Vol.2018-AL-167 No.10
2018/3/8

Limited Discrepancy Search によるフィードバック辺集合の探索

兼本 樹[1,a)]　山口 一章[1,b)]　増田 澄男[1]

概要：フィードバック辺集合問題や，その一般化である線形順序付け問題は古くから研究が行われている問題である．本稿では，分枝限定法に基づいて，有望な解を含むと思われる解空間を優先的に探索する Limited Discrepancy Search (LDS) という探索法を用いてフィードバック辺集合問題の解を探索するアルゴリズムを示し，その際のいくつかの工夫について述べる．また，計算機実験により，LDS を用いたときの振る舞いを，整数計画問題として定式化してソルバー（CPLEX）で解いた場合と比較する．頂点数 30,50,100，辺密度 0.1,0.5,1.0 のランダムグラフの入力計 45 通りに対して計算機実験を行い，半数以上の（特に頂点や辺の多い）入力に対して，LDS が CPLEX より高速に良い解や最適解に到達することを示した．

キーワード：フィードバック辺集合問題，線形順序付け問題，階層描画，LDS

Itsuki Kanemoto[1,a)]　Kazuaki Yamaguchi[1,b)]　Sumio Masuda[1]

1. まえがき

有向グラフ $G = (V, E)$ と E の部分集合 S について，G の部分グラフ $(V, E \backslash S)$ が閉路を持たないとき，S は G のフィードバック辺集合であるという．有向グラフが与えられたときに最も要素数が少ないフィードバック辺集合を求める問題は，フィードバック辺集合問題（Feedback Arc Set Problem，以降，FASP と略記する）と呼ばれる．FASP は NP 困難であり [1]，その入力を任意の 2 頂点 u, v 間に有向辺 (u, v) または (v, u) のどちらか一方の辺のみが存在するトーナメントグラフと呼ばれるグラフに限定した問題もまた NP 困難である [2]．

具体的応用として，オブジェクトの集合について，オブジェクト間の関係性を表現・解析する際に有向グラフを用いると有効な場合がある．例えば複数の参加者が他の参加者と 1 対 1 の対戦を行う大会を考える．このような大会において，各参加者に対して順位付けを行いたい．このとき，大会の参加者を頂点とし，各対戦について勝者から敗者への有向辺を接続した有向グラフを考える．この有向グラフに閉路がない場合は，参加者の順位を勝敗に矛盾のないように決定できる．閉路がある場合，その閉路を構成している辺を削除（対戦結果を無視）して順位付けをすることが考えられるが，妥当な順位付けを考えるためには，削除する辺の数をなるべく少なくする必要がある．このような問題は，FASP として定式化できる．また，FASP は産業連関表をはじめとするするネットワーク構造を持つデータの解析 [3][4] やネットワーク解析などに用いられる確率伝搬法への応用 [5]，有向グラフを階層構造として描画する際の [6] の頂点の階層割り当てや辺交差数削減などへの応用が考えられている．

本稿では， Limited Discrepancy Search[7] (以降，LDS と略記する) という探索手法によって FASP の解を求める際のいくつかの工夫とその効果について述べる．LDS は，深さ優先探索による分枝限定法を基にした解探索手法である．LDS に似た探索法に反復深化深さ優先探索（Iterative Deepening Depth First Search，以降，IDDFS と略す）がある．IDDFS は探索の深さ d を徐々に大きくして探索空間を広げていくが，LDS ではより有望と思われる解空間を広く探索する点が異なる．LDS では発見的手法のアイデアを探索に利用することができ，多くの問題に利用可能で

[1] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University, Kobe-shi 657-8501, Japan
[a)] ij.kanemoto@gmail.com
[b)] ky@kobe-u.ac.jp

