Simple learning techniques for graph coloring

Jin-Kao Hao

Université d'Angers, France
Email: jin-kao.hao@univ-angers.fr
Web: www.info.univ-angers.fr/pub/hao

Collaborators: Daniel Porumbel, Yangming Zhou, Béatrice Duval, Pascale Kuntz

October 2018, Paris

Outline

1. Introduction
2. Case 1: Multidimensional scaling for search space cartography of graph coloring
3. Case 2: Probability learning based search for grouping problems (graph coloring)

Grouping problems - Graph coloring

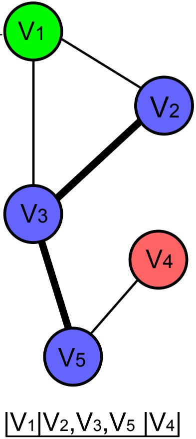
The general coloring problem (COL) (given graph $G(V, E)$):

- Determine the chromatic number χ, i.e. minimum k such that G can be colored with k colors with adjacent vertices (linked by an edge) receiving different colors

The k-coloring problem (k-COL) (given G and k):

- Determine if G can be colored with k colors
- If yes, find a legal coloring with the k given colors

COL can be approximated by a solving a series of k-COL problems



k-Coloring applications and algorithmic difficulty

Application examples

Assigning frequencies in mobile networks, scheduling/timetabling, register allocation in compilers, supply chain management, air traffic flow management and many others

Complexity

- NP-complete for $k > 2$ and one of the most studied combinatorial problems
- Some random graphs with 150 vertices still resist the best exact algorithms
- Cannot be approximated within a constant factor in polynomial time

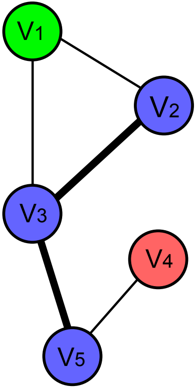
Graph k-coloring - Many heuristics

1. Sequential construction–very fast, not particularly good
2. Local search
   - **Tabu Search** [Hertz & de Werra 1997, Blöchliger et al. 2008]
   - Simulated Annealing [Johnson et al. 1991] and Quantum Annealing (Titiloye & Crispin 2011)
   - Iterated Local Search [Chiarandini & Stützle, 2002], VNS [Avanthay et al. 2003], Variable Search Space [Hertz et al. 2008]
3. **Evolutionary, distributed and hybrid methods**
   combination of local optimization and solution recombination [Morgenstern 1991; Fleurent & Ferland 1996; Dorne & Hao 1998; Galiner & Hao 1999; Malaguti et al 2008; Lü & Hao 2010; Porumbel et al 2010, Moalic & Gondran 2018]
4. **'Reduce and coloring' approach**
   graph reduction by extracting pairwised disjoint large independent sets combined with a coloring algorithm [Wu & Hao 2012; Hao & Wu 2012]

k-coloring with tabu search

Graph k-coloring (given G and k):

- Find a legal coloring with k given colors
- Minimize the number of **edges whose endpoints share the same color** (conflicts)
  - The search space contains all possible k-colorings (legal and illegal colorings)
  - The **objective** is to minimize the number of conflicts
- Start with a conflicting k-coloring and iteratively improve it
  - change the color of a *conflicting* vertex to decrease conflicts
  - use tabu list to avoid cycling of the search process



$|V_1|V_2, V_3, V_5\ |V_4|$

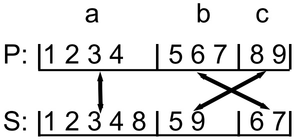
Graph k-coloring search space cartography

Understanding the search space

- Relevant questions about combinatorial search spaces
  - What is the spatial distribution of the local and global optima?
  - Given a local search process, how does its trajectory look like?
  - Which are the regions the process is more likely to explore?
- These issues can be tackled using a **distance metric** to capture the proximity among the configurations of the search space.

Distance between colorings

- **A coloring = a partition** of the vertex set V
  - distance(P, S) = the **minimal number of elements that need to change their class** so as to transform P into S
  - similarity(P, S) = maximum number of shared elements, that do not need to change their class so as to transform P into S



=>

$$\begin{array}{c|ccc} \cap & S_a & S_b & S_c \\ \hline P_a & 4 & 0 & 0 \\ P_b & 0 & 1 & 2 \\ P_c & 1 & 1 & 0 \end{array}$$

=> Similarity = 4+2+1=7
Distance = |V| - 7 = 2

=> 2 class transfers (i.e. 5 and 8) can transform P into S

- The maximum assignment on above matrix can be computed with the Hungarian algorithm in $O(|V| + k^3)$
- And can be done more efficiently in $O(|V|)$ for close partitions (Porumbel-Hao-Kuntz 2011)

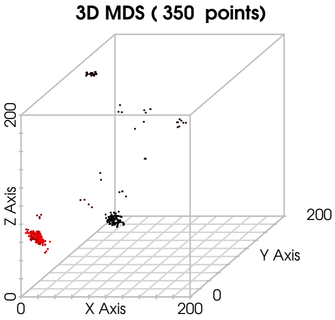
Spatial distribution of the best configurations

Multidimensional Scaling (MDS)–a **data mining tool** visualizing the level of similarity of high dimension elements:

- Projection: *n-dimensional space* $\longrightarrow$ *Euclidean 2D/3D space*
- Euclidean distance between 3D points = **approximation** of the real distance between the associated colorings.

