Calculation of typical running time of a branch-and-bound algorithm for the vertex-cover problem

Joni Pajarinen, Joni.Pajarinen@iki.fi

October 21, 2007

1 Introduction

The vertex-cover problem is one of a collection of NP-complete problems studied actively in the science community. The problem's solving time is in the worst case exponential, but the running time varies for different vertex-cover realizations. From experimental data, different phases can be identified for the running time and the analysis of vertex-cover algorithms back the observed results up. Interestingly the worst-case typical average solutions are located at the $x_c(c)$ phase boundary, which is studied analytically in the book [2] and in the paper [3]. This summary is based on these two documents.

We start by giving a short description of the vertex-cover problem and introducing symbols related to the problem. We proceed and introduce a branch-and-bound algorithm for finding vertex covers of a given size for a graph. Experimental results are displayed in order to show the running time landscape we are dealing with. The experimental results show different phases, which can be studied analytically further. The vertex-cover algorithm's traversal of the configuration tree is explained. The relationship between different kinds of configuration tree traversals and the phases shown in the experimental results is described.

The analysis of the vertex-cover algorithm is started by introducing the first-moment method. The method provides us with a lower bound for the $x_c(c)$ phase boundary for large c values. It also shows a general way of using the average as a bound that can be exploited in similar situations. We start the analysis of the typical running time by examining the behaviour of the algorithm during the first descent into the configuration tree. This produces results that can be used to find the nodes in the configuration tree, where a solution is unrecoverable and a subtree must be backtracked. In the backtracking analysis the typical running time for the largest subtree is approximated. This yields an analytical solution for the typical running time in the backtracking phase. The analytical results are compared to the experimental results presented before.

As the last step in this paper, a summary of the most important results is given.

2 A branch-and-bound algorithm for the vertex-cover problem

In this section we will quickly describe the vertex-cover problem and outline a branch-and-bound algorithm for finding a vertex-cover of a maximum size for a graph. Typical running time experimental data for the branch-and-bound algorithm from simulation runs is presented and the relationship of the configuration tree traversal to the typical running time of the algorithm is discussed.

2.1 Vertex-cover problem

A graph is denoted by $G = (V, E)$. V is the set of vertices and E is the set of edges. Graphs can be sampled randomly from either a fixed number N of vertices and fixed amount of edges M ensemble $\mathcal{G}(N, M)$, where each possible edge has the same probability of appearance, or from an ensemble $\mathcal{G}(N, p)$, where N is the fixed amount of vertices and p the edge probability. A graph sampled from the ensemble $\mathcal{G}(N, p)$ does not have a fixed number of edges. A valid vertex cover covers vertices so that all edges connect to at least one covered vertex: $V_{vc} \subseteq V : i \in V_{vc} \lor j \in V_{vc}, \forall (i, j) \in E$. The minimum vertex-cover is then $\arg\min_{V_{vc}} |V_{vc}|$. The size of the largest allowed vertex-cover is xN, where $x \in [0, 1]$ and $N = |V|$. x is used here in the vertex-cover size, because it is independent of graph size.

2.2 A branch-and-bound vertex-cover algorithm

The branch-and-bound vertex-cover algorithm goes through different partial vertex covers by marking vertices as either covered or uncovered at each step. It backtracks if it reaches a state, where all the available covering marks xN have been used. The goal is to find a vertex-cover of size at most xN. The algorithm terminates, when it has found a valid vertex-cover or if it has gone through all possible configurations. The process of the algorithm is described by an configuration tree. In the configuration tree a node specifies the current state of all the vertices of the graph.

All vertices are marked as free at the start of the algorithm. The algorithm proceeds by marking a random free vertice as covered if the vertex has free or uncovered neighbours. If the xN covering marks are not all used, the algorithm can go on with the tree traversal, otherwise it has to backtrack. If the algorithm returns to the node by backtracking, then the vertex is uncovered and the other branch in the configuration tree is taken. If a node's all neighbours are covered, it is first marked as uncovered. A simple bound is used to prune the configuration tree: don't mark a vertex uncovered, if it has uncovered neighbours.

In the example figure 1 marking a vertex covered is equal to a left branch in the configuration tree and as uncovered is equal to a right branch.

