = \Theta(\log n/\log\log n)$ *for* $pn = \Theta(1)$.

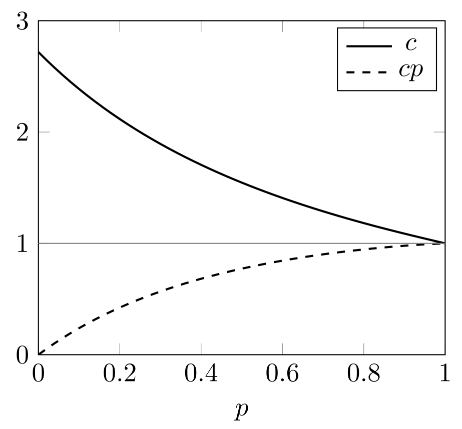

Figure 1: Plot of c and cp against p.

Proof. A path $(v_0, v_1, \ldots, v_k)$ of length k is called ordered if $1 \leq v_0 < v_1 < \ldots < v_k \leq n$. (Recall that A is assumed to be the identity permutation.) It is straightforward to see that $Y_{\mathbb{G}}(n,p) = k$ if and only if the length of a longest path is equal to k. Hence, in order to show an upper bound of k for Y that holds a.a.s., it is enough to show that the expected number of ordered paths of length k is $o(1)$. The first moment method will immediately imply that a.a.s. there is no such path and so a.a.s. the desired bound will hold.

Fix a positive integer k. For a given sequence of vertices $v_0, v_1, \ldots, v_k$ such that $0 \leq v_0 < v_1 < \ldots < v_k \leq n$, let $Z(v_0, v_1, \ldots, v_k)$ be the indicator random variable that is equal to one if $(v_0, v_1, \ldots, v_k)$ yields an ordered path; and it is equal to zero otherwise. We are interested in the following random variable

$$Z = \sum_{0 \leq v_0 < v_1 < \ldots < v_k \leq n} Z(v_0, v_1, \ldots, v_k)$$

counting the number of ordered paths of length k.

Let ϵ' be any positive constant (arbitrarily small) and assume that $p \leq 1-\epsilon'$. The result for $p = 1-o(1)$ will hold by monotonicity of $Y_{\mathbb{G}}(n,p)$. Let $\epsilon = \epsilon(n)$ be a function tending to zero slowly enough (it will be determined soon, depending on the case we will consider).

Let $k = k(n) := (c+\epsilon)pn$, where c is defined in (3). Using Stirling's formula we get

$$\begin{aligned}
\mathbb{E}[Z] &= \binom{n}{k+1} p^k = \frac{n-k}{k+1} \cdot \frac{n! p^k}{k!(n-k)!} \\
&= \Theta(n/k) \frac{\sqrt{2\pi n}(n/e)^n p^k}{\sqrt{2\pi k}(k/e)^k \sqrt{2\pi(n-k)}((n-k)/e)^{n-k}} \\
&= \Theta\left(n/k^{3/2}\right) \frac{n^n p^{(c+\epsilon)pn}}{((c+\epsilon)pn)^{(c+\epsilon)pn}((1-(c+\epsilon)p)n)^{(1-(c+\epsilon)p)n}} \\
&= \Theta\left(n/k^{3/2}\right) (c+\epsilon)^{-(c+\epsilon)pn}(1-(c+\epsilon)p)^{-(1-(c+\epsilon)p)n} \\
&= \Theta\left(n/k^{3/2}\right) \exp\Big(-n\ f(p, c+\epsilon)\Big), \qquad (4)
\end{aligned}$$

where

$$f(p,x) = xp\log(x) + (1-xp)\log(1-xp).$$

Note that the derivative of f with respect to c satisfies

$$f'(p,c) = p\log(x) - p\log(1-cp) = \frac{f(p,c) - \log(1-px)}{x} = \frac{-\log(1-px)}{x},$$

as it follows from the definition of $c = c(p)$ (see (3)) that $f(p,c) = 0$.

For $p = o(1)$, we get $f'(p,c) \sim p$ and so

$$f(p, c+\epsilon) = f(p,c) + (1+o(1))f'(p,c)\epsilon \sim p\epsilon.$$

It follows that

$$\mathbb{E}[Z] = \Theta\left(n/k^{3/2}\right) \exp\Big(-pn\epsilon\Big) = o(1),$$

provided that $pn \geq \omega \log n$ for some $\omega = \omega(n)$ tending to infinity as $n \to \infty$ (by taking, say, $\epsilon = 1/\sqrt{\omega} = o(1)$). Part (b) follows.

For $p = \Theta(1)$, we get $f'(p,c) = \Theta(1)$. It follows that

$$\mathbb{E}[Z] = \Theta\left(n/k^{3/2}\right) \exp\Big(-n\Theta(\epsilon)\Big) = o(1),$$

provided that, say, $\epsilon = 1/\sqrt{n} = o(1)$. Part (a) follows.

On the other hand, for $pn = O(\log n)$ but $pn = \Omega(1)$, and $x > e$ but $xp = o(1)$, we have

$$f(p,x) = xp\log(x) - (1-xp)\left(xp + (1+o(1))\frac{(xp)^2}{2}\right) > xp\log(x) - xp.$$

Now, one can take $x > e$ to be large enough such that $(x\log(x) - x)pn = \log n$, and $k = k(n) := xpn \to \infty$ to get

$$\mathbb{E}[Z] = \Theta\left(n/k^{3/2}\right) \exp\Big(-nf(p,x)\Big) = o(n)\exp(-\log n) = o(1).$$

If $pn = \Theta(\log n)$, then x is large enough constant. If $pn = o(\log n)$, then $x \to \infty$ and so $x\log(x)pn \sim \log n$; that is $x \sim \beta$. Part (c) and (d) follow and so the proof of claimed upper bounds is finished. $\square$

3.2.4 Lower bound

Now, let us move to a lower bound and start with the dense case. The proof of part (a) can be easily adapted to the sparse case. However, we do not do it as for $pn = o(\log n)$ this result does not match the upper bound anyway. We will treat the sparse case independently later.

Theorem 3.11. *For any $0 < p = p(n) \le 1$,*

$$\max\Big\{\mathbb{E}\big[|X_k|\big] : k \in \mathbb{N}\Big\} \le \mathbb{E}\big[|X_0|\big] \le p^{-1}.$$

If $pn \gg 1$, then a.a.s.
(a) $Y_{\mathbb{G}}(n,p) \ge (1+o(1))pn$.
In fact, the following stronger properties hold a.a.s.
(b) If $pn \gg 1$ and $p = o(1)$, then almost all $|X_k|$'s are at most

$$u = u(n) := p^{-1} + p^{-2/3} \sim p^{-1} \sim |X_0|$$

and the sum of cardinalities of all larger ones is $o(n)$.
(c) If $pn \gg 1$ and $pn = o(n/\log n)$, then $U_{\mathbb{G}}(n,p) \le u \sim |X_0|$.
(d) If $pn \gg 1$ and $pn \le 0.49 \log n$, then $U_{\mathbb{G}}(n,p) = |X_0| \sim p^{-1}$.
(e) If $pn = c$ for some $c \in \mathbb{R}$, then $U_{\mathbb{G}}(n,p) = |X_0| \sim n(1-e^{-c})/c$.

