Supplementary Material of The Manuscript "Using Statistical Measures and Machine Learning for Graph Reduction to Solve Maximum Weight Clique Problems"

Yuan Sun, Xiaodong Li, and Andreas Ernst

THIS document provides additional experiments to support our findings in the main paper. In particular, we investigate whether the performance of our ranking-based measure and correlation-based measure can be improved by tuning the parameters in Section A. In Section B, we investigate whether the performance of our MLPR model (Machine Learning for Problem Reduction) can be improved by enlarging the training datasets. Finally we investigate whether our MLPR model trained on small and medium sized synthetic graphs can be generalized to very large real-world graphs in Section C.

A Parameter Tuning

In Section 5.2 of the main paper, we have presented experimental results showing that 1) our ranking-based measure f_r (with $\epsilon_r = 0.01$) can not significantly reduce problem size for medium-sized graphs M^{te}; and 2) our correlation-based measure f_c (with $\epsilon_c = 0$) is not very effective in reducing the size for large graphs L^{te}. Here we investigate whether the performance of our ranking-based measure and correlation-based measure can be improved by tuning the parameters.

A.1 Parameter Tuning for Ranking-based Measure

We test two parameter values ($\epsilon_r = 0.01$ and $\epsilon_r = 0.03$) to investigate whether the performance of our ranking-based measure f_r can be improved when used to solve the medium-sized graphs (M^{te}) from DIMACS. We apply our ranking-based measure with each parameter setting to reduce problem size for these graphs as a pre-precessing step, and use the TSM algorithm to solve the reduced problem. The best objective value generated by TSM within the cutoff time (1000 seconds) is regarded as an indication of the effectiveness of problem reduction. The mean and standard deviation of the best objective values obtained in 25 independent runs are presented in Table S1.

TABLE S1: The results of TSM-f_r with different ϵ_r values (0.01 or 0.03) when used to solve the 9 hard medium-sized graphs from DIMACS. $\bar{y}$ and σ_y denote the mean and standard deviation of best objective values generated in 25 independent runs within the cutoff time (1000 seconds); and $\bar{p}$ denotes the mean ratio of selected vertices. The statistically best $\bar{y}$ is in bold. The last row $\bar{r}$ is the average ranking of each algorithm across all datasets.

G	TSM-f_r ($\epsilon_r = 0.01$)			TSM-f_r ($\epsilon_r = 0.03$)		
	$\bar{y}$	σ_y	$\bar{p}$	$\bar{y}$	σ_y	$\bar{p}$
M_1^{te}	**10069**	144	0.93	9974	106	0.66
M_2^{te}	7479	277	1.00	7439	229	1.00
M_3^{te}	2431	31	0.98	**2455**	16	0.50
M_4^{te}	7805	231	1.00	7756	242	1.00
M_5^{te}	2445	53	0.79	**2574**	38	0.24
M_6^{te}	34265	0	1.00	34265	1	1.00
M_7^{te}	109870	77	1.00	109840	80	1.00
M_8^{te}	4678	92	1.00	4621	102	1.00
M_9^{te}	5306	168	1.00	5299	146	0.95
$\bar{r}$	1.22			1.11		

We observe that when using a larger parameter value $\epsilon_r = 0.03$, our ranking-based measure f_r is able to remove more vertices from 4 graphs, i.e., M_1^{te}, M_3^{te}, M_5^{te} and M_9^{te}. However, the TSM-f_r ($\epsilon_r = 0.03$) algorithm only improves over TSM-f_r ($\epsilon_r = 0.01$) on 2 of the 4 graphs. For the other 5 graphs, our ranking-based measure f_r with $\epsilon_r = 0.03$ is still unable to remove any vertex, partially because these graphs are very dense as shown in Table 1 of the main paper. Note that we can further increase the value of ϵ_r to potentially reduce the size of these dense graphs, but this will result in removing too many vertices from other graphs (e.g., M_5^{te}).

In Section 5.2 of the main paper, we have shown that our ranking-based measure f_r with $\epsilon_r = 0.01$ works well for large sparse graphs L^{te}. Thus we suggest a general guidance on the parameter setting for our ranking-based measure f_r here: 1) $\epsilon_r = 0.01$ for large sparse graphs; and 2) $\epsilon_r = 0.03$ for medium-sized graphs.