2.3 Experimental results, traversal of the configuration tree

Figure 2 displays experimental results from running the branch-and-bound algorithm on the vertex-cover problem. The simulations were performed by sampling graphs from the ensemble $\mathcal{G}(N, \frac{c}{N})$. Here N is the number of vertices, c the average vertex degree and $p = \frac{c}{N}$ the edge probability. For each x value graphs were sampled from the ensemble and the algorithm run on each graph. The typical running time was calculated as the normalized and averaged logarithm of the number of configuration tree nodes, that the algorithm had to process. In the figure two exponential phases (C, B) and one linear phase (A) are clearly to be seen. The exponential phases C and B are separated by the static phase boundary $x_c(c)$. $x_c(c)$ is the phase boundary value, below which a random graph almost surely has no vertex cover of size xN and above which a random graph almost surely has a vertex cover of size xN, in the thermodynamic limit $N \to \infty$. The exponential phase B and the linear phase A are separated by the dynamical phase boundary $x_b(c)$. The worst case typical running time is observable at the static phase boundary $x_c(c)$.

Figure 1 shows example configuration trees for the branch-and-bound algorithm in the three different phases. In the sub figure A, the example solution (black circle) is found during the first descent into the configuration tree. Here we don't have to backtrack at all and the solution time

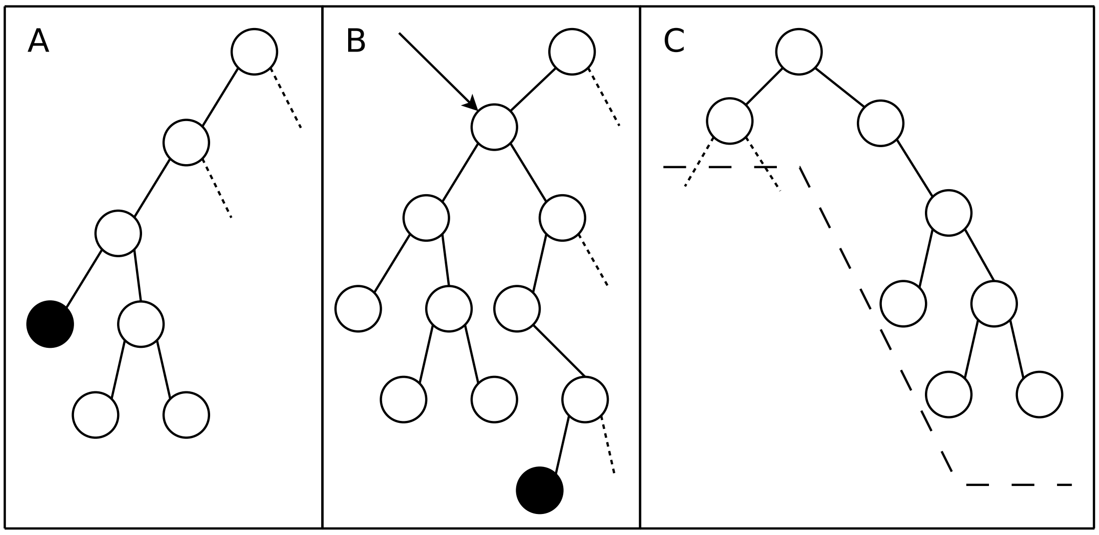

Figure 1: Example configuration trees. The three sub figures A, B and C show examples of configuration trees in the three different phases. In sub figure A the black circle is an example solution in the easy recoverable phase, where only a straight or almost straight descent is necessary. The black circle in sub figure B is an example solution in the hard recoverable phase, where we have to backtrack to the circle pointed to by the arrow. In sub figure C, in the hard unrecoverable phase, we have to traverse the tree until we know a solution can't be found (to the long dashed line).

is obviously linear. The probability for the algorithm to find a solution in linear time is larger, the larger the maximum vertex-cover size xN is. This linear running time can be observed as phase A in figure 2. The example solution (black circle) in the sub figure B had to be found by backtracking. The algorithm had to backtrack the whole left subtree of the node pointed to by the arrow. This is the exponential coverable phase B shown in figure 2. The long dashed line in sub figure C shows the depth of the traversal of the configuration tree, when the maximum vertex-cover size xN is so small that a solution can't be found from the graph. This case can be seen in figure 2 as the unrecoverable exponential phase C in terms of typical running time.