あると考えられる．

本稿の構成は以下のとおりである．まず第 2 章では，FASP の一般化である線形順序付け問題 (Linear Ordering Problem，以降，LOP と略記する) の定式化について示し，FASP と LOP の整数計画問題としての定式化を示す．第 3 章では，FASP の解を LDS で求める方法と，探索の際に行う工夫点について述べる．第 4 章では，様々な問題例を LDS と数理計画ソルバ CPLEX で解き，性能を比較する．最後に第 5 章では，まとめと今後の課題について述べる．

2. FASP の一般化と変形

2.1 線形順序付け問題 (Linear Ordering Problem, LOP)

辺重み付き有向グラフ $G = (V, E)$ に対して V の各頂点 v に順序 $\pi(v)$ が与えられたとき，G の辺 $(u, v) \in E$ について，$\pi(u) < \pi(v)$ ならば辺 (u, v) は順向辺であるといい，そうでなければ辺 (u, v) は逆向辺であるという．線形順序付け問題 (LOP) は，辺重み付き有向グラフ $G = (V, E)$ に対して，全ての逆向辺の重みの和が最小となる頂点の順序 π を求める問題である．このとき，LOP の解 π における逆向辺の集合は，フィードバック辺集合となる．よって LOP は，FASP を辺重み付きの問題に一般化した問題であるといえる．また，FASP は辺に単位重みが与えられた場合の LOP であるといえる．

2.2 整数計画問題 (Integer Programming, IP)

FASP の入力として有向グラフ $G = (V, E)$ が与えられたとき，以下の目的関数と制約条件をもつ整数計画問題 (IP) への変形が可能である．

最小化 $\sum_{(u,v)\in E} Y_{uv}$

制約

$$
\begin{aligned}
X_u - X_v &\le nY_{uv} - 1 && \forall (u,v) \in E \\
X_v - X_u &\le n(1 - Y_{uv}) - 1 && \forall (u,v) \in E \\
Y_{uv} &\in \{0, 1\} && \forall (u,v) \in E \\
X_v &\in \{0, 1, ..., n-1\} && \forall (v) \in V
\end{aligned}
$$

なおこの定式化は，$X_v = a$ は頂点 v に対して与えられる順序が $\pi(v) = a$ であることを表し，$Y_{uv} = 1$ は辺 (u, v) が逆向辺であることを，$Y_{uv} = 0$ は辺 (u, v) が順向辺であること表す．

また，入力の各辺 $(u, v) \in E$ に重み $w((u, v))$ が与えられているとき，目的関数を

最小化 $\sum_{(u,v)\in E} w((u,v))Y_{uv}$

とすることで LOP を IP に変形することができる．

3. Limited Discrepancy Search (LDS)

LDS は，解空間のうち有望な解を含むと思われる部分問題のみに範囲を限定した探索を，少しずつ探索範囲を広げていく探索法である．有望な解空間を先に調べることで，短時間で良質の解を得ることを目指している．LDS の探索は分枝限定法と同様，分枝操作と限定操作からなる．与えられた問題を根とし，探索木の各節点が部分問題に一対一対応するように子節点を作成しながら，深さ優先的に探索木を構築する．探索木の各節点に対して，その探索の有望度に応じた Discrepancy の値（以降，D と記す）を記録する．そして探索中に許容する D の上限値 D_{max} をあらかじめ定め，D_{max} を超えない範囲の解空間のみを探索する．D の値が d であるような節点の子節点については，以下のように計算した Δd を用いて $D = d + \Delta d$ を記録する．まず，分枝変数の集合 V の各要素 v に対して何らかのヒューリスティクスにより順位付けを行い，その順位を $rank(v)$ とする．そして，未処理の分枝変数のうち，最も順位の高い（順位 1 の）分枝変数 $v_0 = rank^{-1}(1)$ を $\Delta d = 0$ とし，以下順位 k の分枝変数は $\Delta d = k - 1$ とする．このように，D の値が小さい節点ほど有望な探索であるとみなし，その節点以降の探索空間を大きくとるのが LDS の特徴である．LDS による探索の擬似コードを以下に示す．