Intuitive representation of best local optima

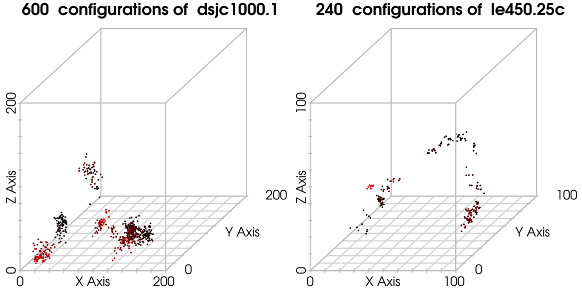
- 350 best local minima represented by Multidimensional scaling
  - $G = dsjc250.5$, $k = 27$ ($k^* = 28$))
- These points form clusters that can be covered by **spheres** of small diameter.



Local search trajectory: MDS representation

We consider Tabu Search process exploring the search space

- we launch TS from a local optimum and we let it explore
- we consider the configurations of high-quality (not worse than the starting point)

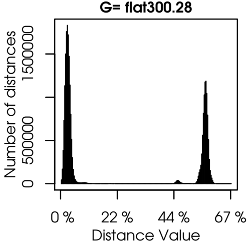


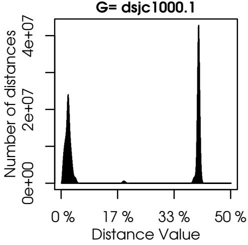
$\Rightarrow$ the visited high-quality colorings are grouped in clusters

Trajectory of long Tabu Search processes

A Tabu Search process explores the search space:

- Record first **40.000 high-quality configurations**
- The **distance histogram** shows the number of (pairs of) configurations distanced by each distance value
- Small distances : configs. in the same cluster
- Large distances : configs. from different clusters
- The small distances are always less than $10\%|V|$
- **The sphere of coloring** C is the set of colorings situated at less than $R = 10\%|V|$ from C





Distance guided local search

We can use the above information for better search **diversification** and **intensification**

1. TS-Div (Tabu Search DIVsified)

$\rightarrow$ Avoid redundant exploration, never visit the same sphere twice

2. TS-Int (Tabu Search INTensified)

$\rightarrow$ In-depth search of a closed perimeter around promising configurations

A new TS-Div algorithm with learning techniques

Typical Local Search

- usually short-sighted with visibility limited to only one step
- no long-term memory

TS-Div

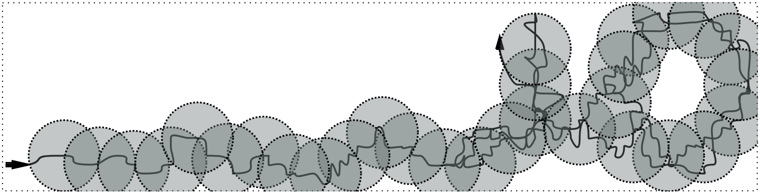
- remember the past, memorize the trajectory
- allowing more time = discover more new regions

Basic principle of TS-Div

1. Memorizing the local search trajectory
   - Complete recording (of each configuration) IMPOSSIBLE

Objective of the new TS-Div algorithm

1. Coarse-grained recording (sphere per sphere) POSSIBLE



2. Avoid already-explored spheres (orient the search toward as-yet-unvisited spheres)

$\Longrightarrow$ Coarse grained Tabu search

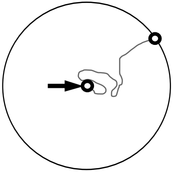
TS-Div : Search process+Learning process

Search process: Tabu Search (TS)

- TS moves from coloring to coloring by changing a color
- At each iteration, TS selects the color change leading to the lowest conflict number
- Each color change has to be not-Tabu, i.e. not performed in the **near past**
  - a move can be re-performed only if it has not been performed during the **last T_ℓ iterations (T_ℓ is called the tabu tenure)**
  - longer T_ℓ = more diversification

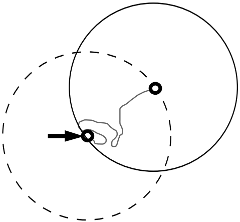
TS-Div : Search process+Learning process

- The learning process records all spheres explored by the search process



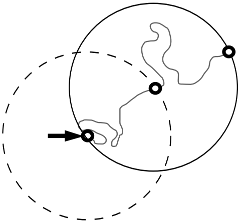
TS-Div : Search process+Learning process

