最大辺重みクリーク問題に対する局所探索法の実験的評価

Some numerical experiments of local search algorithms for the maximum edge weight clique problem

清水 悟司 † 石原 諒大 † 山口　一章 † 増田　澄男 †
Satoshi Shimizu Ryota Ishihara Kazuaki Yamaguchi Sumio Masuda

1. まえがき

無向グラフ $G=(V,E)$ において，V の部分集合 C の任意の 2 頂点間に辺が存在するとき C はクリークと呼ばれる．グラフ $G=(V,E)$ が与えられたとき，G の中で要素数が最大のクリークを見つける問題は最大クリーク問題 (以降，MCP と記す) と呼ばれる．MCP は NP 困難な組合せ最適化問題の中でも重要なものであり，多くの研究が行われている [1].

MCP の一般化として頂点が重みを持つ以下の問題が考えられる．グラフ $G=(V,E)$ と各頂点 v の重み $w(v)$ が与えられたとき，G の中で重みが最大のクリークを見つける問題を最大重みクリーク問題 (以降，MWCP と記す) と呼ぶ．ただし，V の部分集合 V' の重みは $w(V')=\sum_{v\in V'} w(v)$ とする．MWCP に対する効率的な発見的手法として MN/TS[2] や LSCC[3] が提案されている．この二つの手法は局所探索法に基づく．

MCP の別の一般化として G の中で重みが最大のクリークを見つける問題（最大辺重みクリーク問題，以降，MEWCP と記す）も考えられる．ただし，グラフ $G=(V,E)$ の頂点に重みがなくかつ各辺 e に重み $w(e)$ が与えられているとき，V の部分集合 V' の重みを $w(V')=\sum_{u,v\in V'} w((u,v))$ と定める．

筆者らの知る限り，MEWCP については，MCP や MWCP に比べあまり研究が行われておらず，発見的手法は PLS[4] 以外に見当たらなかった．PLS は DIMACS ベンチマーク [5] に対し有効であることは示されているが，MEWCP に対して有効かどうかは確認されていない．

そこで，我々は MEWCP に用いられるように MN/TS, LSCC に変更を加え，PLS と合わせて三つの手法の中で MEWCP に対しどれが有効であるかを調べた．本稿はその比較実験の結果を示すものである．

本稿の構成は以下の通りである．2. で各手法の概要を示す．3. で計算機実験の結果を示す．4. でまとめと今後の課題について述べる．

2. 局所探索法

本稿で比較を行う三つのアルゴリズムはいずれも局所探索法を基にした発見的手法である．それらの中で共通に用いられる用語や記号について以下に定義を示す．

頂点 v に隣接する頂点の集合を $N(v)$ と表す．頂点集合 $S\subseteq V$ に対し，$N(S)=\cup_{v\in S}N(v)$ と定める．局所探索における探索中の暫定的なクリークを C とする．$C_0=N(C)$ と定める．C_1 を以下のように定める．

$$C_1=\{v \mid \exists u,\ v\in N(C\setminus\{u\})\}\setminus(C\cup C_0)$$

すなわち，C_1 は，C からある一つの頂点を取り除けば C_0 に含まれる可能性のある頂点のうち，現時点では C にも C_0 にも含まれないものの集合である．

局所探索法とは，暫定解 C に部分的修正を加えて得られる解の集合（近傍）の中でより良い解を探し新たな C とする処理を繰り返す手法の総称である．三つのアルゴリズムはいずれも C に対する以下の Add, Drop, Swap 操作によって得られる近傍を用いて解を探索する．

- Add : C に頂点 $v\in C_0$ を加える．
- Drop : C から頂点 $v\in C$ を除く．
- Swap : C に頂点 $v\in C_1$ を加え，C から $u\notin N(v)$ なる頂点 u を 取り除く．

以下ではまず PLS について説明し，次に MEWCP に対応できるように MN/TS と LSCC に変更を加えたものについて簡単に説明する．説明の都合上，アルゴリズムの表記が原論文とは異なる形になっている．また，後の比較実験の都合により，各アルゴリズムの計算時間が指定された時間 ($timeLimit$) になった時点で終了するようにしている．