- *Y. Sun and X. Li are with School of Science, RMIT University, Melbourne, 3001, Victoria, Australia. E-mail: yuan.sun@rmit.edu.au; xiaodong.li@rmit.edu.au*
- *A. Ernst is with School of Mathematical Sciences, Monash University, Clayton, 3800, Victoria, Australia. E-mail: andreas.ernst@monash.edu*

Manuscript received May 3, 2019; revised September 30, 2019.

TABLE S2: The results of TSM-f_c with different ϵ_c values (0 or 0.01) when used to solve the 11 hard large-sized real-world graphs. $\bar{y}$ and σ_y denote the mean and standard deviation of best objective values generated in 25 independent runs within the cutoff time (1000 seconds); and $\bar{p}$ denotes the mean ratio of selected vertices. The statistically best $\bar{y}$ is in bold. The last row $\bar{r}$ is the average ranking of each algorithm across all datasets.

G	TSM-f_c ($\epsilon_c = 0.00$)			TSM-f_c ($\epsilon_c = 0.01$)		
	$\bar{y}$	σ_y	$\bar{p}$	$\bar{y}$	σ_y	$\bar{p}$
L_1^{te}	32000	36	0.38	**32060**	40	0.15
L_2^{te}	26844	398	0.39	26855	305	0.15
L_3^{te}	31748	318	0.39	31629	358	0.16
L_4^{te}	29548	0	0.39	29487	292	0.15
L_5^{te}	32731	303	0.40	32835	0	0.17
L_6^{te}	32572	612	0.39	32807	524	0.13
L_7^{te}	30734	533	0.38	30827	161	0.14
L_8^{te}	46699	2017	0.37	**50025**	166	0.13
L_9^{te}	32969	44	0.37	**33085**	0	0.13
L_{10}^{te}	27775	0	0.40	27774	6	0.16
L_{11}^{te}	26190	0	0.40	**26271**	76	0.17
$\bar{r}$	1.36			1.00		

A.2 Parameter Tuning for Correlation-based Measure

In this subsection, we investigate whether the performance of our correlation-based measure f_c can be improved by tuning the parameter ϵ_c on the very large real-world graphs L^{te}. We test two parameter values $\epsilon_c = 0$ and $\epsilon_c = 0.01$, and apply our correlation-based measure to reduce the size of each large graph. We then use the TSM algorithm to search for a maximum weight clique in the reduced graph with the cutoff time set to 1000 seconds. We repeat this process for 25 times to alleviate randomness, and the mean and standard deviation of the best objective values generated in 25 independent runs are presented in Table S2.

We observe that by using a slightly larger parameter value $\epsilon_c = 0.01$, our correlation-based measure f_c can remove more vertices from these large sparse graphs. Significantly, the TSM-f_c algorithm with $\epsilon_c = 0.01$ consistently generates statistically better or equal solution quality than that with $\epsilon_c = 0$. *It suggests that the performance of our correlation-based measure f_c can be improved by using a slightly larger parameter value ϵ_c for large sparse graphs.*

Thus as a general guidance on the parameter setting for our correlation-based measure f_c, we recommend to set $\epsilon_c = 0.01$ for large sparse graphs; and $\epsilon_c = 0$ for medium-sized dense graphs given its good performance shown in Section 5.2 of the main paper.

B Effects of the Size of Training Set

In Section 5.2 of the main paper, we have trained our MLPR model on 8 easy medium-sized graphs (M^{tr}) and used the trained model to reduce problem size for 9 hard medium-sized graphs (M^{te}) from the DIMACS benchmark. Here we investigate whether the performance of our MLPR model can be improved by enlarging the training datasets.

We increase the size of training set by including 18 more graphs from DIMACS, which are listed in Table S3. These graphs are small ($|V| < 1000$) and can be easily solved to optimality by using the TSM algorithm. As each vertex in a graph is a training instance in our model, the enlarged training set contains 26 (18+8) graphs in total with more than 10000 training instances.

TABLE S3: The 18 small and easy graphs from DIMACS which are used as additional training instances in our experiments. $|V|$ is the number of vertices; $|E|$ is the number of edges; and d is the graph density.