- The learning process records all spheres explored by the search process



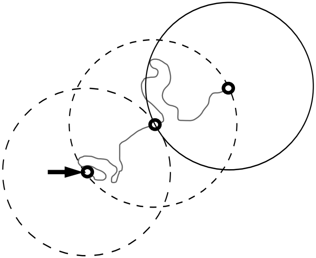
TS-Div : Search process+Learning process

- The learning process records all spheres explored by the search process



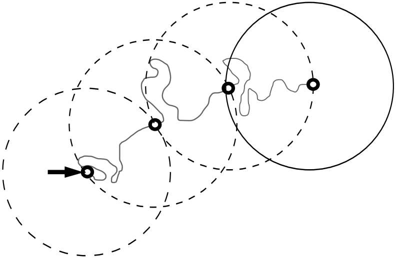
TS-Div : Search process+Learning process

- The learning process records all spheres explored by the search process



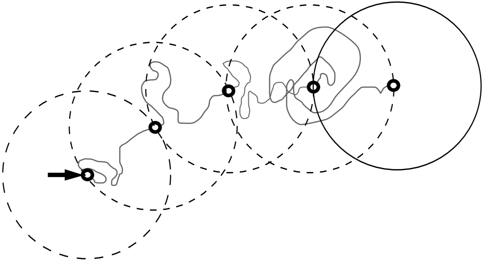
TS-Div : Search process+Learning process

- The learning process records all spheres explored by the search process



TS-Div : Search process+Learning process

- The learning process records all spheres explored by the search process

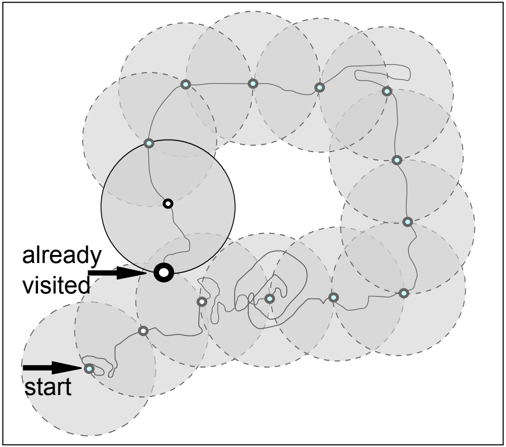


TS-Div : Search process+Learning process

- If TS-Div visits a coloring covered by a visited sphere:

  $\Longrightarrow$

  - increment Tabu list length
  - the search process is FORCED towards other spheres

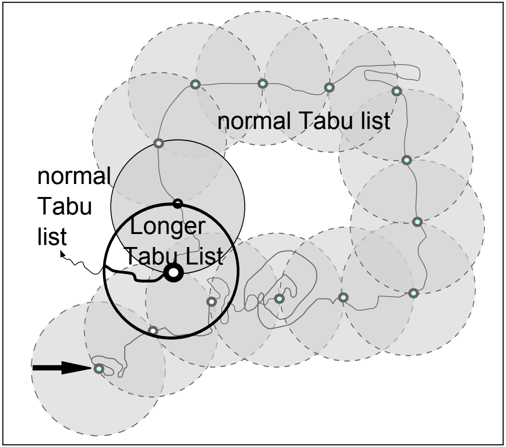


TS-Div : Search process+Learning process

- If TS-Div visits a coloring covered by a visited sphere:

  $\Longrightarrow$

  - increment Tabu list length
  - the search process is FORCED towards other spheres



TS-Int

Intensification issues

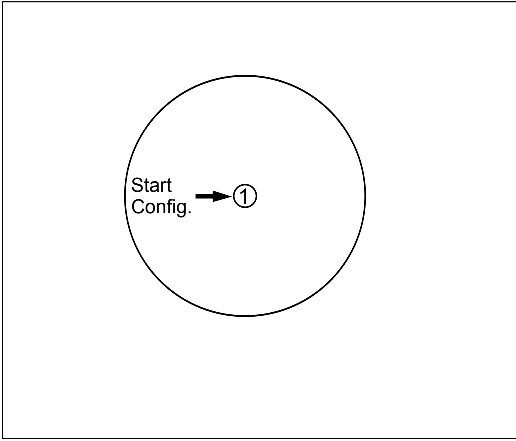
- TS-Div aims to diversify, to go only once through each sphere
- One traversal per sphere can result in missing a "hidden" solution inside that sphere

Objective of TS-Int

- "in-depth" examination of a close perimeter around a **given starting point**
- find any solution from a starting configuration (a sphere center)