Proof. The distribution of X_0 is easy to determine. Vertex i belongs to X_0 if and only if it has no neighbour (in graph G) with a smaller label; that is, there is no edge from i in G to any of $1, 2, \ldots, i-1$. (Recall that A is assumed to be the identity permutation.) Hence, the probability of this event is $p_i = (1-p)^{i-1}$. It follows that $|X_0|$ is distributed as $\mathcal{X}$—the sum of the Bernoulli random variables with parameters p_i $(i = 1, \ldots, n)$. We get

$$\mathbb{E}\Big[|X_0|\Big] = \mathbb{E}\big[\mathcal{X}\big] = \sum_{i=1}^{n} (1-p)^{i-1} = \frac{1-(1-p)^n}{p} \sim p^{-1}, \tag{5}$$

since it is assumed that $pn \gg 1$. (Note that if $pn = c$ for some $c \in \mathbb{R}$, then $\mathbb{E}\big[|X_0|\big] \sim n(1-e^{-c})/c$.) Moreover, $\mathbb{E}\big[|X_0|\big] \le p^{-1}$. If, additionally, $p = o(1)$, then $\mathbb{E}\big[|X_0|\big] \gg 1$ and Chernoff bound easily implies that a.a.s. $|X_0| \sim p^{-1}$.

Clearly, the distribution of $|X_1|$ is affected by X_0 but the probability that ith vertex in the sub-permutation obtained after removing vertices from X_0 has no neighbour with a smaller label is at most p_i. (Note that we condition on the fact that this vertex was not in X_0; that is, it used to have at least one neighbour with a smaller label one step before.) Hence, $|X_1|$ can be stochastically upper bounded by (an independent copy of) the random variable distributed as $\mathcal{X}$.

Formally, one can do the following coupling. Vertex v is called **good** if it belongs to X_0; the only information that is exposed to determine if v is good is whether it has a neighbour with a smaller label. Vertex that is not good exposes more information about the graph and it is called **bad** if it has a neighbour with a smaller label in X_0; note that edges to vertices with a smaller label that are not in X_0 are not exposed. Otherwise, v is called **very bad**. Now, ith vertex in the sub-permutation we consider belongs to X_1 with probability p_i, provided that it is bad, and 0 if it is very bad. The desired coupling is easy to establish. The only purpose of partitioning the set of vertices into good, bad, and very bad is to formally define the coupling and we will not use it later in the proof.

In fact, it can be upper bounded by the sum of $n-|X_0|$ random variables but we are happy with the proposed weaker coupling for parts (a)-(c). We proceed similarly with $|X_k|$ for any $k \in \mathbb{N}$. In particular, we will show that for any k, $\mathbb{E}\big[|X_k|\big] \le \mathbb{E}\big[|X_0|\big] \le p^{-1}$.

Part (a) is straightforward. The established coupling implies that $\sum_{k=0}^{\ell-1} |X_k|$ is stochastically upper bounded by the sum of ℓ independent copies of $\mathcal{X}$, and so itself it is a sum of the Bernoulli random variables with expectation $\ell\,\mathbb{E}\big[\mathcal{X}\big] \le \ell p^{-1}$. Taking $\ell = \lfloor np - (np)^{2/3} \rfloor \sim np$, Chernoff bound implies that

$$\mathbb{P}\left(\sum_{k=0}^{\ell-1} |X_k| \ge n \right) \le \exp\left(-\Theta\Big((np)^{1/3} \Big) \right) = o(1).$$

So a.a.s. at least ℓ rounds are needed to finish the process and part (a) follows.

Part (c) is also easy. Recall that in this case it is assumed that $p^{-1} \ge \omega \log n$ for some $\omega = \omega(n) \to \infty$ as $n \to \infty$. Our goal is to show that a.a.s. during all rounds all $|X_k|$'s are at most $p^{-1} + p^{-2/3} \sim p^{-1}$. Note that the process must end after n rounds (in fact, we know that it will finish earlier a.a.s. but it gives us no advantage here). It follows from Chernoff bound, combined with the coupling mentioned earlier, that for any k

$$\mathbb{P}\Big(|X_k| > p^{-1}(1+\delta) \Big) \le \mathbb{P}\Big(\mathcal{X} > p^{-1}(1+\delta) \Big) \le \exp\left(-\frac{\delta^2 p^{-1}}{3} \right) = o(n^{-1}),$$

by taking $\delta = 2/\sqrt{\omega} = o(1)$. Hence, a.a.s. the desired property holds and part (c) follows.

Part (b) is the most sophisticated. We may assume that $p \ge \epsilon/\log n$ for some $\epsilon > 0$ as, otherwise, part (c) provides a stronger statement. This time our goal is to show that a.a.s. during the first $3pn$ rounds almost all $|X_k|$'s are at most $p^{-1} + p^{-2/3} \sim p^{-1}$ and the sum of cardinalities of all larger ones is $o(n)$ and so is negligible compared to the total number of vertices, n; that is, almost all vertices belong to small $|X_k|$'s. (Note that it follows from Theorem 3.10 that a.a.s. the process will finish after $(1+o(1))epn$ rounds.)

Let us restrict ourselves to the sequence $(|X_k|)_{k=0}^{3pn-1}$ of $3pn$ random variables. It follows from Chernoff bound, combined with the coupling mentioned earlier, that for any k

$$\mathbb{P}\Big(|X_k| > p^{-1} + p^{-2/3} \Big) \le \mathbb{P}\Big(\mathcal{X} > p^{-1} + p^{-2/3} \Big) \le \exp\left(-\frac{p^{-1/3}}{3} \right) =: q_0 = o(1).$$

Similarly, for any k and any $s \in \mathbb{N}$

$$\begin{aligned}\mathbb{P}\Big(|X_k| > 2^{s+1}p^{-1}\Big) &\leq \mathbb{P}\Big(\mathcal{X} > 2^{s+1}p^{-1}\Big) \leq \mathbb{P}\Big(\mathcal{X} > (1+2^s)p^{-1}\Big) \\ &\leq \exp\left(-\frac{2^{2s}p^{-1}}{2+2^s}\right) \leq \exp\left(-2^{s-1}p^{-1}\right) =: q_s \leq q_{s-1}/4.\end{aligned}$$

The last inequality has a lot of room to spare.

Now, the number of values of k for which $|X_k|$ is more than $p^{-1} + p^{-2/3}$ is stochastically upper bounded by $\mathrm{Bin}(3pn, q_0)$. After applying Chernoff bound one more time we get that a.a.s. for almost all values of k (that is, for all but $o(1)$ fraction of them) we have $|X_k| \leq p^{-1} + p^{-2/3}$. Let $s_{\max}$ be the smallest natural number s such that $2^s p^{-1} \geq \log^2 n$. As before, for any $s \in \mathbb{N}$ such that $1 \leq s \leq s_{\max}$, the number of random variables that are more than $2^{s+1}p^{-1}$ is stochastically upper bounded by $\mathrm{Bin}(3pn, q_s)$. Since $3pn \cdot q_{s_{\max}} = o(1)$, it follows from Markov's inequality that a.a.s. no random variable is more than $2^{s_{\max}+1}p^{-1}$. Finally, for any $1 \leq s \leq s_{\max}$, using Chernoff bound for the last time we get that the number of variables that are more than $2^{s+1}p^{-1}$ is larger than $2 \cdot 3pn \cdot q_s$ with probability $\exp(-\Omega(pnq_s))$. Since $q_s \leq q_{s-1}/4$, a.a.s. corresponding bounds hold for *all* values of s in the range. The conclusion is that a.a.s. the sum of all variables that are more than $p^{-1} + p^{-2/3} \sim p^{-1}$ is of order at most

$$p^{-1} \cdot o(pn) + \sum_{s=1}^{s=s_{\max}} (2^s p^{-1}) \cdot (pnq_s) = o(n) + \sum_{s=1}^{s=s_{\max}} 2^s n q_1 / 4^{s-1} = o(n)$$

and so, as promised, is negligible. Part (b) follows.

