Heuristics for the Minimum Linear Arrangement Problem

A. Duarte, J.J. Pantrigo, V. Campos and R. Martí
ETSII - URJC Madrid
Spain

May-11-2008

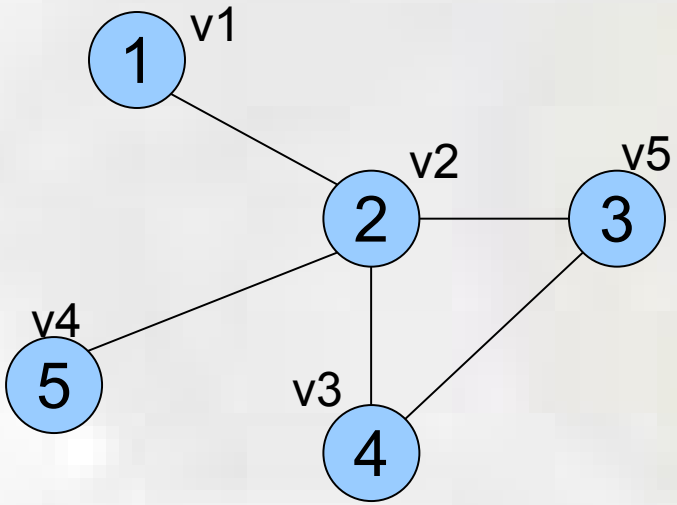
Outline

- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

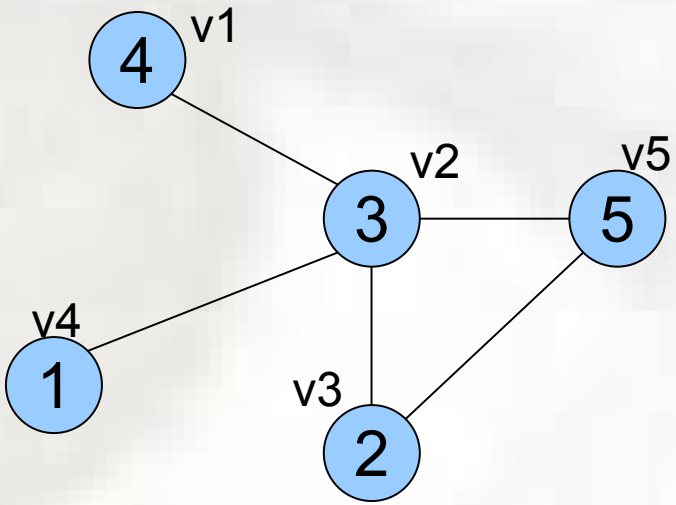
Introduction

- Labelling or linear layout: given a graph $G=(V,E)$ assign a integer value (label) to each vertex of V

Example 1



Example 2



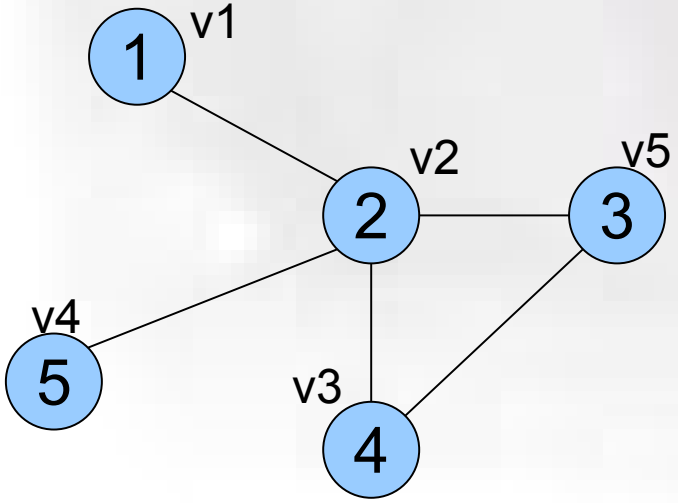
Sol1 = (1,2,4,5,3)

Sol2 = (4,3,2,1,5)

Introduction

- **Linear Arrangement Minimization Problem** (MinLA): Find a labelling that minimizes the sum of the differences between labels of adjacent vertices

- Given a graph $G=(V,E)$ with $|V| = n$ and $|E| = m$, find an arrangement f of G that assign the integers $1,2,..,n$ to the vertices of V



- Sol = (1,2,4,5,3)

Cont. of v1: |1-2|
Cont. of v2: |2-1|+|2-5|+|2-4|+|2-3|
Cont. of v3: |4-2|+|4-3|
Cont. of v4: |5-2|
Cont. of v5: |3-2|+|3-4|
Obj. Func= ½ all cont. = 8

Introduction

- MinLA is an NP-hard problem
- Relations with other well-known layout problems
  - Bandwidth minimization problem
  - Profile minimization problem
- Applications
  - Structural engineering
  - VLSI design
  - Software testing

Introduction

- Previous approaches
  - Harper (1964)
  - Juvan and Mohar (1992)
  - McAllister (1999)
  - Koren and Harel (2002)
  - Petit (2003)
  - Safro et al. (2006)
  - Rodriguez-Tello et al. (2007)

Outline

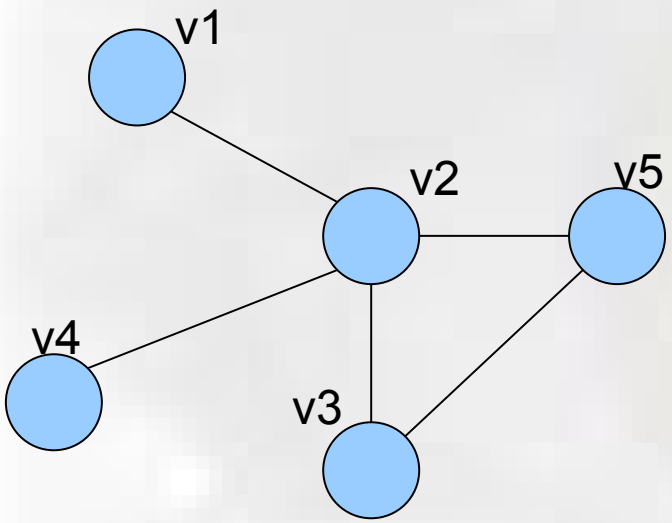
- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