Algorithm 1: $LDS(V, D)$

```
if V = φ then
    解の評価
    return
end
v ∈ V について，rank(v) を計算
foreach D + rank(v) − 1 ≤ D_max を満たす v ∈ V do
    Δd = rank(v) − 1
    LDS(V\{v}, D + Δd)
end
```

LDS は $D_{max} = 0$ でまず探索を行い，以降，$D_{max} = 1, 2, 3, \ldots$ と順に増やし探索範囲を広げて探索を行っていく．この探索法では $D_{max} = x$ で探索する際，$D_{max} = x - 1$ の探索で訪れた節点を再び訪問する．ただし，$D_{max} = x - 1$ で訪問した節点は $D_{max} = x$ のときに比べ少ないので探索の重複による計算量の増加はさほど大きくないと考えられる．

LDS は見つけた解の最適性を検証しないため厳密解法ではなくメタヒューリスティックであると考えられるが，他の多くのメタヒューリスティックと違い最適解を有限時間内に必ず見つけ出すことができる．rank の定め方が適切であれば通常の分枝限定法よりも短い時間で暫定解が改良されていき，また，最適解を早く見つけ出せるが，そうでない場合には解の改善に時間を要する．

図 1 は $D_{max}=1$ において行われる探索を示している．各節点の子は，rank の小さいものから順に左から並んでおり，横の数字は D の値を示している．D が 2 以上の節点には×が付けられているが，その部分問題は $D_{max}=1$ の探索では訪問しないことを表している．$D=2$ の葉については $D_{max}=1$ の探索では訪問されず，$D_{max}=2$ の探索において初めて訪問される．同様に $D=d$ の葉については $D_{max}=d$ の探索において初めて訪問される．

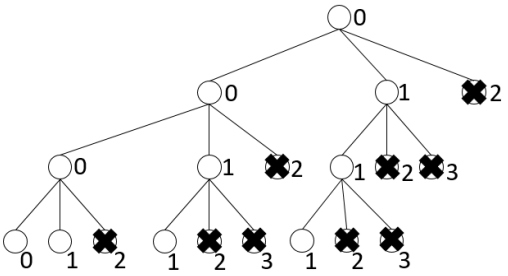

図 1 LDS の探索例（$D=1$）

3.1 FASP に対する構築型解法

FASP に対するヒューリスティックスとして，構築型解法 SORT がある [9]．このアルゴリズムでは，$insert, sort, reverse$ の 3 つの処理を繰り返して解を改善していく．$insert(u,\pi)$ は，部分的な順列 π に頂点 u を挿入する．その際，$|\pi|+1$ 箇所の挿入可能箇所のうち，逆向辺の数が最小となる箇所に u を挿入する．$sort(\pi)$ は，$\pi'=\phi$ に対して，$i=1,2,3,...,|\pi|$ の順に $insert(\pi(i),\pi')$ を繰り返し，新たにできた π' を π とする．$sort$ の操作を，順列に変化が起こらなくなるまで繰り返す処理を $sort*$ とする．$reverse(\pi)$ は，π の順序を逆順に並び替える操作である．このとき，任意の順列 π について，$reverse(\pi)$, $sort*(\pi)$ を順に適用することで，逆向辺の数が元の順列の逆向辺の数以下となることが証明されている．このように $reverse(\pi), sort*(\pi)$ によって逆向辺が減らなくなるまでこれらの処理を繰り返すことで，頂点に順列を定める．