For part (d), assume that $pn \leq 0.49 \log n$ and $pn \gg 1$. This time we have to be slightly more careful with estimating error terms. We have (see (5) for a more coarse version)

$$\begin{aligned}\mathbb{E}\Big[|X_0|\Big] = \sum_{i=1}^{n}(1-p)^{i-1} &= \Big(1 - \exp\Big(-pn + O(p^2 n)\Big)\Big)\, p^{-1} \\ &= \Big(1 - (1+o(1))\exp\big(-pn\big)\Big)p^{-1} \sim p^{-1},\end{aligned}$$

and so by Chernoff bound a.a.s.

$$|X_0| \geq \mathbb{E}\Big[|X_0|\Big] - \log n/\sqrt{p} \geq \mathbb{E}\Big[|X_0|\Big] - \sqrt{n}\log^2 n.$$

(Note that the last inequality holds not only for $pn \gg 1$ but also when $pn = c$ for some $c \in \mathbb{R}$.) On the other hand, for each $k \in \mathbb{N}$ we use the fact and the coupling mentioned earlier that $|X_k|$ can be upper bounded by the sum of $n - |X_0| = n - (1+o(1))/p$ random variables and so its expectation is substantially smaller than the one for $|X_0|$. This will be enough to make sure that not only the corresponding expectations are far apart but with the desired probability $|X_k| < |X_0|$. Indeed we get that for any $k \in \mathbb{N}$

$$\begin{aligned}
\mathbb{E}\Big[|X_k|\Big] &= \sum_{i=1}^{n-|X_0|} (1-p)^{i-1} \\
&\le \Big(1 - \exp\Big(-p\Big(n - (1+o(1))/p\Big) + O(p^2 n)\Big)\Big) p^{-1} \\
&= \Big(1 - (e+o(1))\exp\big(-pn\big)\Big)p^{-1} \\
&\le \mathbb{E}\Big[|X_0|\Big] - \exp\big(-pn\big)p^{-1} \\
&\le \mathbb{E}\Big[|X_0|\Big] - \exp\big(-0.49\log n\big)n/\log n \\
&= \mathbb{E}\Big[|X_0|\Big] - n^{0.51}/\log n.
\end{aligned}$$

It follows from Chernoff bound that for any $k \in \mathbb{N}$, with probability $1 - o(n^{-1})$,

$$\begin{aligned}
|X_k| &\le \Big(\mathbb{E}\Big[|X_0|\Big] - n^{0.51}/\log n\Big) + \log n/\sqrt{p} \\
&\le \Big(\mathbb{E}\Big[|X_0|\Big] - n^{0.51}/\log n\Big) + \sqrt{n}\log^2 n \\
&< \mathbb{E}\Big[|X_0|\Big] - \sqrt{n}\log^2 n \quad \le \quad |X_0|.
\end{aligned}$$

Therefore, by the union bound over at most n values of k, a.a.s.

$$\max\Big\{|X_k| : k \in \mathbb{N}\Big\} < |X_0|.$$

In part (e), the only difference is that $\mathbb{E}\big[|X_0|\big] \sim n(1-e^{-c})/c$ and so the proof is finished. □

As already mentioned, it seems that Theorem 3.10 yields an asymptotic behaviour of Y. However, since our aim is to determine the order of Y, we leave it as an open problem and only concentrate on a weaker result. Theorem 3.11 provides a matching lower bound if $pn \ge \epsilon \log n$ for some $\epsilon > 0$ so it remains to concentrate on the case $pn = o(\log n)$.

Theorem 3.12. *Let $\epsilon > 0$ be any constant. A.a.s. $Y_{\mathbb{G}}(n,p) \ge (1+o(1))\beta pn$, provided that $pn = o(\log n)$ and $pn \ge \epsilon$, where*

$$\beta = \beta(n,p) = \frac{\log n/(pn)}{\log(\log n/(pn))}.$$

In particular, $\beta \to \infty$ and $\beta = \Theta(\log n/\log\log n)$ for $pn = \Theta(1)$.

Proof. We will use the same notation as in the proof of Theorem 3.10. By taking $x = x(n) \to \infty$ such that

$$nf(p,x) \le x\log(x)pn = \log n - \frac{2\log n}{\log\beta} - 2\log\log n \sim \log n$$

we get that

$$x \sim \frac{\log n/(pn)}{\log(\log n/(pn))} = \beta.$$

It follows that using (4) when $k = xpn$ we get

$$\begin{aligned} E[Z] &= \Theta\Big(n/k^{3/2}\Big) \exp\Big(-n\ f(p,x)\Big) \\ &= \Omega\Big(n/\log^{3/2} n\Big) \exp\Big(-\log n + 2\log\log n\Big) \to \infty \end{aligned}$$

as $n \to \infty$. We will use the second moment method to show that a.a.s. $Z \geq 1$.

Let us consider two directed paths of length $k = xpn$, $(v_0, v_1, \dots, v_k)$ and $(w_0, w_1, \dots, w_k)$, and associated with them indicator random variables $Z_v = Z(v_0, v_1, \dots, v_k)$ and $Z_w = Z(w_0, w_1, \dots, w_k)$. Clearly, if they share at most one vertex, then they are edge disjoint and so $\mathbb{C}ov(Z_v, Z_w) = 0$. On the other hand, if they share s vertices (for some $2 \leq s \leq k$), then they share at most $s-1$ edges and so

$$\mathbb{C}ov(Z_v, Z_w) \leq \mathbb{E}[Z_v Z_w] \leq p^{2k-(s-1)}.$$

Denoting the falling factorial by $(q)_r = q(q-1)\cdots(q-r+1)$, the number of pairs of paths of this type is

$$\begin{aligned} \binom{n}{2(k+1)-s}\binom{2(k+1)-s}{s}\binom{2(k+1)-2s}{(k+1)-s} &= \frac{(n)_{2(k+1)-s}}{s!\ (k+1-s)!^2} \\ &= \left(\frac{(n)_{k+1}}{(k+1)!}\right)^2 \cdot \frac{1}{s!} \cdot \left(\frac{(k+1)!}{(k+1-s)!}\right)^2 \cdot \frac{(n)_{2(k+1)-s}}{(n)_{k+1}^2} \\ &= \binom{n}{k+1}^2 \frac{(k+1)_s^2}{s!} \cdot \frac{(n-k-1)_{k+1-s}}{(n)_{k+1}} \\ &\leq \binom{n}{k+1}^2 \frac{(k+1)_s^2}{s!(n-k)^s} \leq (1+o(1))\binom{n}{k+1}^2 \left(\frac{ek^2}{sn}\right)^s, \end{aligned}$$

since $s \leq k = O(\log n)$, $k \to \infty$, and $s! \geq (s/e)^s$. Indeed, there are $\binom{n}{2(k+1)-s}$ ways to select vertices for the two paths, $\binom{2(k+1)-s}{s}$ ways to select vertices that are shares by both paths, and $\binom{2(k+1)-2s}{(k+1)-s}$ ways to assign the remaining vertices to the paths. Hence, the variance can be estimates as follows

$$\begin{aligned} \mathbb{V}ar[Z] &= \sum_{v,w} \mathbb{C}ov(Z_v, Z_w) \\ &\leq \mathbb{E}[Z] + (1+o(1)) \sum_{s=2}^{k} \binom{n}{k+1}^2 \left(\frac{ek^2}{sn}\right)^s p^{2k-s+1} \\ &= \mathbb{E}[Z] + (1+o(1))\mathbb{E}[Z]^2 \cdot p \cdot \sum_{s=2}^{k} \left(\frac{ek^2}{spn}\right)^s. \end{aligned}$$