3 First-moment method

In this section we will show a method for calculating a lower bound on the static phase boundary $x_c(c)$. We will use a graph ensemble $\mathcal{G}(N, M)$ with fixed vertex N and edge number $M = \frac{c}{2}N$. This should give a reasonable result, because even if we would use the graph ensemble $\mathcal{G}(N, p)$, with the specified edge probability $p = \frac{c}{N}$, the edge number would concentrate around the edge average $M = \frac{c}{2}N$ in the thermodynamic limit.

An upper bound for the probability that graph G has a vertex-cover can be obtained by using the average number of vertex-covers. This also explains the name of the method.

$$P(\exists V_{vc}(G), |V_{vc}(G)| = xN) \leq \overline{\mathcal{N}_{vc}(G, xN)}$$

This bound holds, because a graph with a vertex-cover contributes 1 on the left hand side and its number of vertex-covers on the right hand side. Another way to look at it is, that the average is the probability weighted with the number of vertex-covers (≥ 1).

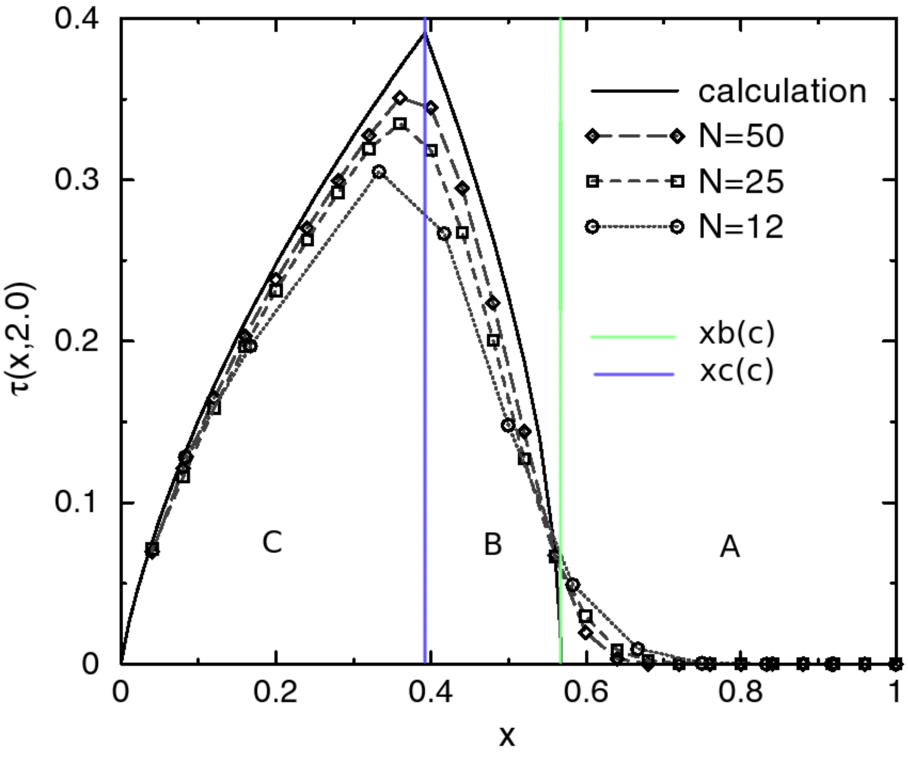

Figure 2: Experimental results for typical running time and the analytical solution. $\tau(x|c) = \lim_{N\to\infty} \frac{1}{N}\overline{\ln t_{bt}(\tilde{G},\tilde{x})}$ is the normalized and averaged logarithm of running time. C and B are the exponential phases and A is the linear phase. Dynamic phase boundary $x_b(c)$ and the static phase boundary $x_c(c)$ are also shown.
Original figure is from [3].

