Efficient Local Search for Maximum Weight Cliques in Large Graphs

Yi Fan, Zongjie Ma and Kaile Su
Department of Computer Science
Jinan University, Guangzhou, China
Institute for Integrated and Intelligent Systems
Griffith University, Brisbane, Australia
yifan.sysu@gmail.com
zongjie.ma@griffithuni.edu.au
k.su@griffith.edu.au

Chengqian Li
Department of
Computer Science
Sun Yat-sen University
Guangzhou, China
lichengq@mail2.sysu.edu.cn

Cong Rao, Ren-Hau Liu and
Longin Jan Latecki
Department of Computer and Information Science
Temple University, Philadelphia, USA
cong.rao@temple.edu
liurhh@temple.edu
latecki@temple.edu

***Abstract*—In this paper, we develop a local search algorithm to solve the Maximum Weight Clique (MWC) problem. Firstly we design a novel scoring function to measure the benefits of a local move. Then we develop a Cycle Estimation based ReStart (CERS) strategy to resolve the cycling issue in the local search process. Experimental results show that our solver achieves state-of-the-art performances on the large sparse graphs as well as large dense graphs. Also we present a theorem which shows the necessity of the restart strategies in current state-of-the-art local search algorithms.**

I. Introduction

The Maximum Weight Clique (MWC) problem is defined over a graph $G = (V, E, w_{\mathcal{V}}, w_{\mathcal{E}})$. $V = \{v_1, \ldots, v_n\}$ is the vertex set, and each edge $e \in E$ is a 2-element subset of V. We denote $w_{\mathcal{V}} : V \mapsto R_{\geq 0}$ and $w_{\mathcal{E}} : E \mapsto R_{\geq 0}$ as the weighting functions on V and E respectively. A clique C is a subset of vertices in G such that each pair of vertices in C is connected. The MWC problem is to find a clique C which maximizes

$$\sum_{v_i \in C} w_{\mathcal{V}}(v_i) + \sum_{v_i, v_j \in C} w_{\mathcal{E}}(\{v_i, v_j\}). \quad (1)$$

Due to the NP-hardness, the research of the MWC problem focuses on developing various heuristics to find a good clique within a reasonable time limit. Up till now, there are mainly two types of algorithms for the MWC problem: complete algorithms [1], [2], [3], [4], [5] and incomplete ones [6], [7], [8], [9], [10]. Furthermore, algorithms for the maximum weight independent set and minimum weight vertex cover problems can also be applied, *e.g.*, [11], [12].

Local search based methods (*e.g.*, [13], [14], [8]) often suffer from the cycling problem, *i.e.*, the method may spend too much time visiting a small local area in the search space. Recently, [13] proposed a strategy called configuration checking (CC), which exploits the problem structure to reduce cycling in local search. Later [8] modified CC to a more conservative version, which is called the Strong Configuration Checking (SCC) strategy, which significantly improves the intensive search.

To improve extensive search, we develop a Cycle Estimation based ReStart (CERS) strategy to guide the local search for the MWC. More specifically if a local optimum is revisited, the search will restart. To validate the effectiveness of our method, we compare CERS with state-of-the-art MWC solvers including [8], [14], [9], [3], [15], [16]. These methods are evaluated on benchmark graphs including large sparse vertex-weighted graphs and large sparse edge-weighted graphs, as well as large dense edge-weighted graphs in an application called image co-localization [17]. Experimental results show that CERS achieves state-of-the-art performance on these datasets.

II. Preliminaries

A. The Benchmarks

Recently the research on large real-world graphs has become popular. Many of these graphs have millions of vertices and dozens of millions of edges. They are really sparse in that the density can be as small as 0.0001. Also they present the power-distribution law [18], *i.e.*, most vertices have small degrees while very few vertices have great degrees. They are originally unweighted, and we adopt the method in [6] to generate vertex-weighted graphs and edge-weighted graphs respectively. More specifically in vertex-weighted graphs,

1) for the i-th vertex v_i, $w_{\mathcal{V}}(v_i) = (i \mod 200) + 1$;
2) for all $e \in E$, $w_{\mathcal{E}}(e) = 0$.

In edge-weighted graphs,

1) for the edge $e = \{v_i, v_j\}$, $w_{\mathcal{E}}(\{e\}) = [(i + j) \mod 200] + 1$;
2) for all $v \in V$, $w_{\mathcal{V}}(v) = 0$.

We also compare state-of-the-art MWC solvers in image co-localization. Given a set of images, the task of object co-localization is to simultaneously localize objects of the same class in each image, where at most one object needs to be localized in each image. To evaluate the performance of our method in comparison to other approaches, experiments are conducted on the PASCAL VOC 2007 image dataset [19], which is a classic dataset in computer vision.

B. Multi-neighborhood Search

Given an edge $e = \{u, v\} \in E$, we say that u and v are neighbors and u and v are connected. We denote the set of neighbors of vertex v as $N(v) = \{u | u \text{ and } v \text{ are neighbors.}\}$. To find a good clique, the local search usually moves from one clique to another until the cutoff of time is reached, then it returns the best clique that has been found. There are three operators, `add`, `swap` and `drop`, which guide the search to move in the clique space. To ensure that these operators preserve the clique property of a solution, [14] define the sets S_{add} and S_{swap} as follows.

$$S_{add} = \begin{cases} \{v | v \notin C, v \in N(u) \text{ for all } u \in C\} & \text{if } |C| > 0; \\ \emptyset, \text{ otherwise.} \end{cases} \tag{2}$$

$$S_{swap} = \begin{cases} \{(u, v) | u \in C, v \notin C, \{u, v\} \notin E, \\ v \in N(w) \text{ for all } w \in C \backslash \{u\}\} & \text{if } |C| > 1; \\ \emptyset, \text{ otherwise.} \end{cases} \tag{3}$$

Besides, given a candidate clique C, a vertex v has two possible states: inside C or outside C. We use $age(v)$ to denote the number of steps since the last time that v changed its state.