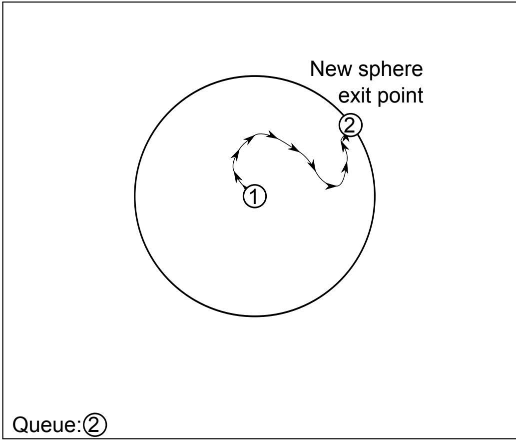
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



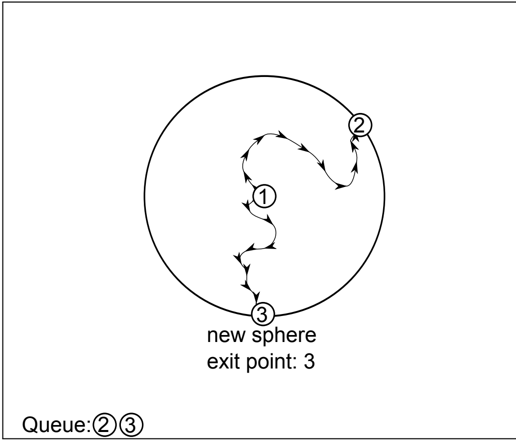
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



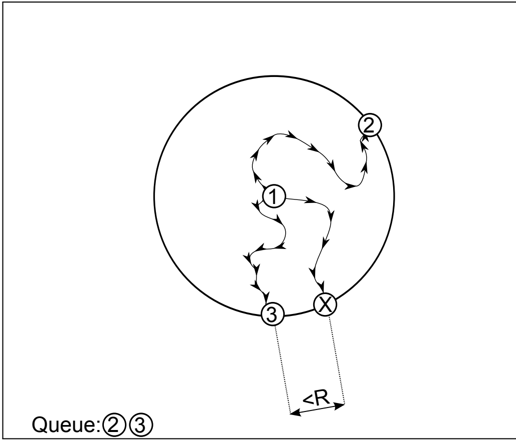
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



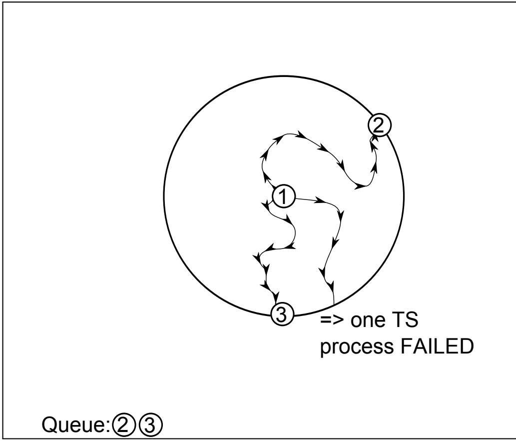
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



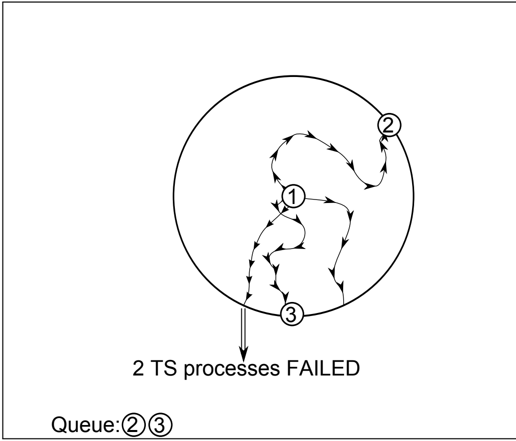
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere “from all angles”
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next “sphere exit point” as start configuration and REPEAT step 1



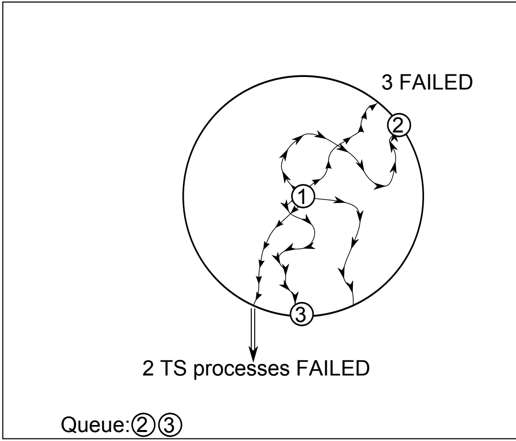
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