Name	$\|V\|$	$\|E\|$	d
brock200_2	200	9876	0.4963
brock200_4	200	13089	0.6577
brock400_2	400	59786	0.7492
brock400_4	400	59765	0.7489
p_hat300-1	300	10933	0.2438
p_hat300-2	300	21928	0.4889
p_hat300-3	300	33390	0.7445
p_hat700-1	700	60999	0.2493
p_hat700-2	700	121728	0.4976
p_hat700-3	700	183010	0.7480
gen200_p0.9_44	200	17910	0.9000
gen200_p0.9_55	200	17910	0.9000
gen400_p0.9_75	400	71820	0.9000
C125.9	125	6963	0.8985
C250.9	250	27984	0.8991
DSJC500.5	500	125248	0.5020
MANN_a27	378	70551	0.9901
keller4	171	9435	0.6491

TABLE S4: The results of TSM, TSM-MLPR$_8$ and TSM-MLPR$_{26}$ when used to solve the 9 hard medium-sized graphs from DIMACS. $\bar{y}$ and σ_y denote the mean and standard deviation of best objective values generated in 25 independent runs within the cutoff time (1000 seconds); and $\bar{p}$ denotes the mean ratio of selected vertices. The statistically best $\bar{y}$ is in bold. The last row $\bar{r}$ is the average ranking of each method across all datasets.

G	TSM		TSM-MLPR$_8$			TSM-MLPR$_{26}$		
	$\bar{y}$	σ_y	$\bar{y}$	σ_y	$\bar{p}$	$\bar{y}$	σ_y	$\bar{p}$
M_1^{te}	10119	0	**10294**	51	0.27	10276	27	0.24
M_2^{te}	7341	0	**8250**	142	0.55	8100	201	0.57
M_3^{te}	2407	0	**2466**	0	0.48	**2466**	1	0.43
M_4^{te}	8228	0	**8898**	161	0.52	**8879**	192	0.51
M_5^{te}	2402	0	2601	42	0.49	**2656**	39	0.42
M_6^{te}	**34259**	0	34253	6	0.98	34202	22	0.96
M_7^{te}	109190	4	**109850**	59	0.98	**109830**	48	0.97
M_8^{te}	4812	0	**4914**	67	0.78	**4927**	57	0.72
M_9^{te}	4762	0	**5260**	144	0.39	**5284**	159	0.36
$\bar{r}$	2.78		1.22			1.44		

We train a machine learning model on the enlarged training set by solving the dual problem of L1-SVM with RBF kernel. We denote this model as MLPR$_{26}$, in contrast to MLPR$_8$ which is trained only on the 8 easy graphs, M^{tr} from Table 1 of the main paper. The parameter settings for MLPR$_{26}$ and MLPR$_8$ are the same: $\epsilon_m = 10$. We test the efficacy of MLPR$_{26}$ and MLPR$_8$ methods on the 9 hard graphs M^{te}. These 9 hard graphs are also from the DIMACS benchmark, and the number of vertices in these graphs varies from 1000 to 4000. We apply the MLPR$_{26}$ and MLPR$_8$ methods to reduce problem size for these 9 hard graphs as a pre-processing step, and use the TSM algorithm to solve the reduced problem. The best objective value generated by TSM within cutoff time (1000 seconds) is regarded as an indication of the efficacy of problem reduction techniques.

The mean and standard deviation of best objective values generated in 25 independent runs by TSM, TSM-MLPR$_8$

TABLE S5: The results of $\text{MLPR}_{\text{small}}$, $\text{MLPR}_{\text{large}}$ and $\text{MLPR}_{\text{none}}$ when incorporated with the 4 algorithms to solve the 11 large real-world hard instances. $y_{\max}$, $\bar{y}$ and σ_y denote the maximum, mean and standard deviation of best objective values generated in 25 independent runs within the cutoff time (1000 seconds); and $\bar{p}$ denotes the mean ratio of selected vertices. The statistically best $\bar{y}$ is in bold and the best $y_{\max}$ is in italic. $\bar{r}$ is the average ranking of each method across all datasets.