C. The Strong Configuration Checking Strategy

The main idea of the SCC [8] strategy is described as follows. After a vertex v is dropped from or swapped from a clique C, it can be added back or swapped back into C only if one of its neighbors is added into C. More specifically the SCC strategy specifies the following rules:

1) Initially $confChange(v)$ is set to 1 for each vertex v;
2) When v is added, $confChange(n)$ is set to 1 for all $n \in N(v)$;
3) When v is dropped, $confChange(v)$ is set to 0;
4) When $(u, v) \in S_{swap}$ are swapped, $confChange(u)$ is set to 0.

D. Detecting the Revisiting

We use a hash table to approximately detect the revisiting of a solution. Given a clique C and a prime number p, we define the hash value of C, denoted by $hash(C)$, as $(\sum_{v_i \in C} 2^i) \bmod p$, which maps a clique C to its hash entry $hash(C)$. We do not resolve the problem of collisions since p is set sufficiently large. If we set p to a larger prime number, the chance of collision will decrease. So the parameter is set simply according to the memory capacity of the machine. Throughout this paper, we use $p = 10^9 + 7$ and the hash table will consume about 1G memory.

III. A Novel Scoring Function

Let S be a set of vertices (S does not need to be a clique), then we define $w(S)$ as

$$\sum_{v_i \in S} w_{\mathcal{V}}(v_i) + \sum_{v_i, v_j \in S \text{ and } \{v_i, v_j\} \in E} w_{\mathcal{E}}(\{v_i, v_j\}). \tag{4}$$

We use $score(v, S)$ to denote the increase of $w(S)$ when v is added into or dropped from S as below

$$score(v, S) = \begin{cases} w(S \cup \{v\}) - w(S) & \text{if } v \notin S; \\ w(S \backslash \{v\}) - w(S) & \text{if } v \in S. \end{cases} \tag{5}$$

Then we use $score(u, v, S)$ to denote the increase of $w(S)$ when u and v are swapped, that is,

$$score(u, v, S) = w(S \backslash \{u\} \cup \{v\}) - w(S), \tag{6}$$

where $u \in S$, $v \notin S$ and $\{u, v\} \notin E$. Note that the value of the score could be negative. In our solver we use these scores to measure the benefits of a local move. Then we maintain the respective scores by Proposition 1.

Proposition 1:

1) $score(u, \emptyset) = w_{\mathcal{V}}(u)$ for all $u \in V$;
2) $score(v, S \backslash \{w\}) = score(v, S) + w_{\mathcal{E}}(\{v, w\})$ for all $v \in (N(w) \cap S)$;
3) $score(v, S \cup \{w\}) = score(v, S) - w_{\mathcal{E}}(\{v, w\})$ for all $v \in (N(w) \cap S)$;
4) $score(v, S \backslash \{w\}) = score(v, S) - w_{\mathcal{E}}(\{v, w\})$ for all $v \in (N(w) \backslash S)$;
5) $score(v, S \cup \{w\}) = score(v, S) + w_{\mathcal{E}}(\{v, w\})$ for all $v \in (N(w) \backslash S)$.

Finally, we compute the score of swapping vertices as below.

Proposition 2: $score(u, v, S) = score(u, S) + score(v, S)$.

In methods such as MN/TS [7], LSCC [8] and LMY-GRS [14], they use Δ_{add}, Δ_{swap} and Δ_{drop} to measure the benefits of the respective operations in local search for maximum vertex weight cliques. Here we have the following proposition which shows that our scoring function is a more general version.

Proposition 3: If $w_{\mathcal{E}}(e) = 0$ for all $e \in E$, then

1) $\Delta_{add}(v, C) = score(v, C)$;
2) $\Delta_{swap}(u, v, C) = score(u, v, C)$;
3) $\Delta_{drop}(u, C) = score(u, C)$.

Although LSCC and LMY-GRS are dedicated to find MWC on vertex-weighted graphs, we will adapt them to deal with the edge-weighted graphs. More specifically we will replace Δ_{add}, Δ_{swap} and Δ_{drop} in them with the $score$ versions.

IV. A Cycle Estimation based Restart Method

We develop a Cycle Estimation based ReStart (CERS) strategy. Each time we obtain a local optimal solution C, we will mark the hash entry for C, *i.e.*, $hash(C)$ (see Line 11 in Algorithm 2). Then we check if the entry $hash(C)$ has been remarked (See Line 12 in Algorithm 2). If so we will drop all the vertices in C.

The top level algorithm is shown in Algorithm 1, where the `localMove()` procedure is shown in Algorithm 2. For simplicity, in Algorithm 2 we write $score(v)$ in short for $score(v, C)$, and $score(u, v)$ in short for $score(u, v, C)$. All ties are broken in favor of the oldest one like LSCC.