3.2 FASP に対する LDS の適用

本稿では FASP の解を LDS で求める方法の一例を示す．FASP は頂点の順列が実行可能解であり，それに対する分枝限定法としては，順列の前から頂点を決定していくようなものが考えられる．本研究では限定操作として，優越ルールによる枝刈りを採用した．優越ルールとは，未探索の部分問題 P に対し P と同等以上の解を持つ部分問題 P' が探索済みのとき，P の探索を行わないことにより探索の高速化を図る手段である．順列の前から頂点を決定していく際，構築中の 2 つの順列に含まれる要素の組み合わせが同一ならば，それらの部分問題 P, P' を根とする部分探索木は同形となる．仮に確定している逆向辺の本数が P の方が多い場合や，P' が探索済で P が未探索であれば，P を探索せずに済ませることができる．そこで，処理済みの頂点の組み合わせに対して，その部分問題で逆向辺となることが確定している辺の数をハッシュテーブル H_{sp} に記録しておく．そして，他の部分問題の探索時に，同じ処理済み頂点の組み合わせが現れたら，確定している逆向辺の数がハッシュに記録されている数と同じかそれ以上のとき，探索を続けても改善解は見つからないのでその時点で枝刈りを行う．

上記のような分枝限定法による探索に LDS を適用すると以下のようなアルゴリズムになる．まず，前節の構築型解法 SORT により近似解の順列を求め，その解の順序に応じて各頂点 v の順位 $rank(v)$ を決定する．また，優越ルールによる枝刈りのため，H_{sp} を準備する．その後，探索中に許容する D の上限値 $D_{max}=0,1,2,...$ として以下のような探索木の構築を繰り返す．探索木の各節点は部分問題として，頂点順序が未決定の頂点からなる誘導部分グラフ $G'=(V',E')$，決定済みの頂点の順列 π，D の 3 つの情報を管理する．根ノードは $G=(V,E)$ と $D=0$ を記憶しており，深さ優先順に各ノードについて以下のような操作を行う．$G'=(V',E')$ にソース v が存在するとき，G' から v を削除し，π の末尾に v を追加する．$G'=(V',E')$ にシンク v が存在するとき，v は最終的な順列の末尾に順序付けされれば逆向辺の数に影響しないため，単に v を削除する．これらの処理をソースとシンクがなくなるまで繰り返す．その後，V' が空ならば，元の入力の G に対して頂点順序 π を与えて逆向辺をカウントし，解が改善されていれば暫定解を更新して，バックトラックする．V' が空でないとき，以下の分枝処理を行う．$rank$ をもとに，V' に含まれる頂点に順位 $rank'$ を定める．頂点 v に対する分枝について，$\Delta d = rank'(v)-1$ である．部分問題 $G'=(V',E')$ に対して，分枝変数 v を $rank'$ の小さい頂点から順に $D+\Delta d$ が D_{max} を超えない範囲で選び，各 v に対して，$V'\backslash\{v\}$ からなる誘導部分グラフ G''，π の末尾に v を挿入した π'，$D'=D+\Delta d$ を記録した子節点を作成する．その際，$H_{sp}[V'\backslash\{v\}]$ の値が π' における確定した逆向辺の数以下の場合，子節点は作成せず枝刈りを行う．そうでなければ，$H_{sp}[V'\backslash\{v\}]$ の値を，π' における確定した逆向辺の数で更新する．また，π' の末尾の 3 頂点からなる部分順列が最適でない場合は枝刈りを行う．このアルゴリズムの擬似コードを以下に示す．

このように LDS は，D の値の小さい節点，すなわち，最初のヒューリスティックスで見つけた解に近い解空間を優先的に探索する．この LDS で探索木全体を探索するためには，$rank'(v)$ が最大のものを選び続けた場合の π を探索する必要があり，$D_{max}=n+(n-1)+...+2+1=n(n+1)/2$ での全探索を行うことが必要となる．しかし，探索後半は