Clearly, $\frac{ek^2}{spn} \geq \frac{ek}{pn} = ex \to \infty$. Moreover, observe that $h(s) := \left(\frac{ek^2}{spn}\right)^s$ is increasing for $s \leq k$. To see this recall that $k/(np) = x > 1$ and so

$$\frac{h(s)}{h(s-1)} = ex\frac{k(s-1)^{s-1}}{s^s} > \frac{e}{\left(1+\frac{1}{s-1}\right)^{s-1}} > 1.$$

It follows that

$$\begin{aligned}
\mathbb{V}ar[Z] &\leq \mathbb{E}[Z] + \mathbb{E}[Z]^2 \cdot p \cdot O(k) \cdot \left(\frac{ek}{pn}\right)^k \\
&= \mathbb{E}[Z] + \mathbb{E}[Z]^2 \cdot o\left(\frac{\log^2 n}{n}\right) \cdot (ex)^{xpn} \\
&= \mathbb{E}[Z] + \mathbb{E}[Z]^2 \cdot o\left(\frac{\log^2 n}{n}\right) \cdot \exp\Big(x\log(x)pn + xpn\Big) \\
&= \mathbb{E}[Z] + \mathbb{E}[Z]^2 \cdot o\left(\frac{\log^2 n}{n}\right) \cdot \exp\left(\log n - \frac{2\log n}{\log\beta} - 2\log\log n + xpn\right) \\
&= \mathbb{E}[Z] + o\left(\mathbb{E}[Z]^2\right) \cdot \exp\left(-\frac{2\log n}{\log\beta} + xpn\right) = o\left(\mathbb{E}[Z]^2\right),
\end{aligned}$$

since $xpn \leq \log n/\log x \leq 2\log n/\log\beta$. As promised, the conclusion follows from the second moment method. $\square$

3.2.5 Distribution of $|X_k|$'s

Let us finish this section with investigating sizes of X_k's for small values of k. We already showed that a.a.s. $|X_0| \sim p^{-1}$ and all remaining $|X_k|$'s are stochastically upper bounded by $|X_0|$. This implies that for any constant C (arbitrarily large), a.a.s. for all $0 \leq k \leq C$ we have $|X_k| \leq (1+o(1))p^{-1}$. In fact, Theorem 3.11(c) shows that this is true for all k, provided $pn \gg 1$ and $p \ll 1/\log n$.

Here we will investigate an asymptotic behaviour of $|X_k|$'s for $0 \leq k \leq C$. In order to show the result we will use the differential equation method [43]. The general setting that is used in this method is a sequence of random processes indexed by n (which in our case is the number of vertices in X_k). The aim is to find asymptotic properties of the random process and the conclusion we aim for is that variables defined are well concentrated, which informally means that a.a.s. they are very close to certain deterministic functions. These functions arise as the solution to a system of ordinary first-order differential equations. One of the important features of this approach is that the computation of the approximate behavior of processes is clearly separated from the proof that the approximation is correct.

Before we state the result, we need to define (recursively) a sequence of functions. Let $y_0 : \mathbb{R} \to \mathbb{R}$ be the particular solution to the following differential equation with the initial condition:

$$y_0'(x) = \exp(-x), \quad y_0(0) = 0.$$

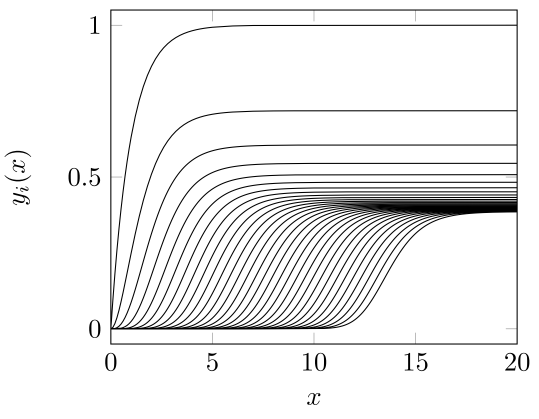

Figure 2: Plot of $y_k(x)$ against x for $k = 0, 1, \dots, 29$.

Now, we recursively define $y_k : \mathbb{R} \to \mathbb{R}$ for any $k \in \mathbb{N}$:

$$y_k'(x) = \exp\left(-x + \sum_{\ell=0}^{k-1} y_\ell(x)\right) - \exp\left(-x + \sum_{\ell=0}^{k-2} y_\ell(x)\right), \quad y_k(0) = 0.$$

It is easy to find y_0 and y_1 explicitly:

$$\begin{aligned} y_0(x) &= 1 - \exp(-x), \\ y_1(x) &= \exp(-\exp(-x) + 1) + \exp(-x) - 2. \end{aligned}$$

Functions with larger indexes can be approximated numerically. The solutions we present below to the corresponding ODE's are given using Verner's "Most Efficient" 8/7 Runge-Kutta method [35], with absolute and relative accuracy set to 10^{-12}. Values of $y_k(x)$ for $k < 30$ are presented on Figure 2.

Finally, for any $k \in \mathbb{N} \cup \{0\}$ we define $b_k = \lim_{x\to\infty} y_k(x)$. It follows that $b_0 = 1$, $b_1 = e - 2$; other values for $k \le 29$ are approximated in Table 1.

Theorem 3.13. *The following hold a.a.s.*

(a) For any $k \in \mathbb{N} \cup \{0\}$, $|X_k| \sim b_k p^{-1}$, provided that $pn \gg 1$ and $p = o(1)$.

(b) For any $k \in \mathbb{N} \cup \{0\}$, $|X_k| \sim y_k(c) p^{-1}$, provided that $pn \to c \in \mathbb{R}_+$.

Proof. We will investigate vertices one by one, starting with vertex of label 1 and finishing with vertex of label n, and expose all edges from the current vertex to vertices with smaller labels. Based on that, we may classify each vertex at the time it is investigated, and put it into the right set X_k. For simplicity, let $Z_k(t)$ be the size of X_k after dealing with vertices with labels at most t.

Clearly, a vertex joins X_0 if it has no neighbour with a smaller label; it follows that

$$\mathbb{E}\Big[Z_0(t+1) - Z_0(t) \mid Z_0(t)\Big] = (1-p)^t = \exp\Big(-pt + O(p^2 t)\Big).$$

k	b_k	k	b_k	k	b_k
0	1.000000	10	0.426076	20	0.395735
1	0.718282	11	0.420582	21	0.394306
2	0.605452	12	0.415981	22	0.393013
3	0.545083	13	0.412078	23	0.391836
4	0.507813	14	0.408730	24	0.390760
5	0.482700	15	0.405829	25	0.389775
6	0.464738	16	0.403294	26	0.388869
7	0.451320	17	0.401063	27	0.388032
8	0.440956	18	0.399084	28	0.387258
9	0.432737	19	0.397318	29	0.386540

Table 1: Values of b_k that were approximated by $y_k(40)$.

It provides some insight if we define real function $f_0(x)$ to model the behaviour of the scaled random variable $Z_0(x/p)/p^{-1}$. If we presume that the changes in the function correspond to the expected changes of the random variable, we obtain the following differential equation

$$f_0'(x) = \exp(-x)$$

with the initial condition $f_0(0) = 0$. The general solution of this equation can be put in the form $f_0(x) = C - \exp(-x)$ with $C \in \mathbb{R}$, and the particular solution is $f_0(x) = y_0(x)$. This *suggests* that the random variable $Z_0(t)$ should behave similarly to the deterministic function $y_0(tp)/p$. In particular,

$$|X_0| \sim p^{-1} \lim_{x\to\infty} y_0(x) = b_0 p^{-1} = p^{-1}.$$

In order to make this argument precise and rigorous, one can use our prediction and transform $Z_0(t)$ into something close to a martingale. Then generalization of the Hoeffding-Azuma inequality can be used to show a concentration. However, since here we do not aim to control error terms, we simply use the general purpose theorem [43, Theorem 5.1] to deal with all random variables discussed below simultaneously.