Algorithm 1: CERS

input : A graph $G = (V, E, w_{\mathcal{V}}, w_{\mathcal{E}})$ and the *cutoff*
output: The best clique that was found

$C^* \leftarrow C \leftarrow \emptyset$; $lastStepImproved \leftarrow$ *true*;
$step \leftarrow 1$; $confChange(v) \leftarrow 1$ for all $v \in V$;
while *elapsed time* $<$ *cutoff* **do** localMove() ;
return C^*;

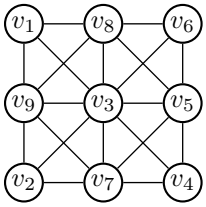
Algorithm 2: localMove

if $C = \emptyset$ **then** add a random vertex into C ;
$v \leftarrow$ a vertex in S_{add} s.t. $confChange(v) = 1$ with the biggest $score(v)$; otherwise $v \leftarrow$ NL;
$(u, u') \leftarrow$ a pair in S_{swap} s.t. $confChange(u') = 1$ with the biggest $score(u, u')$; otherwise $(u, u') \leftarrow$ (NL, NL);
if $v \neq$ *NL* **then**
    if $(u, u') =$ (NL, NL) *or* $score(v) > score(u, u')$ **then** $C \leftarrow C \cup \{v\}$; **else** $C \leftarrow C \backslash \{u\} \cup \{u'\}$;
    $lastStepImproved \leftarrow$ *true*;
else
    if $(u, u') = (NL, NL)$ *or* $score(u, u') < 0$ **then**
        if $lastStepImproved =$ true **then**
            if $w(C) > w(C^*)$ **then** $C^* \leftarrow C$;
            mark $hash(C)$;
            if $hash(C)$ *is remarked* **then**
                drop all vertices in C; **return**
        $lastStepImproved \leftarrow$ *false*;
    else
        $lastStepImproved \leftarrow$ *true*;
    $x \leftarrow$ a vertex in C with the biggest $score$;
    if $(u, u') =$ (NL, NL) *or* $score(x) > score(u, u')$ **then** $C \leftarrow C \backslash \{x\}$; **else** $C \leftarrow C \backslash \{u\} \cup \{u'\}$;
apply SCC rules; $step$++;

V. The Cycling Phenomena

Now we show that without the restart strategy, LSCC, LSCC+BMS, LMY-GRS and CERS (Algorithm 1) may all fall into a dead loop, *i.e.*, no matter how many steps they performs, they may always miss the optimal solution (See Example 1).

Example 1: Consider the graph G, where $w_{\mathcal{V}}(v_i) = i \cdot 10$ for any $i \neq 3$, $w_{\mathcal{V}}(v_3) = 3$ and $w_{\mathcal{E}}(e) = 0$ for all $e \in E$. Obviously the optimal solution in G is $\{v_3, v_5, v_6, v_8\}$.



G

1) Initially $C = \emptyset$. Suppose that we select v_1 as the first vertex and put it into C, then all the four solvers, *i.e.* LSCC, LSCC+BMS, LMY-GRS and CERS, will obtain a clique $C = \{v_1, v_3, v_8, v_9\}$. Then $S_{add} = S_{swap} = \emptyset$. Meanwhile $confChange(v) = 1$ for all $v \in V$.
2) We perform $\texttt{drop}(v_3)$ since it causes the minimum loss. Then C becomes $\{v_1, v_8, v_9\}$. Meanwhile $S_{add} = \{v_3\}$ and $S_{swap} = \emptyset$.
3) Considering that $confChange(v_3) = 0$, we perform $\texttt{drop}(v_1)$. Then C becomes $\{v_8, v_9\}$. Meanwhile $S_{add} = \{v_1, v_3\}$ and $S_{swap} = \{(v_9, v_5), (v_9, v_6), (v_8, v_2), (v_8, v_7)\}$.
4) Since $confChange(v_1) = confChange(v_3) = 0$, we perform $\texttt{swap}(v_8, v_7)$ which causes the minimum loss. Then C becomes $\{v_7, v_9\}$. Meanwhile $S_{add} = \{v_2, v_3\}$ and $S_{swap} = \{(v_7, v_8), (v_7, v_1), (v_9, v_4), (v_9, v_5)\}$.
5) We perform $\texttt{add}(v_2)$. Then $confChange(v_3)$ becomes 1.
6) We perform $\texttt{add}(v_3)$. Then $confChange(v_1)$ and $confChange(v_8)$ become 1.
7) Similar to the previous situation, we first perform $\texttt{drop}(v_3)$, and then perform $\texttt{drop}(v_2)$. Then C becomes $\{v_7, v_9\}$. At the same time, $confChange(v_2)$ and $confChange(v_3)$ become 0.
   Notice that $\{v_7, v_9\}$ is revisited. More specifically after $\texttt{add}(v_2)$, $\texttt{add}(v_3)$, $\texttt{drop}(v_3)$ and $\texttt{drop}(v_2)$, the local search returns to a visited solution.
8) Considering that $confChange(v_8) = 1$, we perform $\texttt{swap}(v_7, v_8)$ which causes the greatest increase of the clique weight.
9) Similar to the previous situation, we first perform $\texttt{add}(v_1)$, and then perform $\texttt{add}(v_3)$. Then C becomes $\{v_1, v_3, v_8, v_9\}$. Also $confChange(v_3)$, $confChange(v_2)$ and $confChange(v_7)$ become 1.
   Notice that the clique $\{v_1, v_3, v_8, v_9\}$ is revisited.