The number of potential vertex covers V_{vc}^* is $\begin{pmatrix} N \\ xN \end{pmatrix}$, because we try to cover xN vertices with the available covering marks from the total of N vertices in the graph. In V_{vc}^* the probability that both vertices of an edge are uncovered is $(1-x)^2$ and that at least one is covered is $1-(1-x)^2 = x(2-x)$. If we consider that a potential vertex-cover must cover all edges, for it to be a valid vertex-cover, then we get the average vertex-cover number as

$$\overline{\mathcal{N}_{vc}(G,xN)} = \begin{pmatrix} N \\ xN \end{pmatrix} \underbrace{[x(2-x)]}_{P(\text{edge covered})}{}^{\overbrace{\frac{c}{2}N}^{\#\text{ edges}}}$$

This gives us

$$\overline{\mathcal{N}_{vc}(G,xN)} = \begin{pmatrix} N \\ xN \end{pmatrix} [x(2-x)]^{\frac{c}{2}N} = \frac{N!\,[x\,(2-x)]^{\frac{c}{2}N}}{(xN)!\,[(1-x)\,N]!}$$

$$= e^{\left\{\ln(N!)+\frac{c}{2}N\ln[x(2-x)]-\ln[(xN)!]-\ln[\{(1-x)N\}!]\right\}}$$

$$\simeq e^{\left\{N\left[-x\ln x-(1-x)\ln(1-x)+\frac{c}{2}\ln\{x(2-x)\}\right]\right\}} \quad (1)$$

Stirling's approximation $\ln(N!) \simeq N\ln N - N$

Now we have in the exponent a value, which determines in the thermodynamic limit ($N\to\infty$), whether the average is zero or above zero. The exponent changes sign at $x_{an}(c) < x_c(c)$. $x_{an}(c)$ is a lower bound on $x_c(c)$, because for values smaller than $x_{an}(c)$ the average $\overline{\mathcal{N}_{vc}(G,xN)}$ goes in the thermodynamic limit to 0 and there is no vertex-cover.

$$0 = -x_{an}(c)\ln x_{an}(c) - (1-x_{an}(c))\ln(1-x_{an}(c)) + \frac{c}{2}\ln\{x_{an}(c)\,(2-x_{an}(c))\} \quad (2)$$

The asymptotic for $x_{an}(c)$ for large average degree c is given by

$$x_{an}(c) = 1 - 2\frac{\ln(c)}{c} + \mathcal{O}\left(\frac{\ln\{\ln(c)\}}{c}\right)$$

and the precise asymptotic is calculated in [1]:

$$x_c(c) = 1 - \frac{2}{c}(\ln(c) - \ln\{\ln(c)\} - \ln(2) + 1) + o(c^{-1})$$

$x_{an}(c) \in [0, 0.5]$ plotted using the equation 2 is shown in figure 3. Comparing $x_{an}(c)$ with $x_c(c)$ shown in figure 4 hints that $x_{an}(c)$ really is a lower bound for $x_c(c)$.

4 Typical running time analysis

4.1 Analysis of first descent into the configuration tree

This subsection deals with analysing the first descent into the configuration tree, i.e. the linear phase, where no backtracking occurs. When searching for a vertex-cover in the graph $G \in \mathcal{G}(N, \frac{c}{N})$, we consider covered vertices and edges as removed from the graph. Thus at time step T, the current graph is $G \in \mathcal{G}(N-T, \frac{c}{N})$. The graph at time step T remains a random graph, because the order of vertex removal is random. For some heuristics this is not true. In each step exactly one vertex is removed from the graph and at time step T the average edge degree is