2.1 PLS (Phased Local Search)

PLS は三つのフェーズを切り替えながら局所探索を行う．Algorithm 1 の line1〜line8 に PLS の大枠を示す．解 C の初期値はランダムに選んだ頂点 v 一つからなる集合 $\{v\}$ とする．その後，ランダムフェーズ (Random)，ペナルティフェーズ (Penalty)，次数フェーズ (Degree) と呼ばれる 3 つのフェーズを切り替えながら局所探索を行う．それぞれのフェーズは 50 回，50 回，100 回ずつ繰り返す．

3 つのフェーズの基礎となる関数 $Phase$（Algorithm 1 line9〜line30）について説明する．以下に示す解の改良処理を $iterations$ によって指定された回数だけ行う．2 回目以降の実行ではパラメータ $Perturb$ で指定される方法により新たな初期解を得る（line28）．

† 神戸大学, Kobe University

探索において解の巡回が起きるのを避けるために，C への追加を許さない頂点の集合 U を保持している．その U による制約を維持しつつ，Add が可能なときは可能な限り繰り返し Add を行い (line12 ～ line19)，できないときは Swap 処理を試みる (line20 ～ line24)．以上をいずれの処理もできなくなるまで繰り返す．

探索中，各頂点に対しペナルティという値が設定される．詳細は省略するが，探索中に C に入った回数が多い頂点ほど値が大きくなるように，処理 *UpdatePenalties* (line27) において適宜更新される．ただし，この値を用いて探索の方針を決めるのは *Penalty* フェーズのみであり，他のフェーズにおいては参照されない．

以下に 3 つのフェーズの *Selection* や *Perturb* の違いを示す．

- Random フェーズ：*RandomSelect* は C に加えられたら $w(C)$ が最も増加する頂点集合の中からランダムに頂点を 1 つ選択する．*Reinitialise* はランダムに頂点を 1 つ選択して，C からその頂点に隣接していない頂点を全て取り除く．
- Penalty フェーズ：*PenaltySelect* は C に加えられたら $w(C)$ が最も増加する頂点集合の中から最もペナルティの低い頂点を 1 つ選択する．*Initialise* はランダムに頂点 v を 1 つ選択して，C を v のみにする．
- Degree フェーズ：*DegreeSelect* は C に加えられたら $w(C)$ が最も増加する頂点集合の中から最も次数の高い頂点を 1 つ選択する．*Reinitialise* は Random フェーズの *Reinitialise* と同じ．

2.2 MN/TS (Multi-neighborhood tabu search)

MN/TS はタブーサーチに基づく解法である．Algorithm 2 に大枠を示す．MN/TS ではタブーリストに頂点を入れておく．そのリストに含まれる頂点は近傍を作る処理では用いられない．近傍の探索中，暫定最良解 $C_{localBest}$ の更新が行われなかった回数 NI (Not Improved) を数える．もし更新が与えられた回数（パラメータ L）行われなければ C を初期化し新たな探索を行う．

初めに *greedyConstruction* により極大クリークを作り C に入れる（line3）．次に，タブーリスト *tabuList* を用いた局所探索を行う（line4 ～line16）．*tabuList* を考慮しながら C から以下に示す三つの近傍 $N_1(C), N_2(C), N_3(C)$ を作る (line8)．ただし $N_1(C)$ は候補集合 $C_0 \setminus tabuList$ から頂点を選び C に Add 操作を行うことで得られるクリークの集合，$N_2(C)$ は候補集合 $C_1 \setminus tabuList$ から頂点を選び C に Swap 操作を行うことで得られるクリークの集合，$N_3(C)$ は C に Drop 操作を行うことで得られるクリークの集合である．これらの近傍の中で最も重みの大きい解 C' を新たな C とする (line9)．この更新により C から除かれた頂点をタブーリストに加える (line11)．もし C の重みが $C_{localBest}$ よりも大きければ $C_{localBest}$ を更新し，NI を 0 にする (line12～line15)．以上の局所探索を暫定最良解 $C_{localBest}$ が L 回更新されなければループを抜け，新たな初期解からの探索を行う．