Algorithm	G	$\text{MLPR}_{\text{none}}$				$\text{MLPR}_{\text{large}}$				$\text{MLPR}_{\text{small}}$			
		$y_{\max}$	$\bar{y}$	σ_y	$\bar{p}$	$y_{\max}$	$\bar{y}$	σ_y	$\bar{p}$	$y_{\max}$	$\bar{y}$	σ_y	$\bar{p}$
TSM	L_1^{te}	32127	**32105**	7	1.00	*32176*	31958	193	0.02	31988	31309	389	0.07
	L_2^{te}	26412	26412	0	1.00	*27190*	**26757**	378	0.06	*27190*	**26794**	505	0.10
	L_3^{te}	31249	31228	76	1.00	*31940*	**31496**	376	0.06	*31940*	**31276**	613	0.10
	L_4^{te}	27972	27972	0	1.00	*29548*	**29492**	280	0.05	*29548*	**29421**	633	0.10
	L_5^{te}	30310	30310	0	1.00	*32835*	**32760**	260	0.05	*32835*	**32797**	188	0.10
	L_6^{te}	31413	31371	23	1.00	33476	32801	523	0.04	*35698*	**35328**	228	0.07
	L_7^{te}	28232	28232	0	1.00	*30885*	**30757**	459	0.05	*30885*	**30788**	198	0.08
	L_8^{te}	49087	48716	331	1.00	*50355*	**49630**	311	0.06	*50355*	48477	1606	0.08
	L_9^{te}	32791	32658	94	1.00	*33085*	**33085**	0	0.05	*33085*	32910	139	0.08
	L_{10}^{te}	25924	25637	57	1.00	*27775*	**27726**	243	0.05	*27775*	**27730**	224	0.11
	L_{11}^{te}	25205	22749	206	1.00	26558	**26323**	72	0.06	*27500*	**26617**	681	0.11
	$\bar{r}$		2.55				1.18				1.36		
LSCC	L_1^{te}	27314	23969	1812	1.00	*32057*	**31570**	283	0.02	31688	30819	501	0.07
	L_2^{te}	23554	21430	1594	1.00	*27190*	**26625**	620	0.06	*27190*	26119	744	0.10
	L_3^{te}	30129	23923	3195	1.00	*31940*	**31401**	695	0.06	*31940*	**31472**	650	0.10
	L_4^{te}	27469	23066	2006	1.00	*29548*	**29500**	239	0.05	*29548*	29142	666	0.10
	L_5^{te}	31897	25843	3039	1.00	*32835*	**32685**	351	0.05	*32835*	**32751**	292	0.10
	L_6^{te}	34058	24984	4322	1.00	35685	**35392**	231	0.04	*35698*	35137	260	0.07
	L_7^{te}	30401	24463	2320	1.00	*30885*	**30776**	454	0.05	*30885*	**30776**	454	0.08
	L_8^{te}	34163	25531	4511	1.00	*50355*	**48305**	4371	0.06	*50355*	47254	3987	0.08
	L_9^{te}	28673	24163	2875	1.00	*32783*	**31011**	808	0.05	31815	30482	736	0.08
	L_{10}^{te}	24215	19373	2411	1.00	*27775*	**27694**	403	0.05	*27775*	**27697**	389	0.11
	L_{11}^{te}	25358	21804	2121	1.00	*27384*	**26877**	301	0.06	27304	26421	477	0.11
	$\bar{r}$		3.00				1.00				1.64		
WLMC	L_1^{te}	25293	25293	0	1.00	31452	29533	720	0.02	*31876*	**31170**	458	0.07
	L_2^{te}	22332	22332	0	1.00	*27190*	**26253**	1063	0.06	*27190*	**26286**	1228	0.10
	L_3^{te}	28044	28044	0	1.00	*31940*	**30907**	1381	0.06	*31940*	**31108**	1058	0.10
	L_4^{te}	20819	20819	0	1.00	*29548*	**29417**	654	0.05	*29548*	**29436**	387	0.10
	L_5^{te}	29398	29398	0	1.00	*32835*	**32797**	188	0.05	*32835*	**32797**	188	0.10
	L_6^{te}	26557	26557	0	1.00	33224	32649	527	0.04	*35650*	**33676**	1221	0.07
	L_7^{te}	24560	24560	0	1.00	*30885*	**30757**	459	0.05	*30885*	**30808**	181	0.08
	L_8^{te}	34356	34356	0	1.00	*50355*	**40006**	5735	0.06	*50355*	**41912**	6377	0.08
	L_9^{te}	32167	32167	0	1.00	32168	32167	0	0.05	*32400*	**32308**	104	0.08
	L_{10}^{te}	24991	24991	0	1.00	*27775*	**27697**	389	0.05	*27775*	**27620**	539	0.11
	L_{11}^{te}	25205	**25205**	0	1.00	24802	24647	41	0.06	*25519*	**25076**	385	0.11
	$\bar{r}$		2.72				1.45				1.00		
FastWClq	L_1^{te}	31155	**30666**	373	1.00	*31422*	**30728**	386	0.02	31293	**30569**	443	0.07
	L_2^{te}	*27190*	**27025**	386	1.00	*27190*	26678	538	0.06	*27190*	26458	837	0.10
	L_3^{te}	*31940*	**31854**	244	1.00	*31940*	**31738**	349	0.06	*31940*	31116	815	0.10
	L_4^{te}	*29548*	**29548**	0	1.00	*29548*	**29548**	0	0.05	*29548*	**29316**	640	0.10
	L_5^{te}	*32835*	32165	458	1.00	*32835*	**32750**	283	0.05	*32835*	**32718**	317	0.10
	L_6^{te}	35035	34790	120	1.00	35169	**34975**	161	0.04	*35195*	**34835**	214	0.07
	L_7^{te}	*30885*	**30885**	0	1.00	*30885*	**30638**	676	0.05	*30885*	**30731**	473	0.08
	L_8^{te}	*50355*	49912	1932	1.00	*50355*	**50177**	166	0.06	*50355*	48070	1269	0.08
	L_9^{te}	31172	**30251**	525	1.00	*31221*	**30432**	390	0.05	31192	**30511**	433	0.08
	L_{10}^{te}	*27775*	**27775**	0	1.00	*27775*	**27775**	0	0.05	*27775*	**27560**	592	0.11
	L_{11}^{te}	26215	26204	32	1.00	*26814*	**26504**	169	0.06	26773	**26437**	148	0.11
	$\bar{r}$		1.55				1.09				1.45		