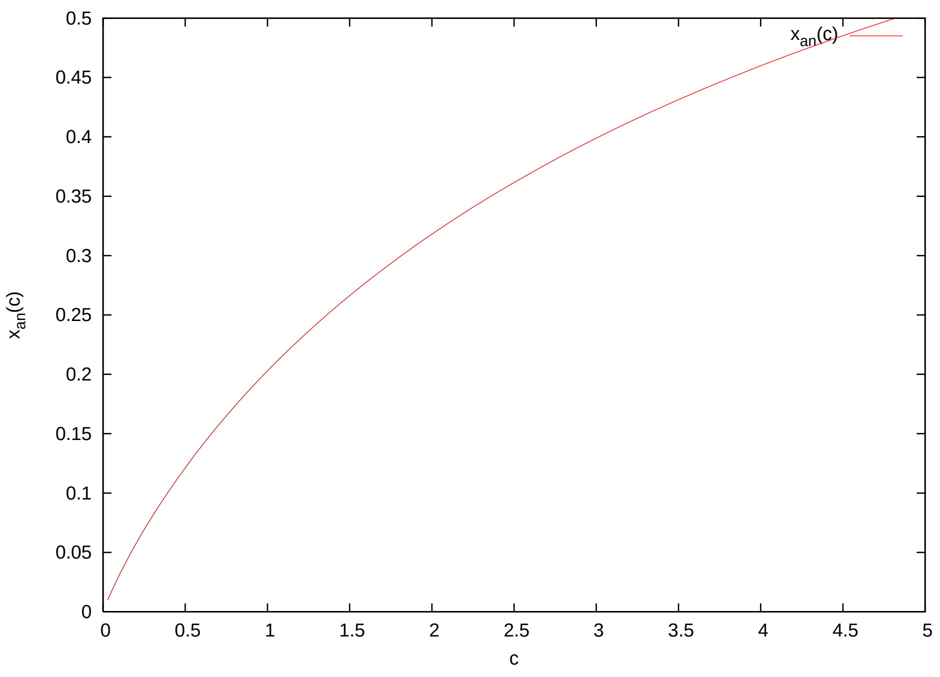

Figure 3: $x_{an}(c) \in [0, 0.5]$ plotted using equation
$0 = -x_{an}(c) \ln x_{an}(c) - (1 - x_{an}(c)) \ln (1 - x_{an}(c)) + \frac{c}{2} \ln \{x_{an}(c)(2 - x_{an}(c))\}$

$c(T) \simeq (N - T)\frac{c}{N} = (1 - \frac{T}{N})c$, because the edge probability $p = \frac{c}{N}$ remains the same over time. We use rescaled time $t = \frac{T}{N}$, which is continuous in the thermodynamic limit. Using a rescaled time simplifies the calculations. The current graph is then $G \in \mathcal{G}((1 - t)N, \frac{c}{N})$.

Because covering marks are put on all vertices except isolated ones, we are interested in the probability that a vertex is not isolated. This can be calculated with the static edge probability and with the number of possible edge endpoints at time t. For a single vertex at time t:

$$P(\text{vertex is not isolated}) = 1 - \underbrace{\left(1 - \frac{c}{N}\right)}_{P(\text{no edge})}^{\overbrace{(1-t)N - 1}^{\#\text{ possible edge endpoints}}}$$

$$= 1 - e^{\left\{((1-t)N-1)\ln\left(1-\frac{c}{N}\right)\right\}} \simeq 1 - e^{\left\{(1-t)\lim_{N\to\infty} N \ln\left(1-\frac{c}{N}\right)\right\}}$$

$$= 1 - e^{\left\{(1-t)\lim_{N\to\infty} -N\frac{c}{N}\right\}} = 1 - e^{\{-(1-t)c\}}$$

With the probability that a vertex is not isolated, i.e. that a vertex is covered, we can calculate the amount of available covering marks by subtracting from the original covering mark amount the amount used at time t:

$$X(t) = xN - N\int_0^t P(\text{vertex is not isolated})dt'$$

$$= xN - N\int_0^t \left(1 - e^{\{-(1-t)c\}}\right) dt'$$

$$= xN - Nt + N\frac{e^{-(1-t)c} - e^{-c}}{c}$$

$x(t)$ is the amount of available covering marks at time t divided by the amount of vertices at time t. The average vertex degree decreases linearly with time. Putting this together gives us the trajectory for the first descent:

$$c(t) = (1-t)c$$

$$x(t) = \frac{X(t)}{N(t)} = \frac{x-t}{1-t} + \frac{e^{-(1-t)c} - e^{-c}}{(1-t)c}$$

Figure 4 displays trajectories in the (x, c) plane. Phase boundary $x_b(c)$, displayed in the figure, can be calculated by setting $x(t') = 1$ and then $t' = 1$:

$$x(t') = 1 \Rightarrow 1 = \frac{x_b(c) - t'}{1-t'} + \frac{e^{-(1-t')c} - e^{-c}}{(1-t')c}$$

$$\Leftrightarrow 1 - t' = x_b(c) - t' + \frac{e^{-(1-t')c} - e^{-c}}{c}$$

$$t' = 1 \Rightarrow 0 = x_b(c) - 1 + \frac{1 - e^{-c}}{c}$$

$$\Leftrightarrow x_b(c) = 1 + \frac{e^{-c} - 1}{c}$$

If x starts below this boundary, an exponential running time is to be expected, because the first descent does not find a solution.