Algorithm 2: $LDS(G' = (V', E'), \pi, D)$

確定している逆向辺の集合 S を計算
if H_{sp} に $H_{sp}[V'] \leq |S|$ を満たすエントリが存在 **then**
  return (優越ルールによる枝刈り)
else
  H_{sp} にエントリ $H_{sp}[V'] = |S|$ を追加
end
if $V' = \phi$ **then**
  シンクとして削除した頂点を π の末尾に追加
  順序 π に対する逆向辺を数え，暫定解より少なければ解を更新する
  return
end
while ソース v が存在 **do**
  π の末尾に v を追加
  $V' \leftarrow V' - v$
end
while シンク v が存在 **do**
  $V' \leftarrow V' - v$
  v をシンクとして記憶
end
$v \in V'$ について，$rank'(v)$ を計算
foreach $D + rank(v) - 1 \leq D_{max}$ を満たす $v \in V$ **do**
  $\Delta d = rank(v) - 1$
  $\pi' \leftarrow \pi$ の末尾に v を追加した順序
  $G'' \leftarrow V' \backslash \{v\}$ からなる誘導部分グラフ
  if π' の末尾から３頂点の順列が最適 **then**
    $LDS(G'', \pi', D + \Delta d)$
  end
end

最適解から遠いであろう解空間の探索であるため，最適解に近い解を探索序盤から出力することができるという特徴を持つ．

4. 計算機実験

LDS の振る舞いを評価するため，FASP を整数計画問題として定式化し数理計画ソルバー CPLEX[8] で解を求めた場合と，LDS で解を求めた場合の解の質を比較する．LDS は分枝変数の rank を決定するため，構築型解法 SORT によって得られた頂点の順序 π_0 を用いる．また，CPLEX にも初期解として π_0 を与える．

入力のグラフは，ランダムに生成した頂点数 $|V| = 30, 50, 100$，辺密度 $\alpha = 0.1, 0.5, 1.0$ の有向グラフであり，それぞれの頂点数と辺密度のグラフ 5 通りずつ，合計 45 通りのグラフを用いる．いずれのグラフも，弱連結な木を生成したのちに，全てのソースに対して出る辺をランダムに追加，全てのシンク対してに出る辺をランダムに追加する処理を行い，その後，辺密度が指定の値になるまで辺を追加できる箇所に一様ランダムに辺を追加する処理を繰り返して作られている．よっていずれの入力のグラフにも，長さ 2 以下の閉路，多重辺，シンク，ソースが存在しない．重み付き問題については，重みなしの問題の各辺に対して，[1,99] の一様ランダムな重みを付けたグラフを用いた．

実行環境は以下のとおりである．

- CPU: core i5-4460 3.2GHz
- メモリ: 16GB
- 言語: Java(TM) SE 8

4.1 実験結果

実験結果を表 1,2 に示す．アルゴリズム SORT は，頂点数 100 のグラフに対しても，1ms 以内に解を出力したため実行時間を表記していない．CPLEX は実行時間が 1 時間を超えてもプログラムが終了しなかったとき，暫定解を出力させている．sol は解の逆向辺の数を示している．sol1m は，プログラム開始から 1 分以内に見つかった最善の解を示している．

4.2 考察

重みなしの有向グラフの入力 (FASP) について，CPLEX が 1 時間以内に最適解を出力した問題は 27/45 問である．そのうち 13/27 問に対して LDS は 1 分以内に最適解に到達している．一方で，5/27 問については SORT の出力した (最適解ではない) 初期解から解の改善が 1 度も起きていない．CPLEX が 1 時間以内に最適解を出力できなかった問題 18/45 問全てに対して，LDS は CPLEX の暫定解以上の解を 1 分以内に発見している．また，実行時間を 1 分に制限した場合，45 問中 5 問は CPLEX が，20 問は LDS がより良い解に到達している．