or TSM-MLPR_{26} within the cutoff time are presented in Table S4. We can observe that 1) both MLPR_8 and MLPR_{26} can significantly boost the performance of the TSM algorithm; 2) the problem size reduced by MLPR_{26} is generally larger than that by TSM-MLPR_8 for these graphs; and 3) MLPR_{26} does not improve over MLPR_8 when incorporated with TSM to solve the 9 hard problems. *It indicates that the training instances collected from the 8 easy graphs (M^{tr}) are sufficient to generalize our MLPR model to solve the hard problems of similar size (M^{te}).* However in the next section, we will show that collecting more training instances from small graphs help generalize our MLPR model to solve very large real-world problems with very different characteristics.

C Scalability of Machine Learning Model for Problem Reduction

In this section, we present additional experiments to investigate the scalability of our MLPR model for problem reduction. We have observed that our MLPR model trained on 8 medium-sized synthetic graphs from DIMACS (M^{tr}) does not generalize well to very large real-world graphs (L^{te}) used in the main paper, as it tends to remove too many vertices from these large graphs. We infer the reason is that the training instances collected from the 8 medium graphs are biased and do not cover the feature space well. We then solved this issue by collecting more training instances from 18 small-sized graphs from DIMACS, listed in Table S3.

Previously, the MLPR model for medium-sized graphs was trained by solving the dual problem of L1-SVM with RBF kernel. However the prediction time used by this model to reduce the size of large graphs is very long (around 300 seconds). Thus we will train the MLPR model by solving the primal problem of linear L2-SVM ($\epsilon_m = 10$), so that the prediction time can be reduced to around 2 seconds. We term this model as $\text{MLPR}_{\text{small}}$, and compare it against $\text{MLPR}_{\text{large}}$ (trained on large easy graphs L^{tr} with $\epsilon_m = 10$) as well as $\text{MLPR}_{\text{none}}$ (without any problem reduction). The problem reduction models are then incorporated with the 4 solution algorithms – TSM, WLMC, LSCC+BMS and FastWClq, to solve the 11 very large real-world hard graphs (L^{te}), and the results are shown in Table S5.