4.2 Backtracking analysis

The coverable exponential running time phase B is shown in figure 2. In this phase backtracking occurs, when there are not enough covering marks available to cover the remaining random subgraph. Backtracking analysis can be difficult, because of configuration tree node dependencies. The problem is approached level by level of the configuration tree and the number of nodes at each level is taken into account. This is feasible, because the computation time depends on the number of nodes, but not on the order they are gone through.

At time $\tilde{t}$ the remaining covering marks $\tilde{x}\tilde{N}$ of the first descent are not enough. From the point of view of the configuration tree, the configuration tree node at time $\tilde{t}$ is pointed to by the blue arrow in the example figure 1. $x_c(c)$ and the first descent cross at time $\tilde{t}$ at $(\tilde{x}, \tilde{c})$. At time $\tilde{t}$ there are $\tilde{x}\tilde{N}$ covering marks remaining. As discussed in the previous section, we always have a random graph, when first descending the configuration tree and removing randomly selected vertices and edges. The subgraph at time $\tilde{t}$ is $\tilde{G} \in \mathcal{G}(\tilde{N}, \frac{\tilde{c}}{\tilde{N}})$ and it must be backtracked in exponential time, because it has no solution. The subgraph $\tilde{G}$ dominates the running time, which can be approximated with the amount of nodes in the subgraph.

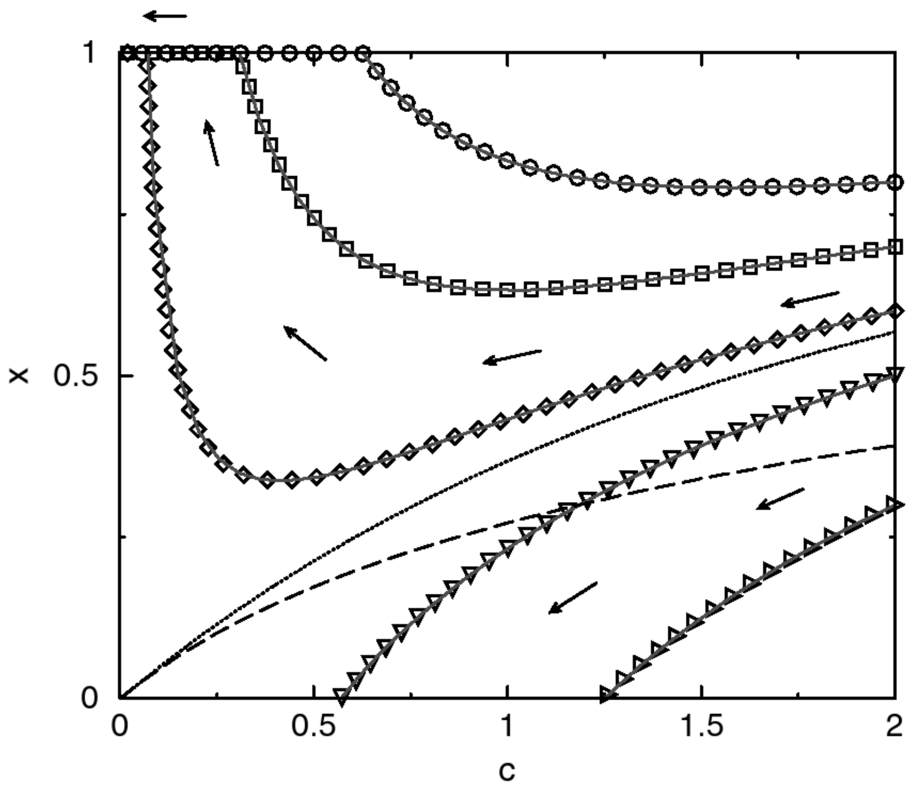

Figure 4: First descent into the configuration tree. Dotted line is $x_b(c)$ and long dashed line is $x_c(c)$. The lines start at $c = 2.0$. The symbols are numerical results from a random graph with 10^6 vertices. Figure is from [3].