Constructive Methods

- **Previous methods**: Frontal Increase Minimization
  - Set *NSel* = *V*, *Sel* = ø and *CL* = ø
  - The first vertex *v* is randomly selected from *NSel* and labelled as 1. In subsequent construction steps, *CL* consists of all the vertices in *NSel* that are adjacent to at least one labelled vertex
  - A vertex *v* is randomly selected from *CL*, labelled with the next available label and deleted from *NSel*
  - The method finishes when all the vertices have received a label

Constructive Methods

- **Previous methods:** Frontal Increase Minimization
  - Select randomly a vertex from *NSel* (v5) and assign the first free label (1)

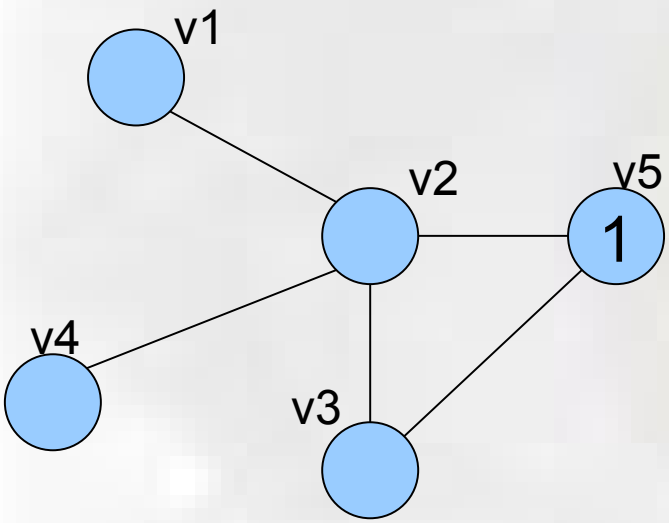


- *Labels*: 1,2,3,4,5
- *NSel*: v1, v2, v3, v4, v5
- *Sel*: ø
- *CL*: ø

Constructive Methods

- **Previous methods**: Frontal Increase Minimization

  • Select randomly a vertex from *CL* (v3) and assign the second free label (2)

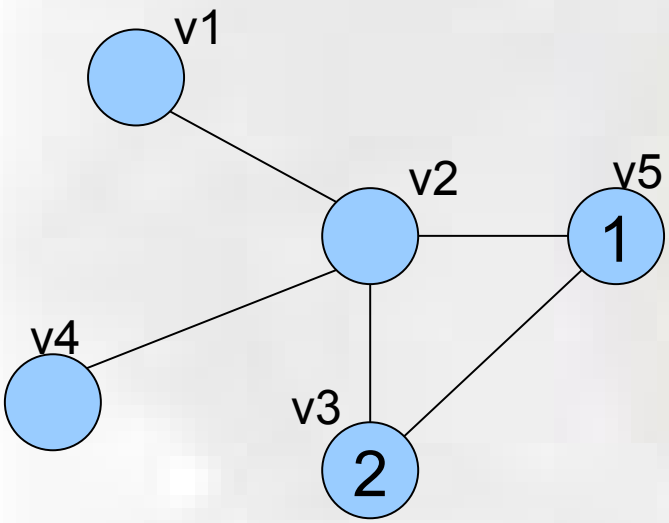


- *Labels*: 2,3,4,5
- *NSel*: v1, v2, v3, v4
- *Sel*: v5
- *CL*: v2,v3

Constructive Methods

- **Previous methods**: Frontal Increase Minimization

  - Select randomly a vertex from *CL* (v2) and assign the third free label (3)

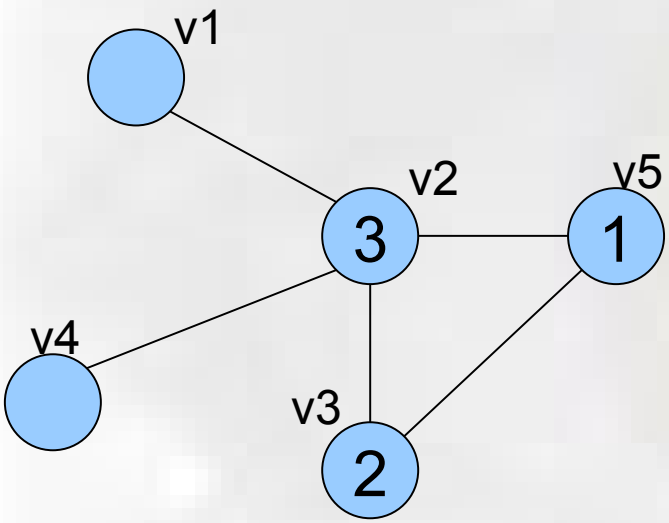


- *Labels*: 3,4,5
- *NSel*: v1, v2, v4
- *Sel*: v5,v3
- *CL*: v2

Constructive Methods

- **Previous methods**: Frontal Increase Minimization

  - Select randomly a vertex from *CL* (v1) and assign the fourth free label (4)