Then the local search repeat the steps above, and is restricted in a cycle. No matter how many steps it performs, it is unable to locate the optimal solution.

We note that [20] proposed the notions of *probabilistic asymptotic complete* and *essentially incomplete*, and show similar phenomena in local search for propositional satisfiability. Informally speaking, an algorithm A is said to be probabilistic asymptotic complete (PAC), if for any graph instance $G = (V, E, w_{\mathcal{V}}, w_{\mathcal{E}})$, the probability that A finds an optimal solution to G can be arbitrarily close to 1 when runtime increases. Otherwise we say that A is essentially incomplete. In this sense, given sufficient time, a PAC algorithm will find an optimal solution almost certainly. Furthermore to prove that an algorithm is essentially incomplete, it is sufficient to find a graph and a reachable state of the algorithm from which no optimal solutions can be attained (See Example 1). Then we have

Theorem 4: Without restarts, CERS, LSCC, LSCC+BMS and LMY-GRS are all essentially incomplete.

Notice that in Example 1, the local search traverses the dead loop in no more than 10 steps. However, LSCC restarts every 4,000 steps due to its default parameter setting, so it might

waste most of the time repeating a sequence of steps. With the cycle estimation based restart strategy, CERS will restart within 20 steps, so it is much more efficient in this example.

VI. Evaluations on Large Sparse Graphs

A. Experimental Setup

For LSCC [8], LSCC+BMS [8] and LMY-GRS [14], the search depth L was set to 4,000 as is in the respective papers. LSCC+BMS further employs the BMS heuristic, and the parameter k was set to 100, as in [8]. For FastWClq [9], the parameters k_0 and k_{max} for the dynamic BMS heuristic were set to 4 and 64 respectively, as in [9]. WLMC [3] was compiled with gcc 4.7.3, while all other solvers were compiled by g++ 4.7.3. The experiments were conducted on a cluster with Intel Xeon E5-2670 v3 2.3GHz with 32GB RAM on CentOS6.

Since WLMC is an exact MWC solver, it was executed only once on each graph instance. The other solvers are randomization based solvers, thus each of them was executed on each instance with seeds from 1 to 10. The cutoff was set to 1,000s for all solvers on all instances. For each algorithm (except WLMC) on each instance, we report the maximum weight ("w_{max}") and averaged weight ("w_{avg}") of the returned cliques. FastWClq and WLMC can sometimes confirm the optimality of the returned solution, and we mark $*$ in the respective table entry.

B. Vertex-weighted Graphs

Table I shows the performances of state-of-the-art MWC solvers on vertex-weighted graphs. Furthermore, we also include FastWClq and WLMC in our experiments. For the sake of space, we do not report results on those graphs where all solvers found the same w_{max} and w_{avg}.

From Table I we find that

1) CERS[1] outperforms LSCC+BMS and LMY-GRS;
2) it is comparable with FastWClq and WLMC.

TABLE I
Experimental results on large sparse vertex-weighted graphs where the five algorithms find different w_{max} or w_{avg} values.

Graph	FastWClq $w_{max}(w_{avg})$	WLMC	LSCC+BMS $w_{max}(w_{avg})$	LMY-GRS $w_{max}(w_{avg})$	CERS $w_{max}(w_{avg})$
ca-coauthors-dblp	37884*(**37884.00**)	37884*	37884(36142.30)	37884(**37884.00**)	37884(**37884.00**)
ca-dblp-2010	7575*(**7575.00**)	7575*	7575(7467.90)	7575(**7575.00**)	7575(**7575.00**)
ca-dblp-2012	14108*(**14108.00**)	14108*	14108(13390.10)	14108(**14108.00**)	14108(**14108.00**)
ca-hollywood-2009	222720*(**222720.00**)	222720*	222720(213219.80)	222720(199902.80)	222720(199902.80)
ca-MathSciNet	2792*(**2792.00**)	2792*	2611(2533.80)	2792(**2792.00**)	2792(**2792.00**)
inf-roadNet-CA	752*(**752.00**)	752*	668(632.90)	752(**752.00**)	752(**752.00**)
inf-roadNet-PA	669(**669.00**)	669*	599(599.00)	669(**669.00**)	669(**669.00**)
inf-road-usa	766*(**766.00**)	766*	598(596.00)	766(**766.00**)	766(**766.00**)
sc-ldoor	4081(**4081.00**)	4081*	4060(3987.20)	4081(**4081.00**)	4081(**4081.00**)
sc-msdoor	4088(**4088.00**)	4088*	4088(4059.30)	4088(**4088.00**)	4088(**4088.00**)
sc-pkustk11	5298(**5298.00**)	5298*	5298(5194.50)	5298(5277.30)	5298(**5298.00**)
sc-pkustk13	**6306***(6306.00)	6306*	**6306**(5937.00)	5928(5922.30)	5928(5928.00)
sc-pwtk	4620(**4620.00**)	4620*	4620(4603.20)	4620(**4620.00**)	4620(**4620.00**)
sc-shipsec1	3540*(**3540.00**)	3540*	3540(3328.70)	3540(**3540.00**)	3540(**3540.00**)
sc-shipsec5	4524*(**4524.00**)	4524*	4524(4423.20)	4524(**4524.00**)	4524(**4524.00**)
soc-BlogCatalog	4803(4795.80)	4803*	4803(**4803.00**)	4803(4790.40)	4803(**4803.00**)
soc-brightkite	3672(**3672.00**)	3672*	3672(3658.8)0	3672(3646.80)	3672(**3672.00**)
soc-delicious	1547(**1547.00**)	1547*	1547(1545.60)	1547(1545.60)	1547(**1547.00**)
soc-digg	5303(**5303.00**)	5303*	4675(4675.00)	5303(4800.60)	5303(**5303.00**)
soc-flickr	7083(6986.80)	7083*	7083(**7083.00**)	7083(**7083.00**)	7083(**7083.00**)
soc-flixster	3805(**3805.00**)	3805*	3805(3652.20)	3805(**3805.00**)	3805(**3805.00**)
soc-FourSquare	3064(**3064.00**)	3064*	3064(3051.60)	3038(2967.20)	3064(3043.20)
soc-gowalla	2335(**2335.00**)	2335*	2335(2263.00)	2335(2232.50)	2335(**2335.00**)
soc-livejournal	21368*(**21368.00**)	21368*	15599(3806.80)	21368(16196.40)	21368(**21368.00**)
soc-orkut	5452(**5452.00**)	5452*	5452(3424.10)	5452(3800.10)	5452(5204.10)
soc-pokec	3191(**3191.00**)	3191*	2341(2145.10)	3191(3108.70)	3191(**3191.00**)
socfb-A-anon	2872(**2872.00**)	2872*	2602(2310.90)	2872(**2872.00**)	2872(**2872.00**)
socfb-B-anon	2662(**2662.00**)	2662*	2513(2032.40)	2620(2544.50)	2662(**2662.00**)
socfb-OR	3523(**3523.00**)	3523*	3523(3512.40)	3523(**3523.00**)	3523(**3523.00**)
socfb-uci-uni	1045(**1045.00**)	1045*	838(699.80)	1045(**1045.00**)	1045(**1045.00**)
tech-as-skitter	5703(**5703.00**)	5703*	5703(5487.90)	5703(5307.10)	5703(**5703.00**)
web-wikipedia2009	3891(**3891.00**)	3891*	3455(2146.80)	3891(3726.20)	3891(**3891.00**)