From the experimental data we know that exponential solution times are log-normal distributed for large N. The typical solution time is $e^{N\tau(x,c)}$. Denote with $t_{bt}\left(G_{\tilde{N},\frac{\tilde{c}}{\tilde{N}}},\tilde{x}\right)$ the backtracking time of the uncoverable $G_{\tilde{N},\frac{\tilde{c}}{\tilde{N}}}$. The normalized averaged logarithmic running time is the quenched average $\tau(x,c)=\lim_{N\to\infty}\frac{1}{N}\overline{\ln\left[t_{bt}\left(G_{\tilde{N},\frac{\tilde{c}}{\tilde{N}}},\tilde{x}\right)\right]}$. Here the logarithm enforces that we obtain the typical running time, because the solution times are log-normal distributed. The backtracking time is upper bounded by the number of leaves of the subtree times the number of vertices in the subgraph $t_{bt}\le\tilde{N}\mathcal{N}_l$. The linear $\tilde{N}\in\mathcal{O}(N)$ contribution can be discarded and we just use the number of leaves to approximate the exponential subgraph running time:

$$\tau(x,c)\le\lim_{N\to\infty}\frac{1}{N}\overline{\ln\left[\mathcal{N}_l\left(G_{\tilde{N},\frac{\tilde{c}}{\tilde{N}}},\tilde{x}\right)\right]}$$

The number of leaves is upper bounded by the distribution of remaining covering marks on the remaining vertices$\begin{pmatrix}\tilde{N}\\ \tilde{x}\tilde{N}\end{pmatrix}$. Similar to the calculations done in equation 1 we get for the unbounded version of the algorithm

$$\tau(x,c)\le\frac{c}{\tilde{c}}\left\{\tilde{x}\ln\tilde{x}+(1-\tilde{x})\ln(1-\tilde{x})\right\}$$

For the bounded version of the branch-and-bound algorithm, we have to take into account the level at which the bound stops the traversing of the configuration tree. We mark with $\kappa\tilde{N}$ the level in the subtree, where the bound becomes effective. Omitting a lot of calculations the result for the bounded version is presented here:

$$\tau(x,c)\simeq\max_{\kappa=\tilde{x},\dots,1}\left[\frac{c}{\tilde{c}}\kappa s_{an}\left(\frac{\tilde{x}}{\kappa},\tilde{c}\kappa\right)\right]\tag{3}$$

$$s_{an}(x^*,c^*)=-x^*\ln x^*-(1-x^*)\ln(1-x^*)+\frac{c^*}{2}\ln(x^*\left[2-x^*\right])$$

The typical running time is approximated with the largest subtree by finding the saddle point 3 on the κ value. Figure 2 shows also this analytical result. It is worth noting that the analytical results seem to fit well with the experimental data.

5 Summary

A simple branch-and-bound algorithm was introduced for the vertex-cover problem. Experimental data of typical running times for the algorithm was shown. Two exponential dynamical phases and one linear phase were discernible from the data.

The first-moment method was explained and a lower bound on $x_c(c)$ was constructed by using the average vertex-cover amount as an upper bound for the vertex-cover probability of a graph.

Analysis of the first descent vertex-cover algorithm was done and (x,c)-trajectories of first descent runs were analysed and experimental data of them displayed. The dynamic phase boundary $x_b(c)$, which shows where exponential solution times start occurring, was calculated. In the analysis, important steps were the considering of the current graph with covered vertices and edges removed, calculating the probability for a not-isolated vertex and calculating the amount of available covering marks at a time step, using a continuous (in the thermodynamic limit) rescaled time.

In the backtracking analysis the typical running time was calculated for the unbounded version of the algorithm using the amount of leaves in the backtracking subtree. For the bounded version the typical running time was found by considering the largest subgraph that was possible with the bound effective.

References

[1] A. M. Frieze. *Discr. Math.*, 81(171), 1990.

[2] Alexander K. Hartmann and Martin Weigt. *Phase Transitions in Combinatorial Optimization Problems.* WILEY-VCH, 2005.

[3] Martin Weigt and Alexander K. Hartmann. Typical solution time for a vertex-covering algorithm on finite-connectivity random graphs. *Physical Review Letters*, 86(8), February 2001.