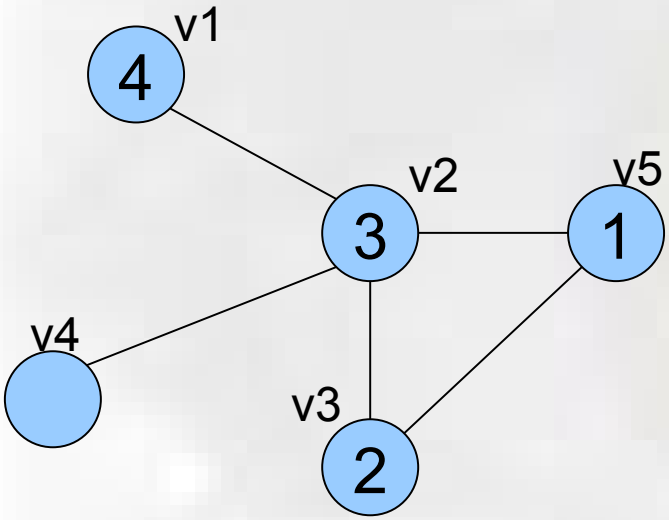


- *Labels*: 4,5
- *NSel*: v1,v4
- *Sel*: v5,v3,v2
- *CL*: v1,v4

Constructive Methods

- **Previous methods**: Frontal Increase Minimization

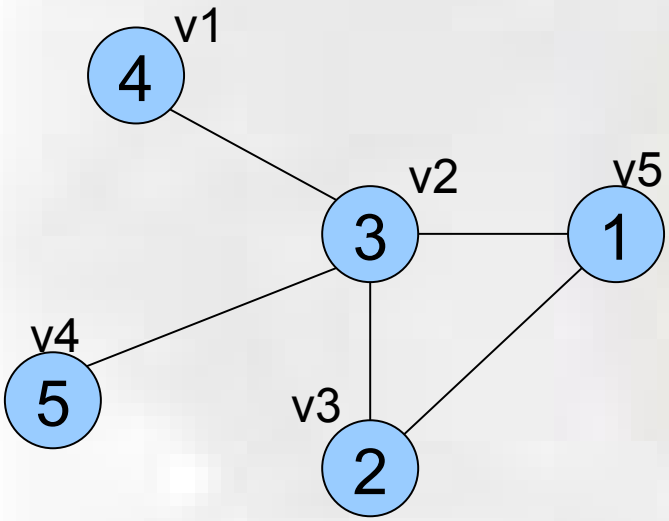
  - Select randomly a vertex from *CL* (v4) and assign the fitht free label (2)



- *Labels*: 5
- *NSel*: v4
- *Sel*: v5,v3,v2,v1
- *CL*: v4

Constructive Methods

- **Previous methods**: Frontal Increase Minimization
  - The algorithm finishes because all the vertices are labelled



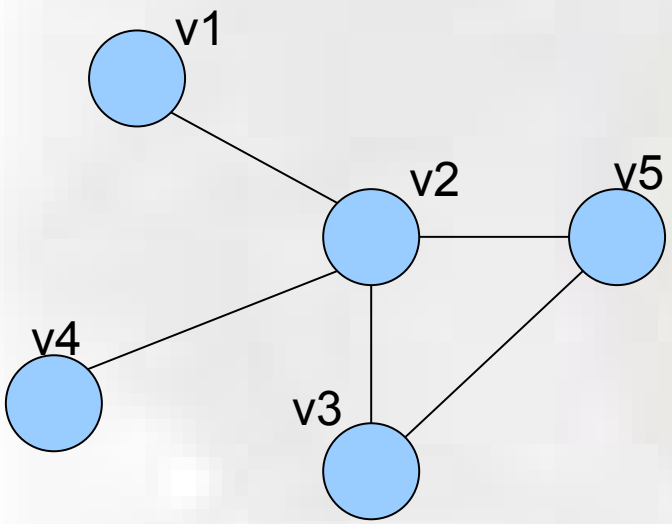
- *Labels*: ø
- *NSel*:
- *Sel*: v5,v3,v2,v1,v4
- *CL*: ø

Constructive Methods

- **Previous methods**: McAllister (C1)
  - Set *NSel* = *V*, *Sel* = ø and *CL* = ø
  - The first vertex *v* is randomly selected from *NSel* and labelled as 1. In subsequent construction steps, *CL* consists of all the vertices in *NSel* adjacent to at least one labelled vertex
  - Compute $sf(v) = d_{NSel}(v) - d_{Sel}(v)$
  - Select the vertex *v* in *CL* with minimum value of *sf*(*v*) with the next available label and deleted from *NSel*
  - The method finishes when all the vertices have received a label

Constructive Methods

- **Previous methods**: McAllister (C1)
  - Select randomly a vertex from *NSel* (v5) and assign the first free label (1)

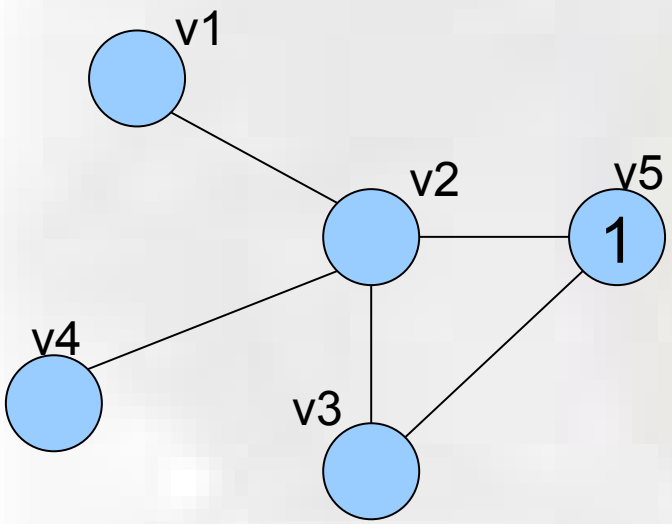


- *Labels*: 1,2,3,4,5
- *NSel*: v1, v2, v3, v4, v5
- *Sel*: ø
- *CL*: ø

Constructive Methods

- **Previous methods**: McAllister (C1)