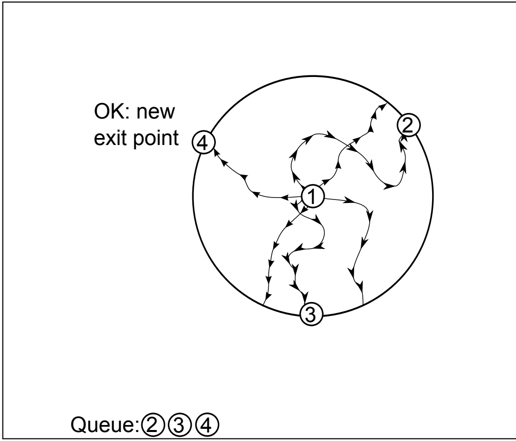
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



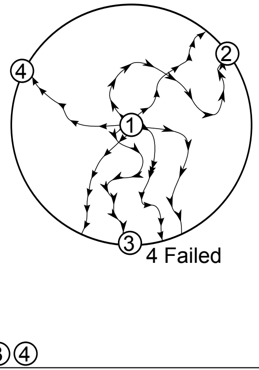
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



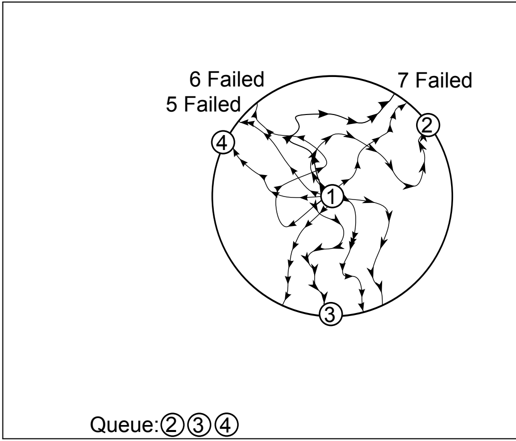
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



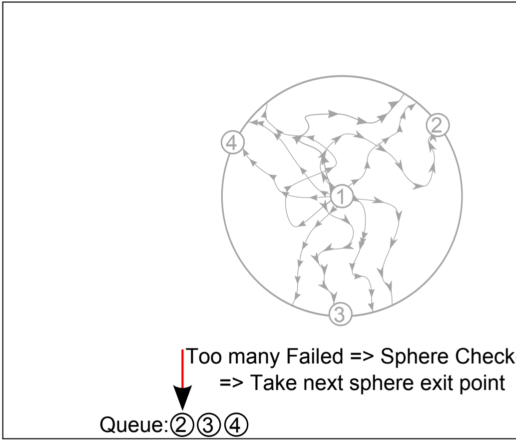
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



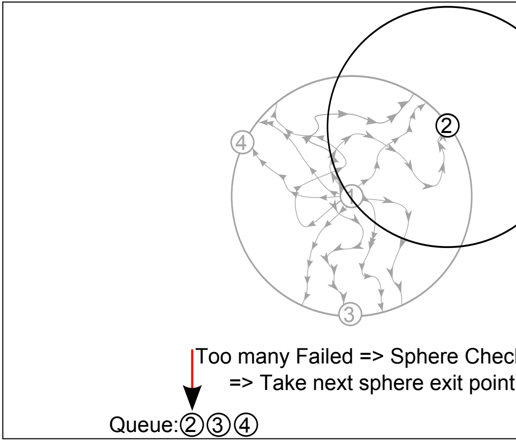
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



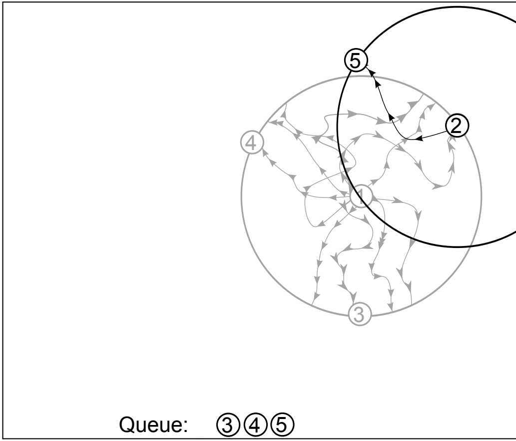
TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



TS-Int

1. Given a start configuration, launch (iteratively) numerous TS processes to explore its sphere "from all angles"
   - Each TS process is stopped when it gets out of the sphere
   - When **enough** processes launched, the sphere is considered **clear** (of better configurations)
2. Take the next "sphere exit point" as start configuration and REPEAT step 1



Selective results of TS-Div/TS-Int

Graphe	k^*	TS–Div + TS–Int	[1]	[2]	[3]	[4]	[5]	[6]	[7]
		2010	2008	2008	2008	1993	1999	2008	2010
*dsjc*1000.1	20	*20*	20	20	20	21	20	20	20
*dsjc*1000.5	83	85	87	88	84	88	83	83	83
*dsjc*1000.9	224	**223**	224	225	224	226	224	225	223
*flat*300.28	28	*28*	28	28	31	31	31	31	29
*flat*1000.76	82	85	86	87	84	89	83	82	82
*le*450.25*c*	25	*25*	26	25	26	25	26	25	25
*le*450.25*d*	25	*25*	26	25	26	25	26	25	25

[1] Hertz et. al. Variable space search for graph coloring, [2] Blöchliger & Zufferey. A graph coloring heuristic using partial solutions and a reactive tabu scheme, [3] Galinier et. al. An adaptive memory algorithm for the k-coloring problem, [4] C. Morgenstern. Distributed coloration neighborhood search (DIMACS), [5] Galinier & Hao. Hybrid evolutionary algorithms for graph coloring, [6] Malaguti et. al. A Metaheuristic Approach for the Vertex Coloring Problem, [7] Lü & Hao. A memetic algorithm for graph coloring

In Column 2, k^* is the best upper bound at the moment when our article was accepted by Computers & O.R.

Other applications of the space cartography

1. Informed (Multi-Parent) Graph Coloring Crossover operator

$\rightarrow$ Converge rapidly towards high-quality individuals/configurations

2. Method for a **Strict** Control of Population Diversity

$\rightarrow$ Keep individuals distanced at all times, avoid premature convergence

$\Rightarrow$ Algorithm <u>Evo-Div</u> (<u>Evo</u>lutionary Algorithm with <u>Div</u>ersity Guarantee)

Grouping problems

Given a set V of n distinct items, a grouping problem is to partition the items into k different groups g_i $(i = 1, \ldots, k)$ (k can be fixed or vary), while taking into account specific constraints and optimization objective.

- Problems with fixed k groups
  - Graph k-coloring
  - Graph k-way partitioning
  - ...
- Problems with variable groups
  - Graph coloring variants (sum coloring...)
  - Bin-packing
  - Clustering
  - ...

Group naming may or may not be relevant–e.g. irrelevant for graph coloring while relevant for sum coloring.

Learning based search for grouping problems

We develop a probability learning approach for grouping problems inspired by reinforcement learning

- use a probability matrix to learn
  - which element should go to which group
  - which elements should stay together
- use a (basic) optimization procedure for solution improvement

Probability matrix

We use a probability matrix P of size $n \times k$ (n - number of items and k - number of groups) where p_{ij} is the probability that the i-th item v_i selects the j-th group g_j, initialized to $1/k$, i.e., each item selects each group with equal probability.

	g_1	g_2	...	g_k
v_1	p_{11}	p_{12}	...	p_{k1}
v_2	p_{21}	p_{22}	...	p_{k2}
...	...	...	...	...
v_n	p_{n1}	p_{n2}	...	p_{nk}

Figure: Probability matrix P

General scheme

Composed of four keys components: a (descent-based) local search procedure, a group selection strategy, a probability learning mechanism, and a probability smoothing technique.

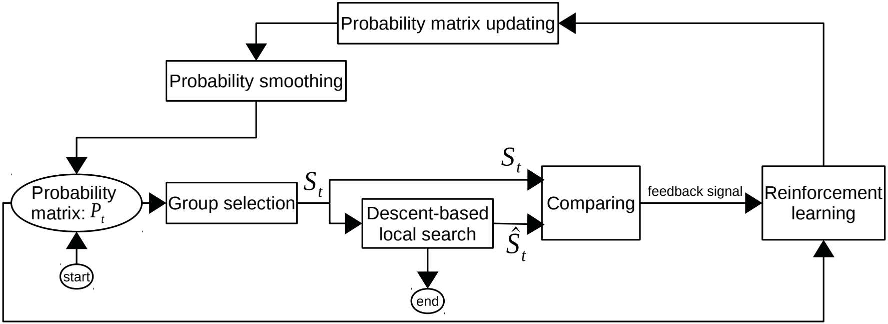


Figure: A schematic diagram of PLS for grouping problems. From a starting solution generated according to the probability matrix, the process iteratively runs until its stopping condition is met.

Group selection – assign items to groups

Given the Probability matrix P, an item can select its group according to different strategies:

- Greedy selection: always select the group g_j such that the associated probability p_{ij} has the maximum value. This strategy is intuitively reasonable, but may cause the algorithm to be trapped rapidly.
- Roulette wheel selection: the chance for an item v_i to select group g_j is proportional to p_{ij}. Thus a group with a large (small) probability has more (less) chance to be selected.
- Hybrid selection: with a noise probability ω, select the group randomly; with probability $1 - \omega$, apply greedy selection.

Experiments show that hybrid selection performs the best.

Optimization algorithm for solution improvement

Any method can be applied. In our case, we used both a steepest descent local search and a simple tabu search algorithm

Probability updating and smoothing

Given a starting solution S_t and an improved solution $\hat{S}_t$.

- If item v_i keeps its original group (say g_u), reward, with reward factor α, the group g_u and update its probability vector p_i

$$p_{ij}(t+1) = \begin{cases} \alpha + (1-\alpha)p_{ij}(t) & j = u \\ (1-\alpha)p_{ij}(t) & \text{otherwise.} \end{cases} \tag{1}$$

- If item v_i moves from its original group g_u in S_t to a new group (say g_v, $v \neq u$), penalize, with penalization factor β, the discarded group g_u, compensate, with compensation factor γ, the new group g_v and update its probability vector p_i

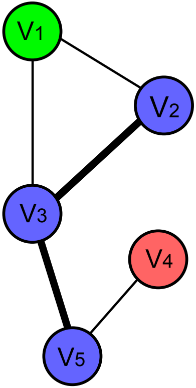
$$p_{ij}(t+1) = \begin{cases} (1-\gamma)(1-\beta)p_{ij}(t) & j = u \\ \gamma + (1-\gamma)\frac{\beta}{k-1} + (1-\gamma)(1-\beta)p_{ij}(t) & j = v \\ (1-\gamma)\frac{\beta}{k-1} + (1-\gamma)(1-\beta)p_{ij}(t) & \text{otherwise.} \end{cases} \tag{2}$$

Probability smoothing: reduce the the probabilities occasionally to forget some old decisions.

Applied to graph k-coloring

Graph k-coloring (given G and k):

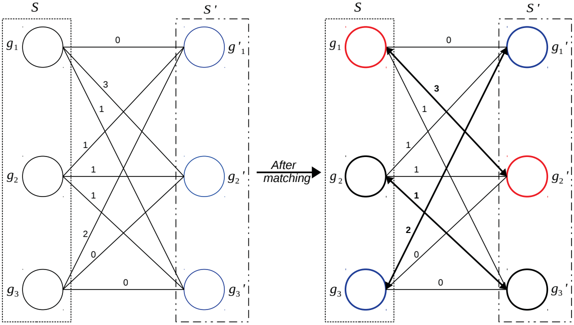
- Find a legal coloring with k given colors
- Minimize the number of **edges whose endpoints share the same color** (conflicts)
  - The search space contains all possible k-colorings (legal and illegal colorings)
  - The **objective** is to minimize the number of conflicts



| V_1 | V_2, V_3, V_5 | V_4 |

Applied to *k*-coloring - Group matching of *k*-colorings

- The numberings of the groups in a coloring are irrelevant and interchangeable.
- Identifying the group correspondences between two solutions (starting solution and improved solution) by a maximum weight matching on a complete bipartite graph with the Hungarian algorithm



(a). a complete bipartite graph

(b). a maximum weight matching

Solution optimization with tabu search (TabuCOL)

Starting with a conflicting k-coloring, TabuCol (Hertz & De Werra 1987, Dorne & Hao 1999, Galinier & Hao 1999) improves iteratively the solutions

- change the color of a *conflicting* vertex such that the resulting coloring minimizes the number of conflicts
- tabu list to forbid the vertex to receive the lost color during tl iterations (to avoid cycling of the search process)

Computational results

Tested on difficult DIMACS Challenge benchmark graphs
Compared with TabuCOL

- Dominate TabuCOL: better results and less computing time

Compared with 10 state of the art coloring algorithms (among MANY algorithms)

- Generally better than local search algorithms
- Competitive with several complex hybrid methods

Computational results

Comparison with TabuCOL on difficult DIMACS graphs.

		PLSCOL					TabuCOL				
Instance	χ/k^*	k	#succ	#gen	#iter	time(s)	k	#succ	#gen	#iter	time(s)
DSJC250.5	?/28	**28**	10/10	3	4.0×10^5	4	**28**	10/10	58	1.1×10^7	102
DSJC500.1	?/12	**12**	07/10	69	7.5×10^6	43	**12**	10/10	8936	1.9×10^9	12808
DSJC500.5	?/47	**48**	03/10	761	7.9×10^7	1786	49	06/10	1122	2.5×10^8	5543
DSJC500.9	?/126	**126**	10/10	187	2.4×10^7	747	127	10/10	362	9.2×10^7	2704
DSJC1000.1	?/20	**20**	01/10	369	2.9×10^8	3694	21	10/10	2	3.7×10^5	4
DSJC1000.5	?/83	**87**	10/10	203	2.7×10^7	1419	89	02/10	492	1.4×10^8	7031
DSJC1000.9	?/222	**223**	05/10	2886	3.1×10^8	12094	229	05/10	270	1.0×10^8	9237
DSJR500.1c	?/85	**85**	10/10	317	3.2×10^7	386	**85**	10/10	554	8.3×10^7	1825
DSJR500.5	?/122	**126**	08/10	464	7.3×10^7	1860	127	01/10	2,701	4.3×10^8	8592
le450_15c	15/15	**15**	07/10	2883	2.8×10^8	1718	**15**	10/10	155	2.1×10^7	238
le450_15d	15/15	**15**	03/10	2787	2.8×10^8	2499	**15**	10/10	766	1.1×10^8	1314
le450_25c	25/25	**25**	10/10	1968	2.0×10^8	1296	26	10/10	1	8.1×10^4	1
le450_25d	25/25	**25**	10/10	2110	2.2×10^8	1704	26	10/10	1	1.1×10^5	2
flat300_26_0	26/26	**26**	10/10	49	4.9×10^6	195	**26**	10/10	31	5.1×10^6	254
flat300_28_0	28/28	**30**	10/10	147	1.5×10^7	233	31	10/10	95	1.9×10^7	242
flat1000_76_0	76/81	**86**	01/10	908	1.1×10^8	5301	89	02/10	339	9.1×10^7	3709
R250.5	?/65	**66**	10/10	865	1.1×10^8	705	**66**	01/10	1793	2.3×10^8	2038
R1000.1c	?/98	**98**	10/10	88	9.1×10^6	256	**98**	10/10	110	2.0×10^7	702
R1000.5	?/234	**254**	04/10	189	3.7×10^7	7818	260	10/10	1	3.1×10^5	124
latin_square_10	?/97	**99**	08/10	666	6.7×10^7	2005	103	10/10	444	9.7×10^7	7769

Computational results

Comparison with 10 reference algorithms on difficult DIMACS graphs

		local search algorithms				population-based algorithms						
		PLSCOL	IGrAl	VSS	Partial	HEA	AMA	MMT	Evo-Div	MA	QA	HEAD
Instance	χ / k^*	k_{best}	2008	2008	2008	1999	2008	2008	2010	2010	2011	2015
DSJC250.5	?/28	**28**	29	*	*	*	**28**	**28**	*	**28**	**28**	**28**
DSJC500.1	?/12	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**
DSJC500.5	?/47	48	50	48	48	48	48	48	48	48	48	**47**
DSJC500.9	?/126	**126**	129	**126**	127	**126**	**126**	127	**126**	**126**	**126**	**126**
DSJC1000.1	?/20	**20**	22	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
DSJC1000.5	?/82	87	94	86	89	83	84	84	83	83	83	**82**
DSJC1000.9	?/222	223	239	224	226	224	224	225	223	223	**222**	**222**
DSJR500.1c	?/85	**85**	**85**	**85**	**85**	*	86	**85**	**85**	**85**	**85**	**85**
DSJR500.5	?/122	126	126	125	125	*	127	**122**	**122**	**122**	**122**	*
le450_15c	15/15	**15**	16	**15**	**15**	**15**	**15**	**15**	*	**15**	**15**	*
le450_15d	15/15	**15**	16	**15**	**15**	**15**	**15**	**15**	*	**15**	**15**	*
le450_25c	25/25	**25**	27	**25**	25	26	26	**25**	**25**	**25**	**25**	**25**
le450_25d	25/25	**25**	27	**25**	25	26	26	**25**	**25**	**25**	**25**	**25**
flat300_26_0	26/26	**26**	*	*	*	*	**26**	**26**	*	**26**	*	*
flat300_28_0	28/28	30	*	**28**	**28**	31	31	31	31	29	31	31
flat1000_76_0	76/81	86	*	85	87	83	84	83	82	82	82	**81**
R250.5	?/65	66	*	*	66	*	*	**65**	**65**	**65**	**65**	**65**
R1000.1c	?/98	**98**	*	*	*	*	*	**98**	**98**	**98**	**98**	**98**
R1000.5	?/234	254	*	*	248	*	*	**234**	238	245	238	245
latin_square_10	?/97	99	100	*	*	*	104	101	100	99	**98**	*

Conclusions

- Most presented techniques can be applied to solve other optimization problems
- Data mining and learning can boost existing optimization methods
- Data mining and learning can help create new optimization methods
- There is still much to be done and explored

Related papers

- D.C. Porumbel, J.K. Hao, P. Kuntz. A search space cartography for guiding graph coloring heuristics. Computers & Operations Research 37(4):769–778, 2010.
- D.C. Porumbel, J.K. Hao, P. Kuntz. An efficient algorithm for computing the distance between close partitions. Discrete Applied Mathematics 159:53–59, 2011.
- Y. Zhou, J.K. Hao, B. Duval. Reinforcement learning based local search for grouping problems: A case study on graph coloring. Expert Sys. with App. 64:412–422, 2016.
- Y. Zhou, B. Duval, J.K. Hao. Improving probability learning based local search for graph coloring. Applied Soft Computing, 65:542–553, 2018.
- Y. Zhou, J.K. Hao, B. Duval. When data mining meets optimization: A case study on the quadratic assignment problem. arXiv:1708.05214, Aug. 2017.
- Y. Zhou, J.K. Hao, B. Duval. Opposition-based memetic search for the maximum diversity problem. IEEE Transactions on Evolutionary Computation, 25(1):731–745, 2017.
- F. Glover and J.K. Hao. Diversification-based learning in computing and optimization. Journal of Heuristics, 2018.

Thank you!