C. Edge-weighted Graphs

Table II shows the performances of state-of-the-art MWC solvers on edge-weighted graphs. For the sake of space, we do not report results on those graphs where all solvers found the same w_{max} and w_{avg}.

From Table II, we find that

1) CERS[2] greatly outperforms all other solvers;
2) it reports a new best-known solution on `soc-orkut`.

VII. Evaluations on Large Dense Graphs

A. Image Co-localization as an MWC Problem

We evaluate our solver on the large dense graphs in the practice of image co-localization. Given a set of images, the task of image co-localization [17] is to simultaneously localize objects of the same class in each image. Co-localizing objects in unconstrained environment can be challenging. The objects from the same class may look very different due to viewpoint, occlusion, pose, non-rigid deformation, etc. Besides, there could be multiple common objects in the given set of images, thus the definition of "common" may be ambiguous.

To achieve robust and efficient object co-localization, we formulate our task as an MWC problem. It is intended to find a group of objects sharing maximal mutual affinity across the images. The RPN [27] is used for generating object candidates in each image. Each object candidate is specified by the coordinates of a bounding box. We select 20 candidates per image with the highest RPN objectness scores. For each candidate, the vector of class distribution output by RPN is used as the descriptor and the dot-product is used to measure the similarity between object candidates. Then, an undirected weighted graph $G = (V, E)$ is constructed, where each node v_i corresponds to an object candidate and the weight $A(i,j) = w_{\mathcal{E}}(\{v_i, v_j\})$ is the similarity between the objects candidates v_i and v_j. We set the weight of each node v_i to zero, namely $A(i,i) = w_{\mathcal{V}}(v_i) = 0, \forall v_i \in V$. To ensure that at most one candidate object is selected from each image, we remove the edge between v_i and v_j if two object candidates v_i and v_j are in the same image. Hence they cannot appear in the same maximum weight clique together. In this way, the image co-localization problem is formulated as an MWC problem and it can be solved by the proposed CERS algorithm.

[1]https://github.com/Fan-Yi/Efficient-Local-Search-for-Maximum-Weight-Cliques-with-Applications-in-Image-Co-localization

[2]https://github.com/Fan-Yi/Local-Search-for-Maximum-Edge-Weight-Clique

TABLE II
EXPERIMENTAL RESULTS ON LARGE SPARSE EDGE-WEIGHTED GRAPHS WHERE THE FOUR ALGORITHMS FIND DIFFERENT w_{max} OR w_{avg} VALUES.