  - Select the vertex in *CL* with minimum value of *sf* (v5) and assign the second free label (2)

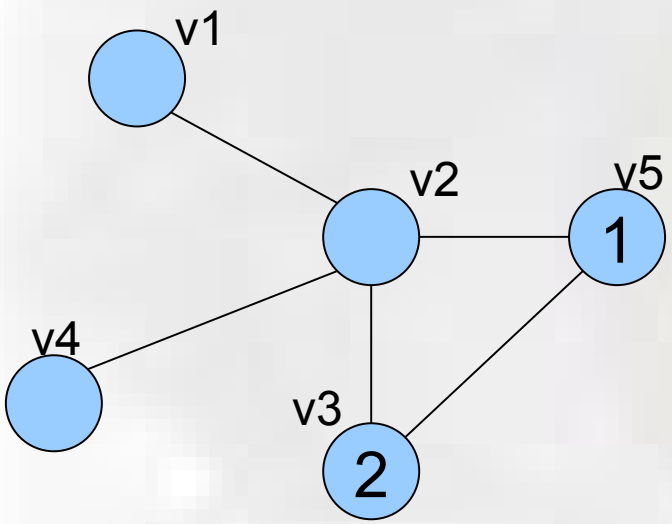


- *Labels*: 2,3,4,5
- *NSel*: v1, v2, v3, v4
- *Sel*: v5
- *CL*: v2,v3
  - *sf*(v2)=3-1 = 2
  - *sf*(v3)= 1-1 = 0

Constructive Methods

- **Previous methods**: McAllister (C1)

  - Select the vertex in *CL* with minimum value of *sf* (v2) and assign the third free label (3)

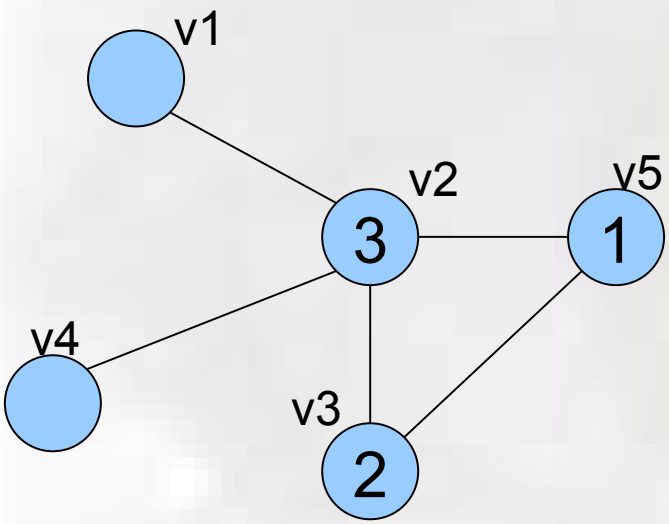


- *Labels*: 3,4,5
- *NSel*: v1, v2, v4
- *Sel*: v5,v3
- *CL*: v2
  - *sf*(v2)=2-2=0

Constructive Methods

- **Previous methods**: McAllister (C1)

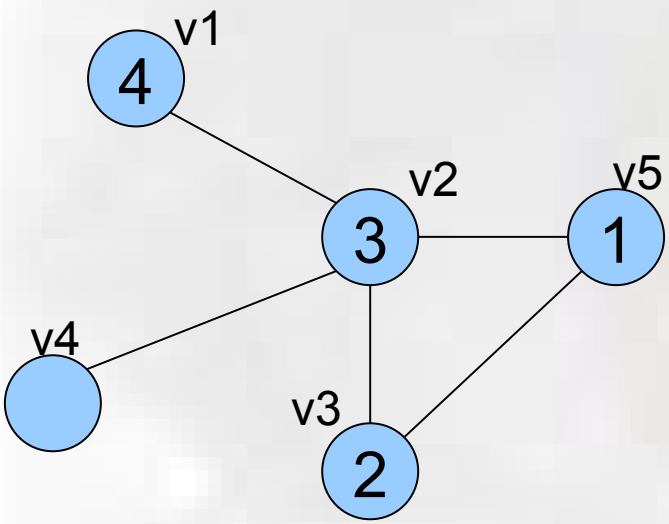
  - Select the vertex in *CL* with minimum value of *sf* (v1) and assign the fourth free label (4)



- *Labels*: 4,5
- *NSel*: v1,v4
- *Sel*: v5,v3,v2
- *CL*: v1,v4
  - *sf*(v1)=0-1=-1
  - *sf*(v4)=0-1=-1

Constructive Methods

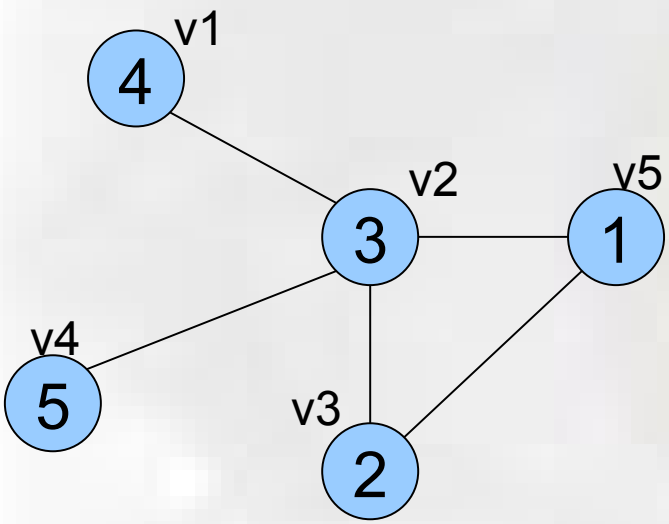
- **Previous methods**: McAllister (C1)
  - Select the vertex in *CL* with minimum value of *sf* (v4) and assign the fifth free label (5)