Now, we move to investigating X_1. Clearly, a vertex joins X_1 if it has at least one neighbour with a smaller label in X_0 but no neighbours outside of X_0; it follows that

$$\begin{aligned}
\mathbb{E}\Big[Z_1(t+1) - Z_1(t) \mid Z_0(t), Z_1(t)\Big] &= (1-p)^{t-Z_0(t)}\Big(1-(1-p)^{Z_0(t)}\Big)\\
&= (1-p)^{t-Z_0(t)} - (1-p)^t\\
&= \exp\Big(-pt + pZ_0(t) + O(p^2t)\Big) + \exp\Big(-pt + O(p^2t)\Big).\\
&= \exp\Big(-pt + pZ_0(t) + O(p)\Big) + \exp\Big(-pt + O(p)\Big)\\
&\sim \exp\Big(-pt + pZ_0(t)\Big) + \exp\Big(-pt\Big),
\end{aligned}$$

provided that $t = x/p$ for some $x \in \mathbb{R}$. Note that trivially $Z_0(t) \leq t$.

This time we obtain the following system of differential equations

$$\begin{aligned} f_0'(x) &= \exp(-x) \\ f_1'(x) &= \exp(-x + f_0(x)) - \exp(-x) \end{aligned}$$

with the initial conditions $f_i(0) = 0$, $i \in \{0, 1\}$. The particular solution is $f_1(x) = y_1(x)$, and arguing as before we get that a.a.s. $Z_1(t) \sim y_1(tp)/p$. In particular

$$|X_1| \sim p^{-1} \lim_{x \to \infty} y_1(x) = b_1 p^{-1} = (e-2)p^{-1}.$$

Generalizing it to X_k for some $k \in \mathbb{N}$ is straightforward. A vertex joins X_k if it has at least one neighbour with a smaller label in X_{k-1} but no neighbours in $\bigcup_{\ell \geq k} X_\ell$; it follows that

$$\mathbb{E}\Big[Z_k(t+1) - Z_k(t) \mid Z_0(t), \ldots, Z_k(t) \Big] = (1-p)^{t - \sum_{\ell=0}^{k-1} Z_\ell(t)} \Big(1 - (1-p)^{Z_{k-1}(t)} \Big),$$

which yields the following differential equation

$$f_k'(x) = \exp\left(-x + \sum_{\ell=0}^{k-1} f_\ell(x) \right) - \exp\left(-x + \sum_{\ell=0}^{k-2} f_\ell(x) \right),$$

with $f_k(0) = 0$. As before, the conclusion is that a.a.s. $Z_k(t) \sim y_k(tp)/p$. The proof of the theorem is finished. □

3.3 Simulation analysis for binomial random graphs

In this section, we continue analyzing the $\mathbb{G}(n, p)$ model but for small n and using different techniques, simulations. In particular, for finite (and, in fact, relatively small) values of n, we want to better understand the difference between U, the number of processors needed in the greedy scheme, and Q, the expected number of processors needed in the optimal execution scheme (denoted by q when we deal with a concrete instance of graph $\mathbb{G}(n, p)$ and schedule A and so Q is a constant, not a random variable).

We analyze small values of n ranging from 100 to 1,000 (in case of calculation of Q) and to 100,000 (to analyze U only, as it is less computing intensive). The number of simulation replications 128 per design point was chosen so that it is enough to get reasonably stable estimates. We choose values of pn from the set $\{5, 10, 15, 20\}$ as in practical applications the case of sparse graph is most relevant (especially when n is large).

It is known that in general, finding concrete value of q for a concrete instance of graph $\mathbb{G}(n, p)$ and schedule A is a NP-hard problem [38]. Therefore we use simulation to assess Q, as it cannot be expected that it is possible to derive it analytically and have to restrict ourselves to small problem instances ($n \leq 1000$ in the paper). In order to find exact value of q we formulate problem of finding it as mathematical programming tasks. There are two natural approaches to this:

Mixed Integer Programming (MIP) [44] and Constraint Programming (CP) [3]. Both approaches were analyzed in the earlier literature for this class of problems and MIP usually is more popular one, see e.g. scheduling handbooks [15, 36]. Below we provide MIP and CP formulation of the optimization problem and compare their performance for $\mathbb{G}(n, p)$ generated schedules.

3.3.1 Mixed Integer Programming formulation

We have n vertices in our problem numbered from 1 to n. Recall that E_0 denotes the list of edges (i, j) in graph S_0. Recall also that finding a longest path in S_0 is equivalent to determining the required number of epochs, y, to finish the process. Our goal is to find q, the minimum number of processors in order to finish the job in y epochs numbered from 0 to $y - 1$. This can be written as the following MIP, where $x_{i,k} \in \{0, 1\}$ is an indicator that vertex i is collected in epoch k:

$$\begin{gathered}
\min q \\
\text{subject to:} \\
\forall k \in \{0, 1, \ldots, y-1\} : q \geq \sum_{i=1}^{n} x_{i,k} \\
\forall i \in \{1, 2, \ldots, n\} : \sum_{k=0}^{y-1} x_{i,k} = 1 \\
\forall (i, j) \in E_0 : \sum_{k=1}^{y-1} k(x_{j,k} - x_{i,k}) \geq 1 \\
\forall i \in \{1, 2, \ldots, n\}, k \in \{0, 1, \ldots, y-1\} : x_{i,k} \in \{0, 1\} \\
q \in \mathbb{N}
\end{gathered}$$

We solve this problem using JuMP mathematical programming modeling language [16] and CPLEX solver [22]. In order to make the optimization more efficient additional constraints giving precomputed earliest start and latest finish of each job were added (the constraints do not affect the solution but significantly reduced presovle phase of optimization).

3.3.2 Constraint Programming Formulation

We keep notation that the first epoch is 0 for consistency. By a_i denote the epoch in which vertex i is collected. As above y is number of epochs needed. CP formulation of our task is the following (in the formulation we use notation

n	5	10	15	20
100	2.1516	1.7781	1.6031	1.5207
1,000	2.8516	2.4234	2.2182	2.1688
10,000	3.3422	2.8367	2.6641	2.5375
100,000	3.7313	3.1156	2.8818	2.7863

Table 2: Table of estimate $\mathbb{E}(Y)/(pn)$. In columns there are different values of pn. The results are averages of 128 simulations.

$[\ell]$ that is equal to 1 if ℓ is true and 0 otherwise):

$$\begin{gathered}\min q \\ \text{subject to:} \\ \forall k \in \{0, 1, \dots, y-1\} : q \geq \sum_{i=1}^{n} [a_i = k] \\ \forall (i,j) \in E_0 : a_i < a_j \\ \forall i \in \{1, 2, \dots, n\} : a_i \in \{0, 1, \dots, y-1\} \\ q \in \{1, 2, \dots, n\}\end{gathered}$$

The solutions were generated using MiniZinc environment and GECODE solver [31] using standard `cumulative` constraint.