Graph	LSCC $w_{max}(w_{avg})$	LSCC+BMS $w_{max}(w_{avg})$	LMY-GRS $w_{max}(w_{avg})$	CERS $w_{max}(w_{avg})$
ca-citeseer	451135(254584.1)	451135(322432.0)	451135(**451135.0**)	451135(**451135.0**)
ca-coauthors-dblp	5661008(5359200.8)	5661008(5109263.7)	5661008(**5661008.0**)	5661008(**5661008.0**)
ca-dblp-2010	276575(221523.5)	276575(206365.0)	276575(**276575.0**)	276575(**276575.0**)
ca-hollywood-2009	245095624(229025126.6)	245095624(**245095624.0**)	245095624(166832908.6)	245095624(207469298.4)
ca-MathSciNet	32364(20300.9)	25393(18195.9)	32364(**32364.0**)	32364(**32364.0**)
inf-roadNet-CA	597(597.0)	597(597.0)	1050(**1050.0**)	1050(**1050.0**)
inf-roadNet-PA	993(993.0)	993(993.0)	1164(**1164.0**)	1164(**1164.0**)
inf-road-usa	567(567.0)	567(558.0)	1092(**1092.0**)	1092(**1092.0**)
rec-amazon	1686(1531.6)	1686(1566.4)	1866(**1866.0**)	1866(**1866.0**)
rt-retweet-crawl	4020(4020.0)	8262(6989.4)	8262(7413.6)	8262(**8262.0**)
sc-ldoor	38990(37260.0)	39570(37846.0)	40610(**40610.0**)	40610(**40610.0**)
sc-msdoor	39550(38236.0)	39550(38314.0)	40250(**40250.0**)	40250(**40250.0**)
sc-nasasrb	51040(50657.6)	51040(50657.6)	51040(**51040.0**)	51040(**51040.0**)
sc-pkustk11	77580(69262.0)	77580(73692.0)	77580(**77580.0**)	77580(**77580.0**)
sc-pkustk13	99915(95501.5)	94080(93468.0)	99915(94393.5)	99915(**99915.0**)
sc-pwtk	51888(50281.2)	51888(50522.8)	51888(**51888.0**)	51888(**51888.0**)
sc-shipsec1	45126(33624.7)	45126(36190.1)	45126(**45126.0**)	45126(**45126.0**)
sc-shipsec5	48576(47223.6)	48576(47335.4)	48576(**48576.0**)	48576(**48576.0**)
soc-digg	123757(94136.7)	123757(102589.9)	123757(111055.5)	123757(**123738.4**)
soc-flixster	47685(**47685.0**)	47685(**47685.0**)	47685(47677.0)	47685(47677.0)
soc-FourSquare	45982(**45982.0**)	45982(**45982.0**)	45982(43121.2)	45982(45982.0)
soc-gowalla	30226(24149.2)	22630(22630.0)	22630(22590.0)	30226(**30226.0**)
soc-lastfm	11266(10412.7)	10047(10047.0)	11266(10887.3)	11266(**11266.0**)
soc-livejournal	81460(28406.8)	81460(32161.4)	2289993(1222371.1)	2289993(**2289993.0**)
soc-orkut	96682(45829.9)	74911(46446.7)	72188(50334.5)	**105549**(95773.3)
soc-pokec	25603(17587.3)	19959(17794.4)	38202(25734.1)	38202(**38202.0**)
socfb-A-anon	25615(20483.1)	32532(20994.7)	32532(28129.3)	32532(**32532.0**)
socfb-B-anon	22441(18330.7)	22441(18198.5)	28384(23161.7)	28384(**28384.0**)
socfb-MIT	54696(**54696.0**)	54696(**54696.0**)	54696(54677.6)	54696(**54696.0**)
socfb-Penn94	93079(**93079.0**)	93079(**93079.0**)	93079(92045.1)	93079(**93079.0**)
socfb-Stanford3	131675(**131675.0**)	131675(**131675.0**)	131675(**131675.0**)	131675(**131675.0**)
socfb-UF	149419(149419.0)	149419(149419.0)	149419(148600.5)	149419(149419.0)
socfb-uci-uni	1362(1031.0)	2210(1184.9)	2210(**2210.0**)	2210(**2210.0**)
tech-as-skitter	179915(158512.6)	179915(162907.7)	179915(166687.1)	179915(**179915.0**)
tech-p2p-gnutella	903(886.5)	903(888.9)	903(**903.0**)	903(**903.0**)
web-it-2004	8035767(4233906.9)	9282115(5332859.0)	9308691(**9308691.0**)	9308691(**9308691.0**)
web-sk-2005	401124(339784.8)	401124(390852.0)	401124(**401124.0**)	401124(**401124.0**)
web-uk-2005	12572200(12559868.4)	12572200(12146305.8)	12572200(**12572200.0**)	12572200(**12572200.0**)
web-wikipedia2009	46785(29626.8)	46785(32679.2)	46832(**46832.0**)	46832(**46832.0**)

B. The Image Dataset and Evaluation Protocol

To evaluate the performance of our method in comparison to other approaches, experiments are conducted on the PASCAL VOC 2007 image dataset [19]. We follow [17] to construct a test set for image co-localization and denote it as the PASCAL07 dataset. The same evaluation protocal is used to measure the co-localization accuracy. The RPN involved in our method is pre-trained independently on the PASCAL 2012 training set, thus the training data is not correlated with the test data. The systsm is tested on a desktop machine with Intel(R) Xeon(R) CPU E5-2637 v4 3.50Hz and 64 GB memory.

C. Comparison with Different Methods

We first compare our method to different MWC solvers. For fair comparison, we apply different MWC solvers on the same graphs constructed from the same problem instances. The average number of nodes in the constructed graphs is 6081.67, and the average number of edges is 2.16×10^7. The average density of the graphs is 0.9962 and the average degree of the nodes is 6061.58. So the graphs involved in our experiments are large dense graphs. For those randomized solvers, we run the evaluation 10 times with different seeds for the random number generator, and we report the average accuracy over these 10 runs. We set the total number of iterations of CMWC [16] and TBMA [15] to 500, thus the run-time of the two solvers is similar to ours. We use the same cutoff time (60 seconds) for [8], [14] and our solver. The detailed results are reported in Table III.