- *Labels*: 5
- *NSel*: v4
- *Sel*: v5,v3,v2,v1
- *CL*: v4
  - *sf*(v4)=0-1=-1

Constructive Methods

- **Previous methods**: McAllister (C1)
  - The algorithm finishes because all the vertices are labelled



- *Labels*: ø
- *NSel*: ø
- *Sel*: v5,v3,v2,v1,v4
- *CL*: ø

Constructive Methods

- **Proposed Method** (C2): Based on GRASP
  - **Greedy**: Sort all candidates according to sf(*v*).
    - *CL*: Elements adjacent to labelled elements
    - *RCL*: a subset of the best elements of CL

    $RCL = \{v \in CL \mid sf(v) < th\}$

    $th = msf + \alpha\,(Msf\text{-}msf) \qquad Msf = \max_{v \in CL} sf(v) \qquad msf = \min_{v \in CL} sf(v)$

  - **Random**: One element is randomly selected form the *RCL*
  - **Adaptive**: Recalculate *sf*(*v*) and the elements in *CL*

Constructive Methods

- **Proposed Method** (C3): Based on GRASP
  - Using C2 and Including the contribution of the selected vertex to the objective function

$$C(v,l) = \sum_{u \in N(v) \cap L} |f(u) - l|$$

  - *RCL* is now formed with the vertices with minimum *sf*-value and with *C*-value lower than *th2*

CL = vertices adjacent to at least one labelled vertex

$CL2 = \{v \in CL \mid sf(v) = msf\}$

$RCL = \{v \in CL2 \mid C(v) < th2\}$

$$th_C = dm_L + \beta\,(dM_L - dm_L) \quad dM_L = \max_{v \in CLmsf} C(v,l) \quad dm_L = \min_{v \in CLmsf} C(v,l)$$

Outline

- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

Improvement Method

- Given to vertices, u and v with label $f(u)$ and $f(v)$ a movement $move(u,v)$ consist of exchanging the labels between both vertices

Before	After
u, $f(u)$	u, $f(v)$
v, $f(v)$	v, $f(u)$

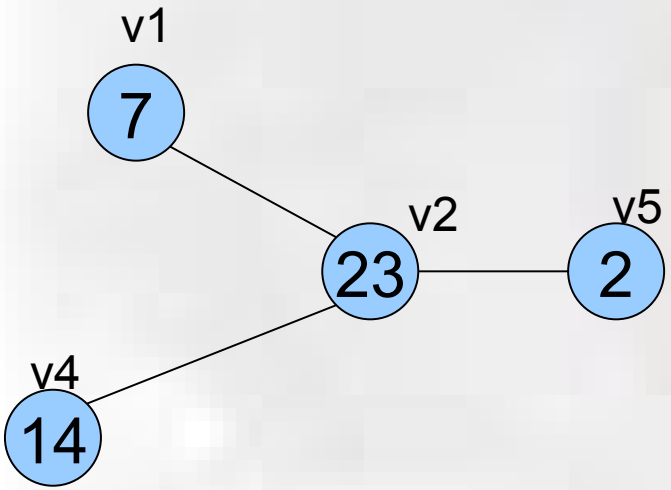
- Change in the objective funciton

$$ObjFunc_{after} = ObjFunc_{before} - MoveValue(u,v)$$

- The larger the *MoveValue*, the better the move

Improvement Method

- **Previous approaches**: The best label for each vertex is the median of their adjacent.
  - Example: Partial graph.



- Contribution of v2

  |23-2|+|23-7|+|23-14|=46

- Median of their adjacents

  Med(v1,v4,v5)=7

- Objective function reduction

  MoveValue(v2,v1)= 18

- If we assign the label 8

  MoveValue(v2,X)= 33

Improvement Method

- It is better to consider a label “close” to the median
- This problem arises when a vertex has an odd-degree
  - The median correspond to the label of one of their adjacents
- It is a huge problem when a vertex has only one edge because the search
  - The selection of the median label could even cycle the search

Improvement Method

- New Improvement method:
  - Not only considering the median value, but exploring a set of candidate labels

$$CL(u)=\{l : |l - Med(u)| \leq width, \ \forall l \neq f(v) \ v \in N(u)\}$$

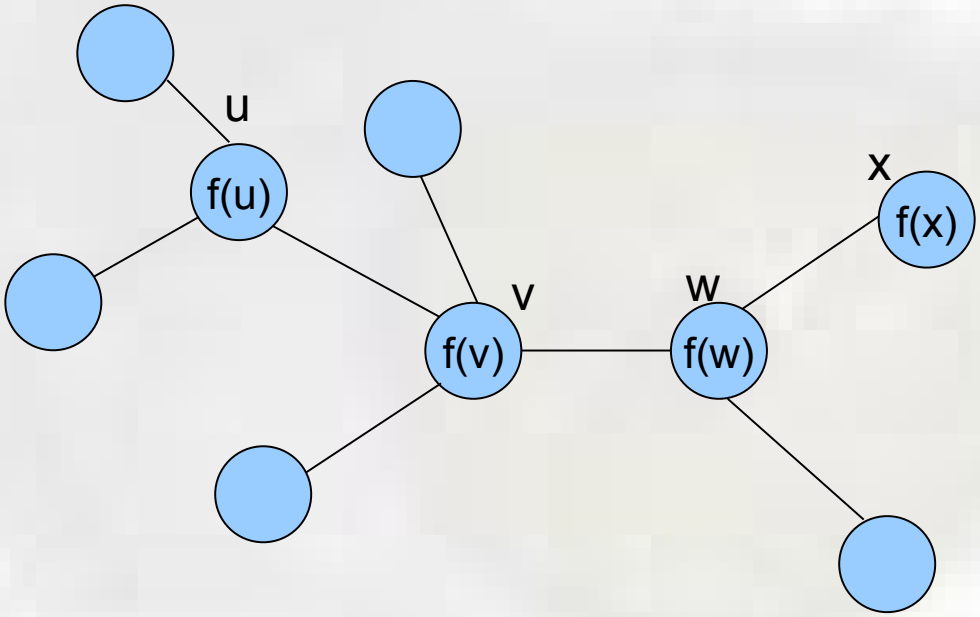
  - Avoiding the labels of adjacent vertices
  - Improvement method based on Ejection Chain Methodology:
    - Changes in selected elements cause other elements to be ejected from their current state
    - Compound moves