3.3.3 Simulation results

We have first compared MIP and CP timings for the average degree $pn = 10$ and the number of vertices $n \in \{1000, 2000, 3000\}$ in single-threaded mode (interestingly, for this task it was found that running solvers in multi threaded mode increased solution time). We report here results of single runs, but the times were consistent thorough many runs. The MIP times were respectively: 1.33, 8.10 and 47.24 seconds. The corresponding CP timings were: 1.44, 4.75 and 14.77 seconds. We could observe that CP becomes more efficient for larger n where number of variables in MIP formulation grows much faster than in CP (we have $\approx ny$ variables in MIP and $\approx n$ variables in CP). Our main focus, for practical reasons, is for sparse graphs. But for dense graphs CP approach is even more efficient than MIP. For instance for a graph $\mathbb{G}(400, 0.5)$ a sample of runtime of MIP was 13.50 seconds and CP was 0.61 seconds. The reason is that for dense graphs MIP produces even more variables as y is large and there are a lot of constraints in the model. For CP high y is not problematic and high number of constraints do not pose a problem as they actually simplify finding the right domains for decision variables. In short, the conclusion is as follows: for this problem CP solver can be recommended over MIP.

In Figure 3 we have a comparison between the number of processors for the greedy and the optimal scheme, respectively. The results are a bit scattered due to a limited number of simulations, but it is clear that: (1) increase of n improves

n	5	10	15	20
100	0.9914	1.0906	1.2059	1.2938
1,000	0.9941	0.9913	1.0099	1.0052
10,000	0.9941	0.9966	0.9953	1.0029
100,000	0.9930	1.0001	1.0005	0.9994

Table 3: Estimate of $\mathbb{E}(U)/p$. In columns there are different values of pn. The results are averages of 128 simulations.

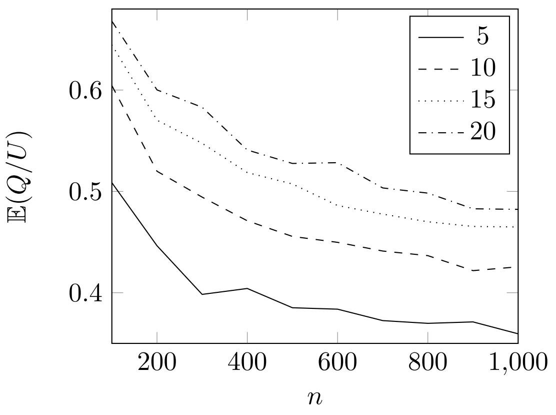

Figure 3: Estimate of $\mathbb{E}(Q/U)$ for $pn \in \{5, 10, 15, 20\}$. Results are based on 128 replications of simulations for the $\mathbb{G}(n,p)$ model.

the benefits from optimal assignment, (2) the higher the average degree, pn, the lower the benefit. In Table 2 we present an estimate of $\mathbb{E}(Y)/(pn)$. We see that they are consistent with the proven theorems: these estimates are greater than 1 and increase with n. In Table 3 we present an estimate of $\mathbb{E}(U)/p$. We see that, again along with the proven theorems, these values converge very fast to p^{-1}, which is asymptotically equal to $\mathbb{E}(|X_0|)$. This is natural to expect; although $\mathbb{E}(U)$ is strictly greater than $\mathbb{E}(|X_0|)$ the concentration theorems show that it should be very close to it.

Additionally, as it was observed earlier trivially $Q \geq \lceil n/Y \rceil$. Therefore, we analyze the ratio $Q/\lceil n/Y \rceil$ which captures how many more processors are needed comparing to the lower bound of U. The results are given in Figure 4. As can be expected, the larger the average degree, pn, the higher the ratio. However, actually the ratio drops and seems to go fast to the lower bound.

4 Bounds for arbitrary graphs

In this section we extend our interest from $\mathbb{G}(n,p)$ graphs, which are the main model of interest in this paper, to arbitrary graphs. We present some preliminary results about $\mathbb{E}[|X_0|]$ for this class of graphs and formulate hypotheses

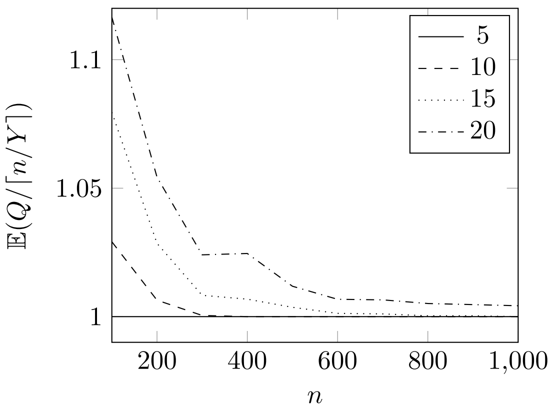

Figure 4: Estimate of $\mathbb{E}(Q/\lceil n/Y\rceil)$ for $pn \in \{5, 10, 15, 20\}$. Results based on 128 replications of simulation for the $\mathbb{G}(n, p)$ model.

about the value of $\mathbb{E}(Y)$ and $\mathbb{E}(U)$.

Theorem 4.1. *Let G be a graph on n vertices with the degree distribution $(d_1, d_2, \ldots, d_n)$. Then,*

$$\mathbb{E}\big[|X_0|\big] = \sum_{i=1}^{n} \frac{1}{d_i + 1} \ .$$

In particular, for any graph on n vertices and m edges we have

$$\frac{n^2}{2m+n} \le \mathbb{E}\big[|X_0|\big] < n - \sqrt{2m} + 2 \ .$$

Moreover, the lower bound is tight for d-regular graphs and asymptotically tight for $m = m(n) \gg n$. Formally, for any $m \gg n$ there exists $G_{\min}$ on n vertices and m edges such that

$$\mathbb{E}\big[|X_0|\big] \sim \frac{n^2}{2m+n} \sim \binom{n}{2}/m.$$

Finally, the upper bound is (almost) tight in the following sense: for any m there exists $G_{\max}$ on n vertices and m edges such that

$$\mathbb{E}\big[|X_0|\big] > n - \sqrt{2m} - 1.$$

Proof. Let A be a random permutation of the vertices of G taken with uniform distribution. For a vertex $v \in V$, let $N^+(v)$ be the number of neighbours of v that follow it in the permutation. The random variable $N^+(v)$ attains each of the values $0, 1, \ldots, \deg(v)$ with probability $1/(\deg(v)+1)$. Indeed, this follows from the fact that the random permutation A induces a uniform, random permutation on the set of $\deg(v)+1$ vertices consisting of v and its neighbours. Therefore, by linearity of expectation,

$$\mathbb{E}\Big[|X_0|\Big] = \sum_{v \in V(G)} \mathbb{P}\Big(N^+(v) = 0\Big) = \sum_{v \in V(G)} \frac{1}{\deg(v)+1} = \sum_{i=1}^{n} \frac{1}{d_i+1},$$

and the claimed equality holds.

It remains to bound the expectation over the family of graphs on n vertices and m edges. Maximizing/minimizing the sum, $\sum_{i=1}^n 1/(d_i+1)$, provided that each $d_i \in \mathbb{R}_+ \cup \{0\}$ and $\sum_{i=1}^n d_i = 2m$ is straightforward. The only two problems are that $d_i \in \mathbb{N} \cup \{0\}$ and that not all sequences are *graphic*, i.e. a sequence of numbers which can be the degree sequence of some (simple) graph. As a result, one needs to be more careful. We consider an upper and a lower bound independently.

Lower bound: After relaxing the condition $d_i \in \mathbb{N} \cup \{0\}$ to $d_i \in \mathbb{R}_+ \cup \{0\}$, the sum is minimized for $d_i = 2m/n$ for all i; that is,

$$\sum_{v \in V(G)} \frac{1}{\deg(v)+1} \geq \frac{n}{2m/n+1} = \frac{n^2}{2m+n}.$$

Clearly, equality holds for d-regular graphs and so this inequality is tight for this class of graphs. As mentioned in the proof of Theorem 3.11, for binomial random graph $\mathbb{G}(n,p)$ we have that a.a.s.

$$\mathbb{E}\Big[|X_0|\Big] \sim p^{-1},$$

provided $pn \gg 1$. It is straightforward to translate results from $\mathbb{G}(n,p)$ to $\mathbb{G}(n,m)$, the model with a given number of edges, $m = p\binom{n}{2}$. Hence, the lower bound is asymptotically tight for $m = m(n) \gg n$.