The results show that our MLPR model trained on small and medium graphs ($\text{MLPR}_{\text{small}}$) generalizes well to very large real-world problem instances. The $\text{MLPR}_{\text{small}}$ model consistently boosts the performance of the 4 solution algorithms, especially for LSCC+BMS and WLMC which are ineffective in solving these large problem instances. Significantly, using our MLPR methods as a preprocessing step, the LSCC+BMS and WLMC algorithms can generate an optimal or near-optimal solution for these large problem instances. We note that the performance of the $\text{MLPR}_{\text{small}}$ model is slightly worse than the $\text{MLPR}_{\text{large}}$ model. This makes sense because the $\text{MLPR}_{\text{large}}$ model is trained on very large real-world graphs with similar size and characteristics to the test graphs; while the $\text{MLPR}_{\text{small}}$ model is trained on medium and small graphs which are very different from the test graphs. The FastWClq algorithm benefits slightly from our problem reduction techniques in terms of solution quality, because it is very effective at solving these 11 large graphs already. *These results indicate that our MLPR model trained on small and medium graphs can be used to effectively reduce the problem size for very large real-world graphs.*

To further show the scalability of our MLPR model, we apply the $\text{MLPR}_{\text{small}}$ model to other very large real-world graphs that have not been considered in the main paper:

1) Collaboration and Citation Networks [1]. We select one citation network "ca-cit-HepTh" and one collaboration network "ca-cit-HepPh" whose density is greater than 0.01. In the ca-cit-HepTh network, a vertex represents a paper, and an edge from vertex u to v indicates that paper u cites paper v. In the ca-cit-HepPh network, vertices represent authors and edges indicate collaborations between authors. These networks are based on the scientific papers from arXiv [2], and have more than ten thousand vertices and more than one million edges.
2) SNAP Social Network [3]. We use 4 web graphs from the Stanford Large Network Dataset Collection [3]. Vertices in the graphs represent web pages and directed edges represent hyperlinks between them [4]. These graphs have more than one hundred thousand vertices and several millions of edges. Thus the density is very low, i.e., around 10^{-5}.

We transfer a directed graph into an undirected graph, and an unweighted graph into a weighted graph based on the same rule used in the main paper. Notably, the maximal weight clique problem in these graphs can be easily solved to optimality by the TSM algorithm, because these graphs are very sparse.

TABLE S6: The results of $\text{TSM-MLPR}_{\text{small}}$ when used to solve other very large real-world graphs. $|V|$ is the number of vertices; $|E|$ is the number of edges. $y_{\max}$, $\bar{y}$ and σ_y denote the maximum, mean and standard deviation of best objective values generated in 25 independent runs; and $\bar{p}$ denotes the mean ratio of selected vertices. The original optimal objective value is marked with *.

Name	$\lvert V\rvert$	$\lvert E\rvert$	$y_{\max}$	$\bar{y}$	σ_y	$\bar{p}$
ca-cit-HepTh	22908	2673133	57469*	57469*	0	0.098
ca-cit-HepPh	28093	4596803	43713*	43713*	0	0.136
web-Google	875713	5105039	4857*	4854	9	0.042
web-NotreDame	325729	1469679	19133*	19133*	0	0.006
web-BerkStan	685230	7600595	22046*	22046*	0	0.004
web-Stanford	281903	2312497	6574*	6481	134	0.007

We use these graphs to test whether our $\text{MLPR}_{\text{small}}$ model can 1) significantly reduce problem size for these graphs; and 2) capture an original optimal solution or near-optimal solution in the reduced graph. We apply our $\text{MLPR}_{\text{small}}$ model to reduce the size for each graph as a pre-processing step, and use the TSM algorithm to solve the maximal weight clique problem in the reduced graph. We repeat this process for 25 independent runs to alleviate randomness, and the maximum, mean, and standard deviation of the best objective values found are shown in Table S6.

We observe that our $\text{MLPR}_{\text{small}}$ model consistently captures the original optimal solution in the reduced graphs for the two collaboration and citation networks as well as two web networks i.e., web-NotreDame and web-BerkStan, in each of the 25 independent runs. Remarkably, the percentage of vertices removed by $\text{MLPR}_{\text{small}}$ from the web-NotreDame and web-BerkStan networks is huge, i.e., more than 99%. For the web-Google and web-Stanford networks, our $\text{MLPR}_{\text{small}}$ model occasionally captures the original optimal solution and overall generates a comparable solution quality, when only a small portion of vertices is selected. *The results from Table S6 confirm that our MLPR model trained on small and medium synthetic graphs can be generalized to reduce problem size for very large real-world graphs.*

Finally, we note that problem reduction for these large sparse graphs is less meaningful, because the maximum weight clique problem in these graphs is typically easy (quick) to solve by simply using an exact solver. The time spent on problem reduction may be even longer than the time required to directly solve the problems in these graphs.

REFERENCES

[1] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
[2] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 2005, pp. 177–187.
[3] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.
[4] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.