Improvement Method

- Ejection Chain for MinLA:
  - Suppose vertices u, v with associated labels $f(u)$, $f(v)$ with $MoveValue(u,v) < 0$
  - The movement could be “good” for u, but “bad” for v
    - Look for the best label for v associated to a vertex w
    - Try to perform $MoveValue(v,w)$
  - Follow the same logic until a given threshold

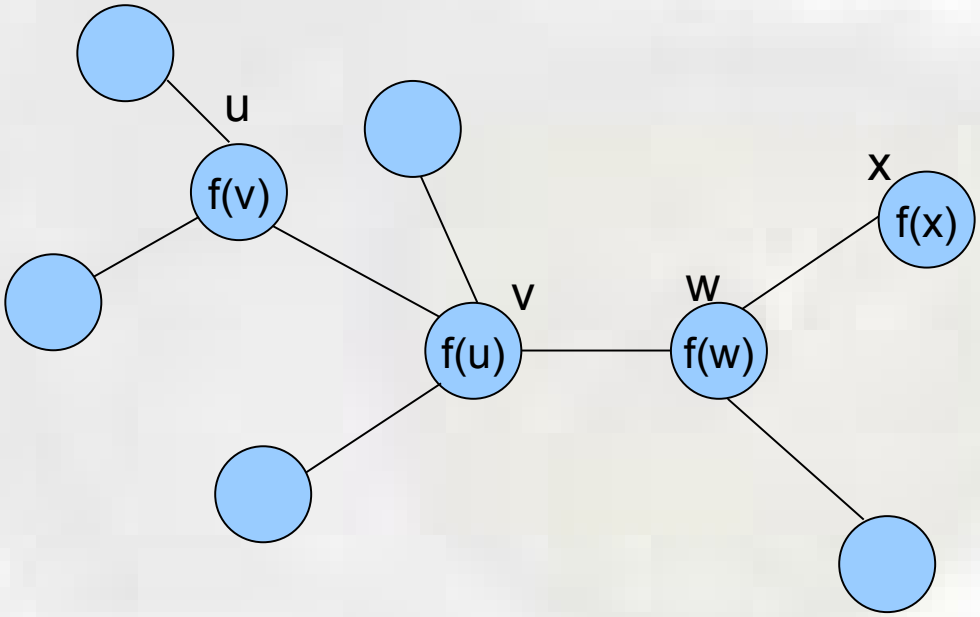
Improvement Method

- Ejection Chain for MinLA:
  - Example of the EC



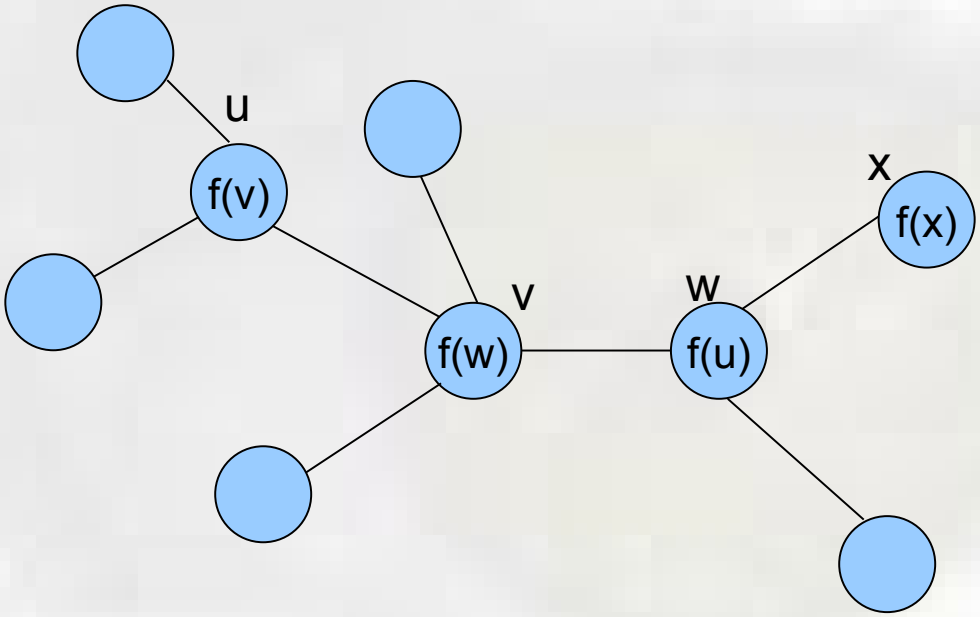
Improvement Method

- Ejection Chain for MinLA:
  - Interchange labels between *u* and *v*



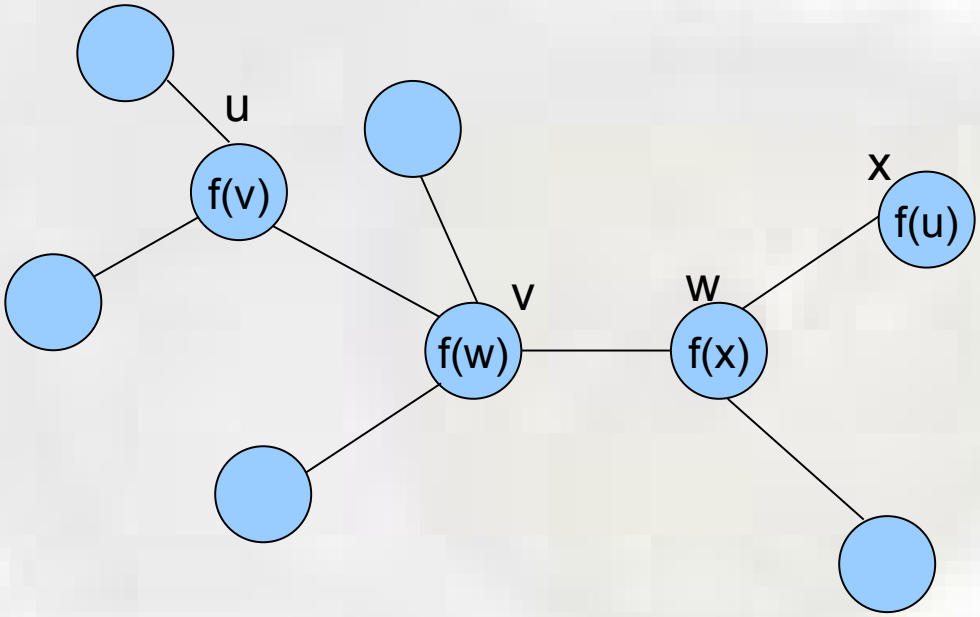
Improvement Method

- Ejection Chain for MinLA:
  - Interchange labels between *v* and *w*



Improvement Method

- Ejection Chain for MinLA:
  - Interchange labels between *w* and *x*



Improvement Method

- Ejection Chain for MinLA:
  - The improvement method starts by searching a “good” label for the vertex *u* under study
  - Restrict the search only to vertices in *CL*(*u*) and select the best vertex *v* in *CL*(*u*)
  - If *MoveValue*(*u*,*v*) ≥ 0 perform the movement
  - If *MoveValue*(*u*,*v*) < 0 look for a new label for *v* in *CL*(*v*) and repeat the same logic
  - The procedure is maintained until the compose movement is positive or a threshold value is reached

Outline

- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

GRASP Algorithm

- Grasp methodology consist of applying iteratively the following two phases:
  - **Construction Phase:** It is guided by a greedy function, that adapts according to the selection of previous steps
  - **Improvement Phase:** It performs a sequence of moves towards a local optimum
- Improvement Phase is the more consuming running time procedure
  - Try to increase the efficiency by estimating the percentage of improvement achieved by this improvement phase

GRASP Algorithm

- The percentage of improvement is defined as:

$$P(i)=\frac{LA(G,f_i)-LA(G,f^*_i)}{LA(G,f_i)}$$

- After *n* GRASP iterations, the mean and standard deviation can be estimated as:

$$\hat{\mu}_P=\frac{\sum_{i=1}^{n} P(i)}{n} \qquad \hat{\sigma}_P=\sqrt{\frac{\sum_{i=1}^{n}(P(i)-\hat{\mu}_P)^2}{n-1}}$$