Algorithm 1 PLS

INPUT: $G = (V, E)$, $w[\cdot]$, $timeLimit$
OUTPUT: $C_{globalBest}$

```
1:  C_globalBest ← ∅, U ← ∅
2:  C ← {a randomly selected vertex fromV}
3:  while time < timeLimit do
4:      Phase(50, RandomSelect, Reinitialise)
5:      Phase(50, PenaltySelect, Initialise)
6:      Phase(100, DegreeSelect, Reinitialise)
7:  end while
8:  return C_globalBest

9:  function Phase(iterations, Select, Perturb)
10:     do
11:         do
12:             while C_0 \ U ≠ ∅ do
13:                 v ← Select(C_0 \ U)
14:                 C ← C ∪ {v}
15:                 U ← ∅
16:             end while
17:             if w(C) > w(C_globalBest) then
18:                 C_globalBest ← C
19:             end if
20:             if C_1 \ U ≠ ∅ then
21:                 v ← Select(C_1 \ U)
22:                 U ← U ∪ (C \ N(v))
23:                 C ← (C ∩ N(v)) ∪ {v}
24:             end if
25:         while C_0 ≠ ∅ or C_1 \ U ≠ ∅
26:         iterations ← iterations − 1
27:         Update Penalties
28:         Perturb
29:     while iterations > 0
30: end function
```

Algorithm 2 MN/TS

INPUT: $G = (V, E)$, $w[\cdot]$, L, $timeLimit$

OUTPUT: $C_{globalBest}$

1: $C_{globalBest} \leftarrow \emptyset$
2: **while** $time < timeLimit$ **do**
3: $Initiate\ C\ greedyConstruction$
4: $Initiate\ tabuList$
5: $NI \leftarrow 0$
6: $C_{localBest} \leftarrow C$
7: **while** $NI < L$ **do**
8: $Construct\ N_1, N_2\ and\ N_3$ from C and $tabuList$
9: $C \leftarrow Select\ best\ neighbor\ C' \in N_1 \cup N_2 \cup N_3$
10: $NI \leftarrow NI + 1$
11: $Update\ tabuList$
12: **if** $W(C) > W(C_{localBest})$ **then**
13: $NI \leftarrow 0$
14: $C_{localBest} \leftarrow C$
15: **end if**
16: **end while**
17: **if** $W(C_{localBest} > W(C_{globalBest})$ **then**
18: $C_{globalBest} \leftarrow C_{localBest}$
19: **end if**
20: **end while**
21: **return** $C_{globalBest}$

MN/TS は MWCP に対する手法であるが，重みの計算を頂点重みから辺重みに変えるだけでそのまま MEWCP に用いることができる．本手法は他の手法に比べ近傍が広いため，単位時間当たりの解空間の探索量は他の手法に比べ少ない．

2.3 LSCC (Local search with scc)

LSCC は scc(strong configuration checking) を利用した局所探索である．scc は cc(configuration checking)[6] を基にしているが，以下では cc に関する説明は省略し，LSCC における scc についてのみ説明する．

scc は各頂点 v に対し 0 または 1 の値を取る $confChange(v)$ を保持している．$confChange(v)$ が 0 の頂点 v は ADD 操作や SWAP 操作に用いることができない．$confChange$ は Add,Drop,Swap や初期化によってに以下のルールに従い変化する．

- InitialRule : V に含まれる全ての頂点 v に対して $confChange(v) \leftarrow 1$ とする．
- AddRule : 頂点 v が C に加えられる時，$v' \in N(v)$ に対して $confChange(v') \leftarrow 1$ とする．
- DropRule : 頂点 v が C から除かれるとき，$confChange(v) \leftarrow 0$ とする．
- SwapRule : 頂点 v が加えられて頂点 u が除かれるとき，$confChange(u) \leftarrow 0$ とする．