Upper bound: It is natural to expect that $\mathbb{E}[|X_0|]$ is maximized for a graph $G_{\max}$ that consists of a complete graph with, perhaps, one more non-isolated vertex. Formally, let s be the largest integer such that $\binom{s}{2} \leq m$. We construct graph $G_{\max}$ by starting with a complete graph on s vertices. If $\ell := m - \binom{s}{2} > 0$, then we add another vertex and connect it to ℓ vertices in the complete graph and then we add $n-s-1$ isolated vertices; otherwise (that is, if $\ell = 0$), we simply add $n-s$ isolated vertices.

It is perhaps surprising that this claim does not seem to have a short proof. We found an argument that is relatively simple but quite long and tedious. Fortunately, since we aim for an upper bound that is tight up to an additive constant, it is enough to prove the claim for the case when $\ell = 0$. We start with an arbitrary graph G on n vertices and m edges. If the number of isolated vertices is less than $n-s$, then we show that one can modify G by moving some edges around so that there is one more isolated vertex and, more importantly,

$\mathbb{E}[|X_0|]$ increases. We continue this process until the number of isolated vertices is equal to $n-s$, that is, $G = G_{\max}$.

Suppose that G has $k > s$ non-isolated vertices; that is, there are k non-zero values of d_i. Let v_i $(1 \le i \le n)$ be a vertex of degree d_i. Without loss of generality, we may suppose that $d_i = 0$ for $k < i \le n$, d_k is the smallest non-zero value, and vertex v_k (of degree d_k) is adjacent to vertices $v_1, v_2, \dots, v_{d_k}$ (observe that $d_k < k$).

Note that it is possible to remove all d_k edges adjacent to v_k and put them back (arbitrarily) between vertices from $\{v_i : i < k\}$. Our goal is to show that this operation increases $\mathbb{E}[|X_0|]$, the sum $\sum_{i=1}^n \frac{1}{d_i+1}$. Let (d_i) and $(\hat{d_i})$ be the degree distributions before and after the operation. Clearly,

$$\begin{aligned}
\hat{d_i} &= d_i - 1 + \delta_i \qquad (1 \le i \le d_k)\\
\hat{d_i} &= d_i + \delta_i \qquad (d_k + 1 \le i \le k-1)\\
\hat{d_k} &= 0\\
\hat{d_i} &= d_i = 0 \qquad (k+1 \le i \le n),
\end{aligned}$$

where $0 \le \delta_i \le d_k$ and $\sum_{i=1}^{k-1} \delta_i = 2d_k$. Indeed, d_k edges of the form v_iv_k $(1 \le i \le d_k)$ are removed and then they are put back: δ_i of them become adjacent to v_i $(1 \le i \le k-1)$. It follows that

$$\begin{aligned}
&\sum_{i=1}^n \frac{1}{\hat{d_i}+1} - \sum_{i=1}^n \frac{1}{d_i+1}\\
&\quad= \sum_{i=1}^{d_k} \left(\frac{1}{d_i+\delta_i} - \frac{1}{d_i+1} \right) + \sum_{i=d_k+1}^{k-1} \left(\frac{1}{d_i+1+\delta_i} - \frac{1}{d_i+1} \right) + 1 - \frac{1}{d_k+1}\\
&\quad= \sum_{i=1}^{d_k} \frac{1-\delta_i}{(d_i+\delta_i)(d_i+1)} + \sum_{i=d_k+1}^{k-1} \frac{-\delta_i}{(d_i+1+\delta_i)(d_i+1)} + 1 - \frac{1}{d_k+1}.
\end{aligned}$$

We aim to show that it is positive so it is enough to show that

$$\begin{aligned}
L := \sum_{i=1}^{d_k} \frac{\delta_i}{(d_i+\delta_i)(d_i+1)} + \sum_{i=d_k+1}^{k-1} \frac{\delta_i}{(d_i+1+\delta_i)(d_i+1)} + \frac{1}{d_k+1}\\
< 1 + \sum_{i=1}^{d_k} \frac{1}{(d_i+\delta_i)(d_i+1)}.
\end{aligned}$$

In fact, we will show something slightly stronger, namely, that L (the left hand side of the above inequality) is at most 1. Indeed,

$$\begin{aligned}
L &\le \sum_{i=1}^{d_k} \frac{\delta_i}{(d_i+1)^2} + \sum_{i=d_k+1}^{k-1} \frac{\delta_i}{(d_i+1)^2} + \frac{1}{d_k+1}\\
&\le \sum_{i=1}^{k-1} \frac{\delta_i}{(d_k+1)^2} + \frac{1}{d_k+1} = \frac{2d_k}{(d_k+1)^2} + \frac{1}{d_k+1} \le \frac{3}{d_k+1},
\end{aligned}$$

which is at most 1, provided $d_k \geq 2$. For $d_k = 1$, we get $L \leq \frac{2d_k}{(d_k+1)^2} + \frac{1}{d_k+1} = 1$.

The claim is proved, provided that $\ell = 0$. It follows that if $m = \binom{s}{2}$ for some integer s, then the upper bound

$$u(m) := \max \left\{ \sum_{i=1}^{n} \frac{1}{\deg_G(v_i) + 1} : G = (V, E), V = \{v_1, \ldots, v_n\}, |E| = m \right\}$$

attains its maximum for $G_{\max}$, that is,

$$u(m) = s \cdot \frac{1}{(s-1)+1} + (n-s) \cdot \frac{1}{0+1} = n - s + 1.$$

In general, s is the largest integer such that $\binom{s}{2} \leq m$. Note that $u(m)$ is clearly a decreasing function of m; indeed, adding an edge to any graph G, decreases the corresponding sum. We get

$$n - s = u\left(\binom{s+1}{2}\right) < u(m) \leq u\left(\binom{s}{2}\right) = n - s + 1.$$

Finally, note that $n - s + 1 < n - \sqrt{2m} + 2$ as $(s+1)^2/2 > \binom{s+1}{2} > m$ and $n - s > n - \sqrt{2m} - 1$ as $(s-1)^2/2 < \binom{s}{2} \leq m$. The conclusion is that the upper bound is (almost) sharp and the proof is finished. $\square$

The above theorem provides natural observations about bounds for $\mathbb{E}(Y)$ and $\mathbb{E}(U)$ using the graphs that provide tight bounds for $\mathbb{E}[|X_0|]$, namely:

Observation 4.2. *Consider a set $\mathbb{G}$ of all graphs with n vertices and m edges. Then:*

(a) there exists $G \in \mathbb{G}$ for which $\mathbb{E}(Y) \leq 2m/n + 1$;
(b) there exists $G \in \mathbb{G}$ for which $\mathbb{E}(Y) \geq \sqrt{2m}$;
(c) for all $G \in \mathbb{G}$ we have $\mathbb{E}(U) \geq n^2/(2m+n)$ and the bound is tight;
(d) there exists $G \in \mathbb{G}$ for which $\mathbb{E}(U) \geq n - \sqrt{2m} - 1$.