- At a given iteration *i* (after the first *n* iterations), the minimum percentage of improvement necessary for a construction to be better than the best solution is:

$$imp(i)=\frac{LA(G,f_i)-LA(G,f_{best})}{LA(G,f_i)}$$

GRASP Algorithm

- If *imp*(*i*) is close to the mean it is “likely” that applying the improvement method to the current construction produce a better solution than the best solution
- In order to save computational time, the improvement method is only applied to promising constructions

$$imp(i) < \hat{\mu}_p + \delta \{ \hat{\sigma}_p ¿$$

- Where δ represents the number of standard deviations away from the estimated mean

Outline

- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

Path Relinking

- **Path Relinking Methodology**: Exploration of the paths between solutions
- Our implementation to MinLA has two phases
  - Construct an Elite set with the solutions provided by the GRASP algorithm
    - Quality Solutions
    - Diversity solutions
  - Relinking process between each pair of solutions

Path Relinking

- Construction of the Elite set:
  - **Quality solutions**: the best objective function (lowest value)
  - **Diverse solutions**: the highest distance among them
    - Distance between solutions (permutations):

$$d(p,q)=\sum_{i=1}^{n}\partial_i \qquad \partial_i=\begin{cases}1\ if\ \ p_i\neq q_i \quad and \quad p_i\neq n-q_i+1\\ 0\ otherwise\end{cases}$$

    - This definition avoids reverse labellings

Path Relinking

- Relinking process
  - Applied to each pair of solutions in Elite Set (initiating and guiding solutions)
  - The path is basically constructed assigning iteratively the label of a vertex in the guiding solution to the initiating solution
  - Every certain number of iterations, the Improvement method procedure is applied to the solution under construction

Outline

- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

Computational Experiments

- Experiments were performed on a personal computer, 3.2 GHz Intel Xenon processor and 2.0 GB of RAM
- Set of instances: 20 graphs ($62 \leq n \leq 10{,}240$ and $125 \leq m \leq 30{,}380$), introduced in Petit (2003a)
  - Designed to be difficult; i.e. in that they cannot be optimally solved by a brute force algorithm
  - Random graphs, VLSI, graph-drawing and engineering (fluid-dynamics and structural mechanics).