Algorithm 3 に大枠を示す．$\Delta_{add}, \Delta_{swap}, \Delta_{drop}$ はそれぞれ Add，Swap，Drop 操作を行った際の C の重みの変化値である．$C'_0 = \{v \in C_0 \mid confChange(v) = 1\}$，$C'_1 = \{v \in C_1 \mid confChange(v) = 1\}$，とする．

初めに C_0 からランダムな頂点を選びながらグリーディーに C を極大化する (line3)．次に，局所探索による遷移を L 回行う (line6)．

局所探索の手順は以下の通りである．まず，C'_0 の中で最も C の重みを増加させる v_{add} と C'_1 の中で最も C の重みを増加させる v_{swap}, u を求める (line7〜line8)．もし C'_0 が空でないなら，Δ_{add} と Δ_{swap} を比較して，値が大きいほうの操作 (Add or Swap) を C に対して行う (line10〜line14)．もし，C'_0 が空なら，C の中で最も C の重みの減少が小さい v_{drop} を求める (line16)．Δ_{swap} と Δ_{drop} を比較して，値が大きいほうの操作 (Swap or Drop) を C に対して行う (line17〜line21)．最後に前出のルールに従って $confChange(\cdot)$ を更新する (line24)．

LSCC では ADD 操作が可能な場合は DROP 処理による評価値を調べないため，N_1, N_2, N_3 の全てについて評価値を計算する MN/TS に比べ計算量が抑えられる．また，$confChange$ は前出のルールのみによって更新されるので，ループ回数によって変化するタブーリストよりも処理が簡単になっている．よって，遷移 1 回当たりの時間量は PLS よりは多いものの MN/TS よりも少ない．

3. 計算機実験

PLS，MN/TS，LSCC を C++ で実装し比較を行った．実験に使用した計算機の CPU は Intel® Xeon® E5-2650 2.00GHz，メモリは 128GB，OS は Linux2.6 である．使用したコンパイラは g++ 4.9.3 で最適化オプション –O2 を利用した．

3.1 予備実験

まず予備実験として三手法を MCP, MWCP に対し用いた場合の比較を行った．MCP の実験では DIMACS ベンチマークと BHOSLIB ベンチマーク [7] のグラフを用いた．MWCP のベンチマーク問題として適切なものが見当たらなかったので，DIMACS ベンチマークと BHOSLIB ベンチマークのグラフの各頂点 v_i に対して $(i \bmod 200) + 1$ の重みを与えることで問題を作成した．

各問題に対し 10 回試行を行い（1 回当たり 60 秒）比較を行った結果を表 1 に示す．紙面の都合上，全問題に

対し得られた解の評価値の平均値のみを示している．この平均値自体は意味のある値ではないが，他の手法より最適解に近い解を多く出力した手法の値が大きくなるため，相対的な評価に使うのには差支えないと判断した．

表 1 によると，DIMACS には PLS が，BHOSLIB では MN/TS が優れており，LSCC は PLS より劣っているという結果となった．MWCP については，MN/TS が最も良く，次に LSCC で，PLS は他に比べてかなり悪いという結果となった．これらの結果から，MEWCP に対しては MN/TS が最も良いという結果が予想される．

Algorithm 3 LSCC

INPUT: $G = (V, E)$, $w[\cdot]$, L, $timeLimit$
OUTPUT: $C_{globalBest}$

$C_{globalBest} \leftarrow \emptyset$
while $time < timeLimit$ **do**
  $Initiate\ C\ greedyConstruction$
  $step \leftarrow 0$
  $C_{localBest} \leftarrow C$
  while $step < L$ **do**
    $v_{add} \leftarrow$ *select a best vertex in* C'_0
    $v_{swap} \leftarrow$ *select a best vertex in* C'_1, $u \leftarrow$ the vertex in $C \setminus N(v_{swap})$
    if $C'_0 \neq \emptyset$ **then**
      if $\Delta_{add} > \Delta_{swap}$ **then**
        $C \leftarrow C \cup \{v_{add}\}$
      else
        $C \leftarrow C \cup \{v_{swap}\} \setminus \{u\}$
      end if
    else
      $v_{drop} \leftarrow$ *select a best vertex in* C
      if $\Delta_{swap} > \Delta_{drop}$ **then**
        $C \leftarrow C \cup \{v_{swap}\} \setminus \{u\}$
      else
        $C \leftarrow C \setminus \{v_{drop}\}$
      end if
    end if
    $step \leftarrow step + 1$
    $Update\ confChange$
    if $W(C) > W(C_{globalBest})$) **then**
      $C_{globalBest} \leftarrow C$
    end if
  end while
end while
return $C_{globalBest}$