Besides, we report the results of some state-of-the-art localization methods based on Weakly Supervised Learning (WSL). Such WSL methods exploit additional information from the negative images, which is not used in our approach. The corresponding results are reported in Table IV. The co-localization accuracies of related methods are directly taken from related literatures. The results show that our method outperforms the existing methods for co-localization and is comparable to the state-of-the-art WSL methods.

To conclude, our CERS based solver demonstrates state-of-the-art performance in the application of image co-localization. This validates its effectiveness of searching MWC in large dense graphs.

VIII. CONCLUSIONS

In this paper, we develop a local search based algorithm that solves the MWC problem on large graphs. We demonstrate the benefits of the proposed algorithm in the large sparse vertex-weighted and edge-weighted graphs, as well as in the application of image co-localization where large dense graphs are involved. Our method shows superior performance over other state-of-the-art MWC solvers. We also prove a theorem which shows the necessity of the restart strategies in current state-of-the-art local search algorithms.

IX. ACKNOWLEDGMENTS

We thank all anonymous reviewers for their valuable comments. This work is supported by ARC Grant FT0991785, NSF Grant No. 61463044, NSFC Grant No. 61572234, Grant No. [2014]7421 from the Joint Fund of the NSF of Guizhou province of China, and the US NSF under Grant IIS-1302164.

We gratefully acknowledge the support of the Griffith University eResearch Services Team and the use of the High Performance Computing Cluster "Gowonda" to complete this research. This research was also supported by use of the NeCTAR Research Cloud and by QCIF (http://www.qcif.edu.au). The NeCTAR Research Cloud is a collaborative Australian research platform supported by the National Collaborative Research Infrastructure Strategy.

REFERENCES

[1] K. Yamaguchi and S. Masuda, "A new exact algorithm for the maximum weight clique problem," *ITC-CSCC: 2008*, pp. 317–320, 2008.

[2] Z. Fang, C. Li, and K. Xu, "An exact algorithm based on maxsat reasoning for the maximum weight clique problem," *J. Artif. Intell. Res. (JAIR)*, vol. 55, pp. 799–833, 2016. [Online]. Available: http://dx.doi.org/10.1613/jair.4953

[3] H. Jiang, C. Li, and F. Manyà, "An exact algorithm for the maximum weight clique problem in large graphs," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 830–838. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14370

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	MBike	Person	Plant	Sheep	Sofa	Train	TV	Avg
CMWC [16]	36.9	27.8	34.6	33.8	12.3	35.8	27.7	57.2	21.2	34.0	36.7	40.8	47.8	36.9	21.7	21.6	20.2	43.5	55.2	33.6	34.0
TBMA [15]	69.0	55.0	56.6	47.6	0.2	62.9	51.5	79.3	31.5	45.9	0.5	68.4	68.2	63.7	37.7	28.6	33.2	51.6	76.4	62.0	49.5
LSCC [8]	71.7	45.3	59.1	47.8	22.4	66.4	**52.7**	**82.7**	37.3	**48.7**	53.2	**74.5**	72.0	66.6	**38.5**	29.6	33.2	**67.7**	**79.1**	62.9	55.6
LSCC + BMS [8]	**71.9**	41.7	**59.3**	47.9	25.7	66.4	**52.7**	**82.7**	37.7	48.6	53.2	**74.5**	**72.1**	**66.7**	**38.5**	**29.7**	**33.4**	**67.7**	**79.1**	**63.0**	55.6
LMY-GRS [14]	**71.9**	**56.1**	**59.3**	**48.3**	**32.5**	**67.2**	**52.7**	**82.7**	**37.8**	48.6	**64.7**	74.3	**72.1**	**66.7**	**38.5**	29.6	33.1	67.3	78.8	**63.0**	**57.3**
CERS	**71.9**	**56.1**	**59.3**	**48.3**	**32.4**	66.9	**52.7**	**82.7**	**37.8**	48.6	**64.7**	73.9	**72.1**	**66.7**	38.2	29.6	33.3	67.3	78.8	**63.0**	57.2

TABLE III
IMAGE CO-LOCALIZATION ACCURACY (%) OF DIFFERENT MWC ALGORITHMS.

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	MBike	Person	Plant	Sheep	Sofa	Train	TV	Avg
[17]	73.1	45.0	43.4	27.7	6.8	53.3	58.3	45.0	6.2	48.0	14.3	47.3	69.4	66.8	24.3	12.8	**51.5**	25.5	65.2	16.8	40.0
[21]	37.7	58.8	39.0	4.7	4.0	48.4	70.0	63.7	9.0	54.2	33.3	37.4	61.6	57.6	30.1	31.7	32.4	52.8	49.0	27.8	40.2
[22]	66.4	59.3	42.7	20.4	21.3	63.4	74.3	59.6	21.1	58.2	14.0	38.5	49.5	60.0	19.8	39.2	41.7	30.1	50.2	44.1	43.7
[23]	**79.2**	56.9	46.0	12.2	15.7	58.4	71.4	48.6	7.2	**69.9**	16.7	47.4	44.2	75.5	**41.2**	39.6	47.4	32.2	49.8	18.6	43.9
[24]	67.1	66.1	49.8	34.5	23.3	68.9	**83.5**	44.1	27.7	71.8	49.0	48.0	65.2	79.3	37.4	42.9	65.2	51.9	62.8	46.2	54.2
[22]	68.9	**68.7**	**65.2**	42.5	**40.6**	**72.6**	75.2	53.7	29.7	68.1	33.5	45.6	65.9	**86.1**	27.5	**44.9**	**76.0**	62.4	66.3	**66.8**	58.0
[25]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**60.9**
[26]	67.3	63.3	61.3	22.7	8.5	64.8	57.0	80.5	9.4	49.0	22.5	72.6	73.8	69.0	7.2	15.0	35.3	54.7	75.0	29.4	46.9
CERS	71.9	56.1	59.3	**48.3**	32.4	66.9	52.7	**82.7**	**37.8**	48.6	**64.7**	**73.9**	**72.1**	66.7	38.2	29.6	33.3	**67.3**	**78.8**	63.0	57.2