Computational Experiments

- Computational Experiments are divided in two different parts
  - Preliminary experiments:
    - Adjust the search parameters
    - Show the merit of the proposed mechanisms
  - Comparison with previous approaches

Computational Experiments

- Comparison of constructive algorithms

Method	Best	Worst	Dev. Best	Time (seconds)
C1	53,9734.5	365,513.4	174.79%	2.25
C2(rand)	313,168.7	279,603.8	16.50%	2.75
C2(0.1)	313,168.7	279,516.6	16.50%	2.50
C2(0.2)	312,248.5	279,331.3	20.12%	2.75
C2(0.3)	312,932.7	279,401.6	19:97%	2.75
C2(0.4)	312,932.7	279,401.6	19:97%	2.75
C2(0.5)	313,168.2	279,344.8	16.50%	2.75
C3(rand)	327,343.5	282,040.0	13.77%	2.75
C3(0.1)	266,755.0	315,570.2	29.04%	2.50
C3(0.2)	266,852.7	322,400.5	31.71%	2.75
C3(0.3)	266,534.5	320,361.7	39.10%	2.75
C3(0.4)	265,768.2	321,176.7	36.76%	2.75
C3(0.5)	262,767.2	324,549.7	37.04%	2.75

Computational Experiments

- Comparison of constructive+improvement method

ºMethod	Best	Worst	Dev. Best
C4	263,636.7	509,565.2	11.26%
C4+EC(1,1)	264,574.5	457,632.0	13.34%
C4+EC(1,5)	263,224.8	327,512.8	13.15%
C4+EC(1,10)	262,739.3	370,197.0	13.19%
C4+EC(1,15)	264,512.5	319,452.0	13.58%
C4+EC(1,20)	262,872.3	316,349.0	13.05%
C4+EC(5,1)	262,025.0	454,472.5	11.45%
C4+EC(5,5)	261,836.8	309,190.0	11.21%
C4+EC(5,10)	263,258.3	303,643.8	11.23%
C4+EC(5,15)	260,298.3	324,966.8	11.14%
C4+EC(5,20)	262,929.3	332,535.3	11.30%
C4+EC(10,1)	261,698.0	302,020.5	10.51%
C4+EC(10,5)	260,345.3	313,030.5	10.37%
C4+EC(10,10)	263,572.3	439,862.0	10.88%
C4+EC(10,15)	262,983.8	311,830.8	10.84%
C4+EC (10,20)	261,869.8	452.744.8	10.91%

Computational Experiments

- GRASP Filter

δ	# Skip	Dev. Best	Time
0	42.7	10.59%	426.0
0.5	51.7	10.51%	292.0
1	37.5	10.40%	550.2
1.5	30.0	10.90%	485.2
2	25.7	10.86%	485.7
2.5	43.2	11.07%	447.0

Computational Experiments

- Path Relinking

pr	best	Dev. Best	Time
5	254,964.5	9,59%	824.5
10	253,773.5	8,51%	1,041.7
15	253,516.5	8,00%	1,268.0
20	252,476.5	8,04%	1,521.7

Computational Experiments

- Comparison of the best methods

	GRASP+PR		C4+HC		SAN		TSSA	
	Val	Dev(%)	Val	Dev(%)	Val	Dev(%)	Val	Dev(%)
randomA1	914882	2.31	950394	6.28	894205	0.00	948868	6.11
randomA2	6572444	0.00	6708192	2.07	6596880	0.37	6625307	0.80
randomA3	14336736	0.00	14463797	0.89	14346700	0.07	14441751	0.73
randomA4	1779181	1.17	1824564	3.75	1758560	0.00	1816732	3.31
randomG4	179138	0.00	206123	15.06	299571	67.23	185912	3.78
bintree10	4267	0.00	13951	226.95	14247	233.89	4440	4.05
hc10	523776	0.00	538116	2.74	540512	3.20	523776	0.00
mesh33x33	32703	0.00	35509	8.58	38481	17.67	33464	2.33
3elt	431737	0.00	1369880	217.30	867560	100.95	509337	17.97
airfoil1	322611	0.00	867560	168.92	1369880	324.62	392989	21.82
whitaker3	1307540	0.00	4857190	271.48	4857190	271.48	1313857	0.48
c1y	65084	4.23	70896	13.54	73867	18.30	62441	0.00
c2y	82665	4.38	89029	12.41	89029	12.41	79199	0.00
c3y	136103	9.66	144902	16.75	163785	31.96	124117	0.00
c4y	125720	9.19	146651	27.36	146651	27.36	115144	0.00
c5y	109279	12.71	122652	26.51	123891	27.79	96952	0.00
gd95c	506	0.00	529	4.55	506	0.00	507	0.20
gd96a	114377	18.83	107945	12.15	111144	15.47	96253	0.00
gd96b	1421	0.35	1527	7.84	1483	4.73	1416	0.00
gd96c	519	0.00	531	2.31	519	0.00	523	0.77
gd96d	2414	0.84	2399	0.21	2421	1.13	2394	0.00
Avg. Time	529.14		552.90		1274.14		870.57	
Avg. Dev.	3.03%		49.89%		55.17%		2.97%	
#Best	11		0		4		9	

Outline

- Introduction
- Constructive Methods
- Improvement Methods
- GRASP Algorithm
- Path Relinking
- Computational Experiments
- Conclusions

Conclusions

- Heuristic procedure based on the GRASP provide high quality solutions to the Linear Arrangement Minimization Problem
- The procedure is coupled with a Path Relinking post-processing for improved outcomes over a long term horizon
- Overall experiments with previously reported instances were performed to
  - Study the contribution of the different elements in our procedure
  - Compare it with previous methods

Conclusions

- Preliminary experiments do show the merit of the proposed mechanisms, such as:
  - The use of ejection chains within GRASP
  - The filter of low quality constructions
  - The selection of diverse solutions for Path Relinking
- We hope other researchers might find effective in other combinatorial optimization problems
- GRASP with Path Relinking implementation is highly effective, rivalling (and surpassing) the best procedures in the literature

GAVAB

Heuristics for the Minimum Linear Arrangement Problem

A. Duarte, J.J. Pantrigo, V. Campos and R. Martí
ETSII - URJC Madrid
Spain

May-11-2008