3.2 実験方法

以下，MEWCP に対する実験結果を示す．グラフは以下のようにして作成した．グラフは DIMACS ベンチマークと BHOSLIB ベンチマークのグラフとし，端点 i, j の各辺に対して $(i+j) \bmod 200 + 1$ の重みを与えた．

各インスタンスに対して制限時間 60 秒で，異なる乱数の種で 10 回実行した．各手法のパラメータは，その手法が提案された論文に従って $T_1 = 7$,$L = 4000$ とした．

表の各項目について説明する．$dens$ はグラフの辺密度を表す．W_{best} は全試行において得られたクリークの中で，最も大きい重みの値である．success は 10 回の試行で W_{best} に到達した回数，time は W_{best} に到達した試行のみの平均時間（秒）を示している．

DIMACS, BHOSLIB のインスタンスのうち，三つの手法とも短時間で W_{best} に辿り着いたものに関しては実験結果を省略した．また，記号 $< \epsilon$ は実験環境において計測可能な最短時間 0.01 秒より短い計算時間で W_{best} に到達したことを表している．

3.3 DIMACS

DIMACS の実験結果を表 2 に示す．PLS はグラフのサイズが大きい C や johnson，keller 以外のほとんどのインスタンスにおいて他の手法よりも W_{best} に到達した回数が多い，あるいは短い時間で到達していた．さらに MANN においては唯一，PLS のみが W_{best} に到達できた．一方，グラフのサイズが大きい C や johnson，keller に関しては MN/TS の方が他の手法よりも W_{best} に到達した回数が多い，あるいは短い時間で到達していた．三つの手法に対し重みなしの（つまり DIMACS そのものの）グラフを用いて実験を行ったのとほぼ同じ傾向になった．

3.4 BHOSLIB

BHOSLIB の実験結果を表 3 に示す．多くのインスタンスにおいて MN/TS が他の手法と比べて最も W_{best} に到達した回数が多かった．三つの手法に対し重みなしのグラフを用いて実験を行った場合とほぼ同じ傾向になった．

4. あとがき

MWCP に対する発見的手法である MN/TS, LSCC を MEWCP 用に辺の重みを考慮するなど一部変更を加えて実装し，PLS との比較を行った．DIMACS を基にしたグラフに対しては PLS が有効で，BHOSLIB を基にしたグラフでは MN/TS が有効であった．LSCC は多くのインスタンスにおいて他の手法よりも劣っていた．この結果は MCP と傾向がほぼ同じで，MWCP とは傾向が異なった．

以上のような結果が得られたのは，クリーク中に含まれる辺の本数は頂点数の二乗に比例するため，MEWCP

表 1: comparison for MCP and MWCP

	PLS	MN/TS	LSCC
DIMACS	60.1	59.7	59.8
BHOSLIB	45.2	45.7	44.7
DIMACS (MWCP)	6333.5	6369.5	6368.7
BHOSLIB (MWCP)	4828.1	4877.5	4867.7