It is conjectured that all bounds given above are tight. However, we do not provide the proofs of these conjectures in this text. Let us just highlight, in contrast to our results for the $\mathbb{G}(n,p)$ model, that the sequence $(\mathbb{E}[|X_k|])$ does not have to be decreasing for general graphs. For example, consider 3×3 regular toroidal grid (every vertex has degree 4). Table 4 shows the exact values for $\mathbb{E}[|X_k|]$ and it can be seen that the maximum is achieved for $k = 2$. Similarly, for a cycle consisting of 4 vertices we have $\mathbb{E}[|X_0|] = 4/3$, $E[|X_1|] = 5/3$, $\mathbb{E}[|X_2|] = 2/3$, $\mathbb{E}[|X_4|] = 1/3$ with maximum for $k = 1$.

5 Concluding remarks

In this paper we have given bounds on random schedules for: the shortest time required to finish parallel execution of the schedule Y, the number of

k	$\mathbb{E}[\lvert X_k\rvert]$	k	$\mathbb{E}[\lvert X_k\rvert]$	k	$\mathbb{E}[\lvert X_k\rvert]$
0	$9/5 = 1.800$	3	$173/112 \approx 1.545$	6	$337/1680 \approx 0.201$
1	$25/14 \approx 1.786$	4	$193/168 \approx 1.149$	7	$223/5040 \approx 0.044$
2	$4679/2520 \approx 1.857$	5	$155/252 \approx 0.615$	8	$1/240 \approx 0.004$

Table 4: Values of $\mathbb{E}[|X_k|]$ for regular toroidal 3×3 grid.

processors required in greedy allocation scheme U, and the number of processors required in optimal allocation Q. There are, however, several other topics that are interesting and left for further research.

First, while we have relatively good bounds for Y and U, the bounds for Q could be improved, especially for sparse graphs. Our simulation study shows the direction for the research, but we left the problem of formally proving the proper rate of growth of Q open.

The second avenue for more analysis is the derivation of bounds for Y, U and Q for schedules generated by random orderings of arbitrary graphs. We have provided some preliminary results in the text, but the problem in general is left open. Apart from arbitrary graphs, one could concentrate on analysis of special classes of graphs, e.g. grids, geometric graphs, etc., that are important from practical point of view.

Another research avenue would be to analyze Q not under assumption of an optimal allocation but some suboptimal allocation generated by one of the scheduling heuristics mentioned in the Introduction.

Finally, the model analyzed in the text can be naturally extended to weighted graphs or other assumptions about schedule generation policy; in particular, schedules generated by sampling vertices from initial graph with replacement.

Acknowledgments

Authors would like to thank Wit Jakuczun for help with setting up computations for comparison of MIP and CP solvers efficiency.

References

[1] M. Albert and A. Frieze. Random graph orders. *Order*, 6:19–30, 1989.

[2] N. Alon, B. Bollobás, G. Brightwell, and S. Janson. Linear extensions of partial random order. *The Annals of Applied Probability*, 4(1):108–123, 1994.

[3] K. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.

[4] A. Barak and P. Erdos. On the maximal number of strongly independent vertices in a random acyclic directed graph. *SIAM J. Alg. Disc. Meth.*, 5:508–514, 1984.

[5] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Handbook on scheduling: from theory to applications*. Springer, 2007.

[6] B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.

[7] B. Bollobás and G. Brightwell. The width of random graph orders. *Math. Scientist*, 20:69–90, 1995.

[8] B. Bollobás and G. Brightwell. The dimension of random graph orders. In R. L. Graham and J. Nešetřil, editors, *The Mathematics of Paul Erdös II*, pages 51–69. Springer Berlin Heidelberg, 1997.

[9] B. Bollobás and G. Brightwell. The structure of random graph orders. *SIAM Journal on Discrete Mathematics*, 10(2):318–335, 1997.

[10] P. Brucker. *Scheduling Algorithms*. Springer, 2007.

[11] P. Brucker and S. Knust. *Complex Scheduling*. Springer, 2012.

[12] H. S. Chwa, J. Lee, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin. Global edf schedulability analysis for parallel tasks on multi-core platforms. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1331–1345, 2017.

[13] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J.-M. Vincent, and F. Wagner. Random graph generation for scheduling simulations. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 60:1–60:10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[14] R. Couturier, editor. *Designing scientific applications on gpus*. CRC Press, 2013.

[15] E. L. Demeulemeester and W. S. Herroelen, editors. *Project Scheduling: A Research Handbook*. Springer, 2002.

[16] I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[17] A. Frieze and M. Karoński. *Introduction to random graphs*. Cambridge University Press, 2015.

[18] R. M. Fujimoto. *Parallel and distributed simulation systems*. John Wiley & Sons, 2000.

[19] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Labs Technical Journal*, 45(9):1563–1581, 1966.

[20] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[21] L. Hamill and N. Gilbert. *Agent-based modelling in economics*. John Wiley & Sons, 2015.

[22] IBM. Ibm ilog cplex optimization studio cplex user's manual, version 12 release 7. Technical report, IBM Corp, 2017.

[23] S. Janson, T. Łuczak, and A. Ruciński. *Random graphs*. John Wiley & Sons, 2011.

[24] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[25] B. Karrer and M. E. J. Newman. Random graph models for directed acyclic networks. *Phys. Rev. E*, 80:046110, 2009.

[26] V. Kindratenko, editor. *Numerical Computations with GPUs*. Springer, 2014.

[27] G. Kunz. Parallel discrete event simulation. In *Modeling and Tools for Network Simulation*, pages 121–131. Springer, 2010.

[28] J. K. Lenstra, A. R. Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of discrete mathematics*, 1:343–362, 1977.

[29] J. Li, Z. Luo, D. Ferry, K. Agrawal, C. Gill, and C. Lu. Global edf scheduling for parallel real-time tasks. *Real-Time Syst.*, 51(4):395–439, 2015.

[30] T. Łuczak. First order properties of random posets. *Order*, 8(3):291–297, 1991.

[31] N. Nethercote, P. Stuckey, R. Becket, S. Brand, G. Duck, and G. Tack. Minizinc: Towards a standard cp modelling language. In C. Bessiere, editor, *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, pages 529–543. Springer, 2007.

[32] C. M. Newman. Chain lengths in certain random directed graphs. *Random Structures and Algorithms*, 3(3):243–253, 1992.

[33] C. M. Newman and J. E. Cohen. A stochastic theory of community food webs iv. theory of food chain lengths in large webs. *Proc. R. Soc. Lond. B*, 288:355–377, 1986.

[34] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 5 edition, 2016.

[35] C. Rackauckas and Q. Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[36] C. Schwindt and J. Zimmermann, editors. *Handbook on Project Management and Scheduling*, volume 1. Springer, 2015.

[37] K. Simon, D. Crippa, and F. Collenberg. On the distribution of the transitive closure in a random acyclic digraph. In T. Lengauer, editor, *Algorithms—ESA '93: First Annual European Symposium Bad Honnef, Germany September 30–October 2, 1993 Proceedings*, pages 345–356. Springer Berlin Heidelberg, 1993.

[38] O. Sinnen. *Task scheduling for parallel systems*. John Wiley & Sons, 2007.

[39] F. Squazzoni. *Agent-based computational sociology*. John Wiley & Sons, 2012.

[40] L. Tesfatsion and K. Judd, editors. *Handbook of computational economics*, volume 2. Elsevier, 1996.

[41] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.

[42] U. Wilensky. Netlogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL., 1999-2018.

[43] N. C. Wormald. The differential equation method for random graph processes and greedy algorithms. In M. Karoński and H. Prömel, editors,

Lectures on Approximation and Randomized Algorithms, volume 73 of *Advanced topics in mathematics*, pages 73–155. Polish Scientific Publishers PWN, 1999.

[44] L. A. Wosley and G. L. Nemhauser. *Integer and Combinatorial Optimization.* Wiley-Interscience, 1999.