TABLE IV
IMAGE CO-LOCALIZATION ACCURACY (%) OF DIFFERENT WSL METHODS ON THE PASCAL07 DATASET. COMPARED TO THESE WSL METHODS, OUR METHOD DOES NOT EXPLOIT EXTRA INFORMATION FROM THE NEGATIVE IMAGES.

[4] B. Alidaee, F. Glover, G. Kochenberger, and H. Wang, "Solving the maximum edge weight clique problem via unconstrained quadratic programming," *European Journal of Operational Research*, vol. 181, no. 2, pp. 592 – 597, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706004759

[5] L. Gouveia and P. Martins, "Solving the maximum edge-weight clique problem in sparse graphs with compact formulations," *EURO Journal on Computational Optimization*, vol. 3, no. 1, pp. 1–30, 2015. [Online]. Available: http://dx.doi.org/10.1007/s13675-014-0028-1

[6] W. J. Pullan, "Approximating the maximum vertex/edge weighted clique using local search," *J. Heuristics*, vol. 14, no. 2, pp. 117–134, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10732-007-9026-2

[7] Q. Wu, J. Hao, and F. Glover, "Multi-neighborhood tabu search for the maximum weight clique problem," *Annals OR*, vol. 196, no. 1, pp. 611–634, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10479-012-1124-3

[8] Y. Wang, S. Cai, and M. Yin, "Two efficient local search algorithms for maximum weight clique problem," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 2016, pp. 805–811. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11915

[9] S. Cai and J. Lin, "Fast solving maximum weight clique problem in massive graphs," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 568–574. [Online]. Available: http://www.ijcai.org/Abstract/16/087

[10] F. Mascia, E. Cilia, M. Brunato, and A. Passerini, "Predicting structural and functional sites in proteins by searching for maximum-weight cliques," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1833

[11] S. J. Shyu, P. Yin, and B. M. T. Lin, "An ant colony optimization algorithm for the minimum weight vertex cover problem," *Annals OR*, vol. 131, no. 1-4, pp. 283–304, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:ANOR.0000039523.95673.33

[12] W. Pullan, "Optimisation of unweighted/weighted maximum independent sets and minimum vertex covers," *Discrete Optimization*, vol. 6, no. 2, pp. 214–219, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.disopt.2008.12.001

[13] S. Cai, K. Su, and A. Sattar, "Local search with edge weighting and configuration checking heuristics for minimum vertex cover," *Artif. Intell.*, vol. 175, no. 9-10, pp. 1672–1696, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2011.03.003

[14] Y. Fan, C. Li, Z. Ma, A. Sattar, L. Wen, and K. Su, "Local search for maximum vertex weight clique on large sparse graphs with efficient data structures," in *AI 2016: Advances in Artificial Intelligence - 29th Australasian Joint Conference, Hobart, Australia, December 4-8, 2016. Proceedings. To appear.*, 2016.

[15] K. Adamczewski, Y. Suh, and K. M. Lee, "Discrete tabu search for graph matching," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, 2015, pp. 109–117. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.21

[16] T. Ma and L. J. Latecki, "Maximum weight cliques with mutex constraints for video object segmentation," in *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 670–677.

[17] Y. Li, L. Liu, C. Shen, and A. v. d. Hengel, "Image co-localization by mimicking a good detector's confidence score distribution," *European Conference on Computer Vision (ECCV)*, 2016.

[18] S. Eubank, V. S. A. Kumar, M. V. Marathe, A. Srinivasan, and N. Wang, "Structural and algorithmic aspects of massive social networks," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, 2004, pp. 718–727. [Online]. Available: http://dl.acm.org/citation.cfm?id=982792.982902

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results."

[20] H. H. Hoos, "On the run-time behaviour of stochastic local search algorithms for SAT," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA.*, 1999, pp. 661–666. [Online]. Available: http://www.aaai.org/Library/AAAI/1999/aaai99-094.php

[21] X. Wang, Z. Zhu, C. Yao, and X. Bai, "Relaxed multiple-instance svm with application to object discovery," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1224–1232.

[22] H. Bilen, M. Pedersoli, and T. Tuytelaars, "Weakly supervised object detection with convex clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1081–1089.

[23] W. Ren, K. Huang, D. Tao, and T. Tan, "Weakly supervised large scale object localization with multiple instance learning and bag splitting," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 405–416, 2016.

[24] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2016.

[25] M. Shi and V. Ferrari, "Weakly supervised object localization using size estimates," in *European Conference on Computer Vision*. Springer, 2016, pp. 105–121.

[26] X. Wei, C. Zhang, Y. Li, C. Xie, J. Wu, C. Shen, and Z. Zhou, "Deep descriptor transforming for image co-localization," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017, pp. 3048–3054. [Online]. Available: https://doi.org/10.24963/ijcai.2017/425

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.