表 2: results on DIMACS

				MN/TS		PLS		LSCC	
Instance	$\|V\|$	$dens$	W_{best}	time[s]	success	time[s]	success	time[s]	success
brock200_1	200	0.745	21230	1.59	10	0.02	10	3.16	10
brock200_2	200	0.496	6542	5.77	10	0.01	10	5.10	10
brock200_3	200	0.605	10303	7.89	10	$< \epsilon$	10	6.36	10
brock200_4	200	0.658	13967	$< \epsilon$	10	0.01	10	$< \epsilon$	10
brock400_1	400	0.748	35257	-	0	0.27	10	-	0
brock400_2	400	0.749	40738	16.04	9	0.11	10	23.26	8
brock400_3	400	0.748	46785	4.54	10	0.06	10	19.46	8
brock400_4	400	0.749	54304	0.84	10	0.02	10	0.32	10
brock800_1	800	0.649	25050	0.45	10	1.02	10	0.61	10
brock800_2	800	0.651	27932	-	0	10.21	10	-	0
brock800_3	800	0.649	30972	-	0	9.79	10	-	0
brock800_4	800	0.650	30950	-	0	2.67	10	37.19	1
c-fat500-10	500	0.374	804000	1.21	10	$< \epsilon$	10	0.01	10
c-fat500-2	500	0.073	38350	0.27	10	$< \epsilon$	10	$< \epsilon$	10
c-fat500-5	500	0.186	205864	0.51	10	$< \epsilon$	10	$< \epsilon$	10
C1000.9	1000	0.900	234013	17.46	10	21.53	9	-	0
C2000.5	2000	0.500	14927	10.36	9	34.25	6	15.70	10
C2000.9	2000	0.900	317839	58.45	1	-	0	-	0
C4000.5	4000	0.500	19304	34.40	3	-	0	15.65	1
C500.9	500	0.900	164953	0.21	10	0.46	10	3.96	10
DSJC1000_5	1000	0.500	12054	0.13	10	2.35	10	0.27	10
hamming10-2	1024	0.990	13140816	8.47	10	0.09	10	0.43	10
hamming10-4	1024	0.829	83280	30.01	3	14.53	6	30.97	4
johnson16-2-4	120	0.765	3808	0.03	10	$< \epsilon$	10	0.10	10
johnson32-2-4	496	0.879	16330	0.34	10	-	0	1.58	10
keller4	171	0.649	6745	$< \epsilon$	10	0.01	10	0.01	10
keller5	776	0.752	38901	3.06	10	19.89	10	1.69	10
keller6	3361	0.818	176876	-	0	-	0	19.27	1
MANN_a27	378	0.990	802575	-	0	19.88	1	-	0
MANN_a45	1035	0.996	5872099	-	0	45.01	1	-	0
MANN_a81	3321	0.999	60365230	-	0	52.45	1	-	0
san1000	1000	0.502	10661	18.54	8	12.52	10	12.02	10
san400_0.5_1	400	0.500	7442	2.56	10	0.18	10	0.09	10