表 1　FASP $|V| = 30, 50, 100$

$\|V\| = 30$		SORT	CPLEX			LDS
α	No.	sol	time[s]	sol	sol1m	sol1m
0.1	1	14	0.04	11	11	11
	2	18	0.11	16	16	16
	3	14	0.01	14	14	14
	4	15	0.01	14	14	14
	5	14	0.04	14	14	14
0.5	1	53	1.3	52	52	52
	2	52	0.9	52	52	52
	3	54	1	52	52	54
	4	52	0.8	51	51	52
	5	50	1.1	50	50	50
1.0	1	149	2231.7	144	149	148
	2	125	24.7	123	123	125
	3	142	540.7	137	142	140
	4	137	365.9	135	137	135
	5	122	54.2	121	121	121
$\|V\| = 50$		SORT	CPLEX			LDS
α	No.	sol	time[s]	sol	sol1m	sol1m
0.1	1	18	0.2	17	17	17
	2	17	0.2	17	17	17
	3	22	0.6	21	21	22
	4	17	0.3	17	17	17
	5	22	0.6	20	20	21
0.5	1	173	1350.1	163	173	168
	2	177	1985.6	170	177	172
	3	173	506.7	169	173	173
	4	157	151.3	154	157	156
	5	171	308.1	167	170	170
1.0	1	400	3600	400	400	396
	2	423	3600	423	423	423
	3	417	3600	417	417	416
	4	421	3600	421	421	421
	5	418	3600	418	418	413
$\|V\| = 100$		SORT	CPLEX			LDS
α	No.	sol	time[s]	sol	sol1m	sol1m
0.1	1	87	3600	87	87	87
	2	88	2564.5	82	88	85
	3	88	3600	81	88	87
	4	88	2320.7	85	87	87
	5	97	3600	93	97	97
0.5	1	845	3600	844	844	843
	2	868	3600	867	868	867
	3	862	3600	861	862	861
	4	860	3600	860	860	859
	5	834	3600	834	834	834
1.0	1	1882	3600	1882	1882	1881
	2	1903	3600	1903	1903	1898
	3	1905	3600	1905	1905	1902
	4	1891	3600	1891	1891	1890
	5	1916	3600	1916	1916	1914

表 2　LOP $|V| = 30, 50, 100$

$\|V\| = 30$		SORT	CPLEX			LDS
α	No.	sol	time[s]	sol	sol1m	sol1m
0.1	1	639	0.1	535	535	537
	2	569	0.1	555	555	555
	3	545	0.1	510	510	510
	4	626	0.1	543	543	543
	5	464	0.1	365	365	365
0.5	1	2149	1.0	2128	2128	2128
	2	2531	1.2	2276	2276	2293
	3	2239	1.2	2059	2059	2059
	4	2422	1.7	2256	2256	2285
	5	2117	1.2	1928	1928	1928
1.0	1	6392	117.8	6242	6254	6242
	2	5263	7.4	5024	5024	5024
	3	6428	25.6	6043	6043	6043
	4	6278	27.5	6255	6255	6255
	5	5503	5.3	5503	5503	5503
$\|V\| = 50$		SORT	CPLEX			LDS
α	No.	sol	time[s]	sol	sol1m	sol1m
0.1	1	715	0.4	559	559	577
	2	756	0.3	594	594	629
	3	950	0.2	680	680	703
	4	545	0.3	545	545	545
	5	992	0.5	813	813	880
0.5	1	8703	96.3	7721	7728	8224
	2	7740	594.5	7468	7698	7590
	3	7263	70.1	6953	6953	7167
	4	7699	78.9	6986	6987	7078
	5	8385	527.6	7469	7547	8053
1.0	1	19461	3600	19441	19441	18587
	2	20371	3600	20362	20362	20265
	3	19790	3600	19790	19790	19445
	4	18825	3600	18796	18796	18423
	5	19827	3600	19796	19796	19625
$\|V\| = 50$		SORT	CPLEX			LDS
α	No.	sol	time[s]	sol	sol1m	sol1m
0.1	1	3874	636	3424	3612	3705
	2	4006	2250.4	3576	3879	3960
	3	3910	1922.9	3097	3579	3810
	4	4129	3600	3694	3942	4000
	5	4187	875.4	3526	3775	4083
0.5	1	39499	3600	39279	39454	39191
	2	40700	3600	40498	40670	40295
	3	41508	3600	41326	41341	40919
	4	40844	3600	40797	40816	40698
	5	39881	3600	39698	39802	39739
1.0	1	90617	3600	90570	90570	90550
	2	90009	3600	89974	89990	89776
	3	92097	3600	92097	92097	91769
	4	91077	3600	91036	91077	90901
	5	92925	3600	92883	92925	92461

重みありの有向グラフの入力 (LOP) について，CPLEX が 1 時間以内に最適解を出力した問題は 29/45 問である．そのうち 13/29 問に対して LDS は 1 分以内に最適解に到達している．CPLEX が 1 時間以内に最適解を出力できなかった問題 18/45 問全てに対して，LDS は CPLEX の暫定解以上の解を 1 分以内に発見している．LDS は，全ての問題に対して，SORT が出力した (最適解ではない) 初期解を少なくとも 1 回以上改善している．また，実行時間を 1 分に制限した場合，45 問中 16 問は CPLEX が，17 問は LDS がより良い解に到達している．

重みなし，重みありの問題ともに，頂点数や辺の数が少ないグラフに対しては CPLEX のほうが高速に最適解を出力している．一方で，$(|V|, \alpha) = (50, 1.0), (100, 0.5), (100, 1.0)$ などのサイズの大きい問題に対しては，CPLEX と比べて LDS の解の改善が高速に行われ，良い解が得られる傾向にある．これらのことから，現実的な時間で全探索を行えないような大きな入力に対しては，LDS は有望な解空間を効率的に探索できていると考えられる．

5. あとがき

本研究では，LDS によって FASP や LOP を解くアルゴリズムを提案した．またその際，通常の分枝限定法で有効な優越ルールによる枝刈りを行うことで探索を高速化する方法を示した．計算機実験では，使用した入力の半数以上に対して，LDS が CPLEX より高速に良い解や最適解に到達することを確認した．

LDS を用いるメリットとしては，有望な解集合を優先的に探索し解の早期改善を目指すことができること，最適解でない場合に必ず改善解を発見できることなどが挙げられる．一方で，正確な厳密解法とは異なり解の最適性の保証はできないこと，探索のランダム性がないため，初期解の与え方次第では改善解の発見に膨大な時間がかかる可能性があることなどが挙げられる．

今後の課題としては，枝刈りの優越ルールの改善，同じ解空間を探索しないような工夫を施すこと，ラグランジュ緩和などによる下界計算，並列計算による高速化などが考えられる．また，他の組み合わせ最適化問題に対しても，LDS を有効に動作するか確認することなどが考えられる．

参考文献

[1] R.M. Karp: Reducibility among combinatorial problems. in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press , New York, 1972) 85-103.
[2] N. Alon: Ranking tournaments. SIAM Journal on Discrete Mathematics, 20 (2006), 137-142.
[3] H.B. Chenery and T. Watanabe: International comparisons of the structure of production. Econometrica, 26 (1958), 487-521.
[4] W. Leontief: Structure of American Economy 1919-1929 (Harvard University Press, Cambridge, 1941)
[5] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Interface. Morgan Kaufmann Publishers Inc., 1988.
[6] Peter Eades and Nicholas C. Wormald, “Edge Crossings in Drawings of Bipartite Graphs,” Algorithmica, 11(1994), 379-403.
[7] W.D.Harvey and M.L.Ginsberg, “Limited discrepancy search,” Proc. the International Joint Conference on Artificial Intelligence, Montreal, 1995, 607-613.
[8] https://www-03.ibm.com/software/products/ja/ibmilogcple
[9] S. Chanas and P.Kobylanski: Anew heuristic algorithm solving the linear ordering problem. Computational Optimization and Applications, 6 (1996), 191-205.