表 3: results on BHOSLIB

				MN/TS		PLS		LSCC	
Instance	$\|V\|$	$dens$	W_{best}	time[s]	success	time[s]	success	time[s]	success
frb30-15-1	450	0.824	44069	0.06	10	0.07	10	0.56	10
frb30-15-2	450	0.823	44078	0.06	10	0.05	10	0.22	10
frb30-15-3	450	0.824	43414	3.35	10	3.05	10	5.81	10
frb30-15-4	450	0.823	43884	0.14	10	0.03	10	0.84	10
frb30-15-5	450	0.824	43675	3.38	10	2.51	10	5.51	10
frb35-17-1	595	0.842	59629	16.53	10	13.59	10	18.00	9
frb35-17-2	595	0.842	59973	2.94	10	1.61	10	5.13	10
frb35-17-3	595	0.842	60357	1.07	10	1.13	10	17.03	10
frb35-17-4	595	0.842	59653	16.42	10	23.23	9	4.67	5
frb35-17-5	595	0.841	60749	0.29	10	2.01	10	1.01	10
frb40-19-1	760	0.857	79800	2.23	10	14.33	10	27.13	3
frb40-19-2	760	0.857	79004	26.02	9	25.72	8	31.32	1
frb40-19-3	760	0.858	79457	3.57	10	5.68	10	27.69	8
frb40-19-4	760	0.856	79247	13.57	10	33.58	10	30.44	5
frb40-19-5	760	0.856	79223	39.94	5	25.83	4	22.44	3
frb45-21-1	945	0.867	99802	27.55	3	26.22	4	-	0
frb45-21-2	945	0.869	99838	24.49	2	32.00	1	-	0
frb45-21-3	945	0.869	100282	19.04	6	53.43	1	-	0
frb45-21-4	945	0.869	101182	19.10	7	24.69	6	-	0
frb45-21-5	945	0.869	99614	24.16	1	22.49	3	-	0
frb50-23-1	1150	0.879	122931	42.84	1	-	0	-	0
frb50-23-2	1150	0.878	120808	1.34	1	12.42	1	-	0
frb50-23-3	1150	0.877	120840	24.06	2	-	0	-	0
frb50-23-4	1150	0.879	123298	34.08	6	23.22	5	-	0
frb50-23-5	1150	0.879	122846	40.48	2	-	0	-	0
frb53-24-1	1272	0.883	134988	0.62	1	-	0	-	0
frb53-24-2	1272	0.883	136739	49.20	1	-	0	-	0
frb53-24-3	1272	0.884	134907	-	0	8.00	1	-	0
frb53-24-4	1272	0.883	134704	17.96	1	-	0	-	0
frb53-24-5	1272	0.883	134294	55.56	1	-	0	-	0
frb56-25-1	1400	0.888	151823	10.69	1	-	0	-	0
frb56-25-2	1400	0.888	148263	33.38	1	-	0	-	0
frb56-25-3	1400	0.888	150869	24.00	1	-	0	-	0
frb56-25-4	1400	0.888	156615	41.94	2	-	0	-	0
frb56-25-5	1400	0.888	150693	57.79	1	-	0	-	0
frb59-26-1	1534	0.892	170472	14.07	1	-	0	-	0
frb59-26-2	1534	0.893	168783	49.86	1	-	0	-	0
frb59-26-3	1534	0.893	165236	40.63	1	-	0	-	0
frb59-26-4	1534	0.892	164884	28.91	1	-	0	-	0
frb59-26-5	1534	0.893	172795	25.24	1	-	0	-	0

の最適解は頂点数も最大になることが多く，結果的にMCP と同じような性質を持つ問題が多かったからではないかと考えられる．

今回の実験で，MCP に対して優れたアルゴリズムを MEWCP に用いられるように修正すれることでMEWCP に対しても優れたアルゴリズムとなる可能性があることを示した．ただし，現実的な場面において現れる MEWCP が今回作った問題と異なる性質を持つ可能性もあるので，今後，何らかの実データによる実験的検証を行うべきである．

今回は詳細に触れなかったが，MEWCP は近傍解の評価値を得ようとする際，候補集合の各頂点に接続している辺の重みの和を計算するため 1 回の遷移に多くの計算時間がかかっていた．今後の課題としては，評価値計算を工夫し計算量を抑えることが挙げられる．

参考文献

[1] Krishna Kumar Singh and Ajeet Kumar Pandey, “Survey of Algorithms on Maximum Clique Problem,” International Advanced Research Journal in Science, Engineering and Technology, Volume 2, issue 2, pp.18–20, 2015.

[2] Qinghua Wu, Jin-Kao Hao and Fred Glover, “Multi-neighborhood tabu search for the maximum weight clique problem,” Annals of Operations Research, Volume 196, issue 1, pp.611–634, 2012.

[3] Yiyuan Wang, Shaowei Cai and Minghao Yin, “Two Efficient Local Search Algorithms for Maximum Weight Clique Problem,” Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp.805–811, 2016.

[4] Wayne Pullan, “Approximating the maximum vertex edge weighted clique using local search,” Journal of Heuristics, Volume 14, issue 2, pp.117–134, 2008.

[5] The Second DIMACS Implementation Challenge, `https://turing.cs.hbg.psu.edu/txn131/clique.html`, 1992-1993.

[6] Shaowei Cai, Kaile Su and Abdul Sattar, “Local search with edge weighting and configuration checking heuristics for minimum vertex cover,” Artificial Intelligence, Volume 175, Issues 9-10, pp.1672–1696, 2011.

[7] Ke Xu, “BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring),” `http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm`