Multilevel Refinement for Combinatorial Optimisation Problems

C. Walshaw
Computing & Mathematical Sciences,
University of Greenwich, London
www.gre.ac.uk/∼c.walshaw
C.Walshaw@gre.ac.uk

Contents

- examples:
  - the graph partitioning problem
  - the travelling salesman problem
  - the graph colouring problem
- why does it work?
- generic multilevel refinement
- review of experimental results
- iterated multilevel algorithms
- conclusions & further work

aim: to give insight into the multilevel paradigm for combinatorial problems
restriction: summary of my experiences, not comprehensive review

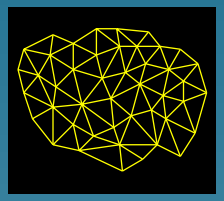
The graph partitioning problem (GPP)

partition the vertices into k sets such that

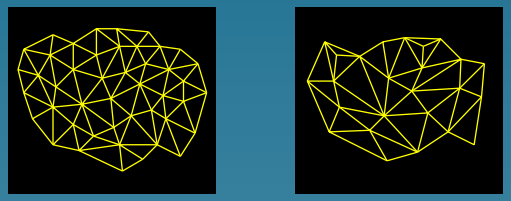
- *– same number of vertices in each set*
- *– the number of edges cut is minimised*

- NP-hard *(cannot be solved in polynomial time)*
- need to use heuristics
- applications in VLSI & parallel processing

Multilevel partitioning

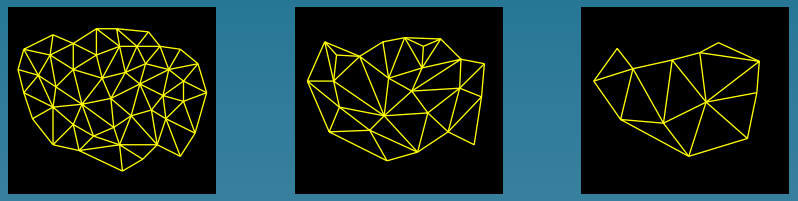


Multilevel partitioning



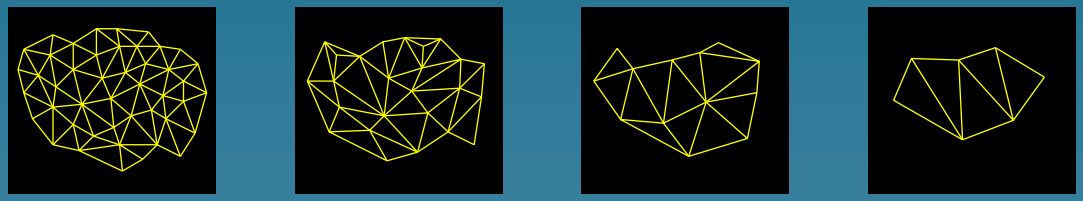
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*

Multilevel partitioning



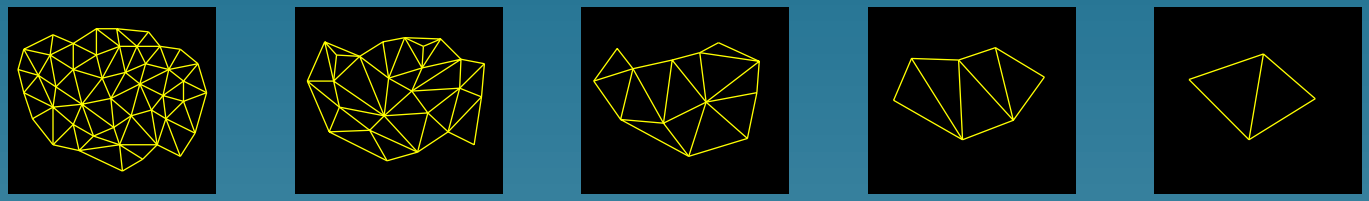
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*

Multilevel partitioning



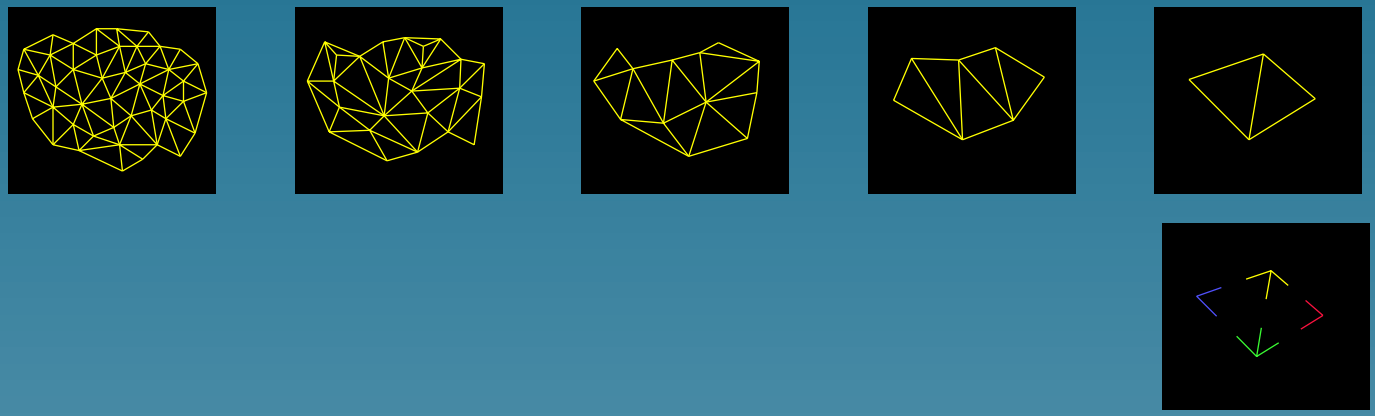
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*

Multilevel partitioning



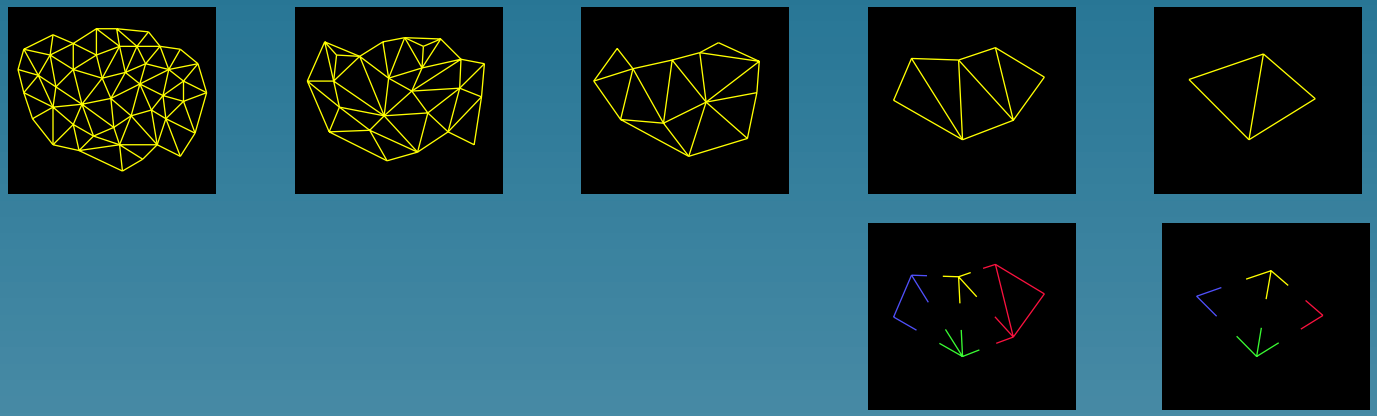
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*

Multilevel partitioning



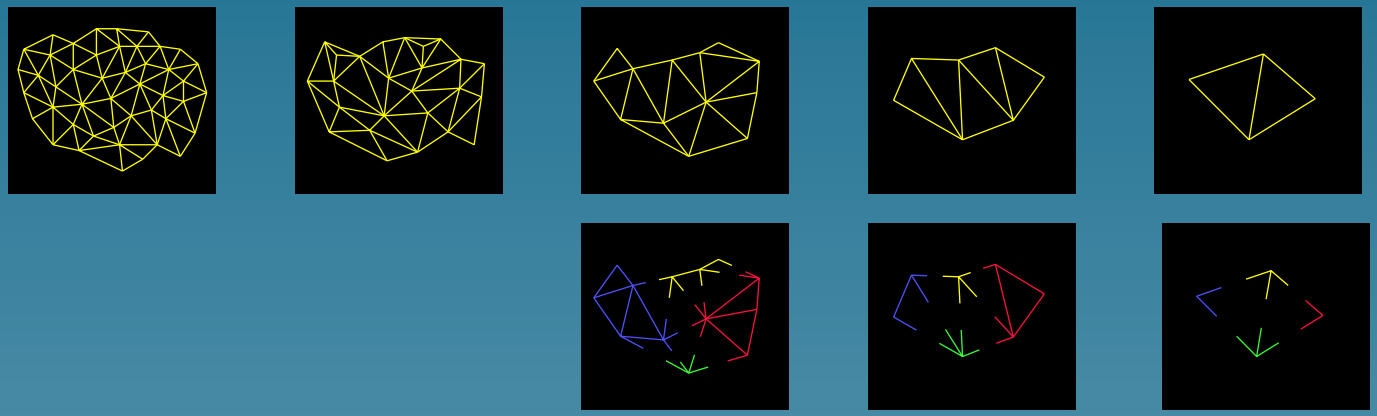
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*
- initial partition of coarsest graph *(very fast)*

Multilevel partitioning



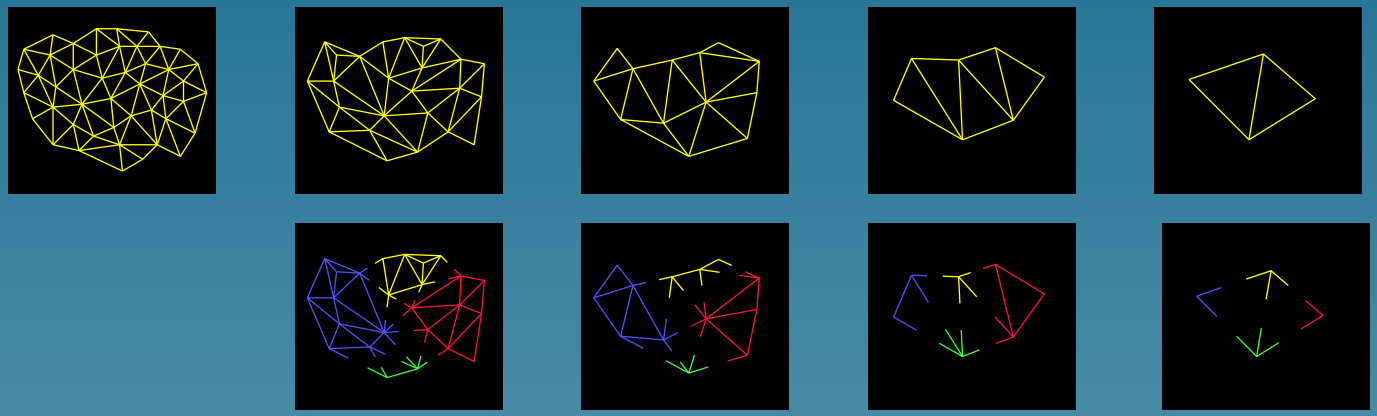
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*
- initial partition of coarsest graph *(very fast)*
- refine partition on each graph *(using local search)*

Multilevel partitioning



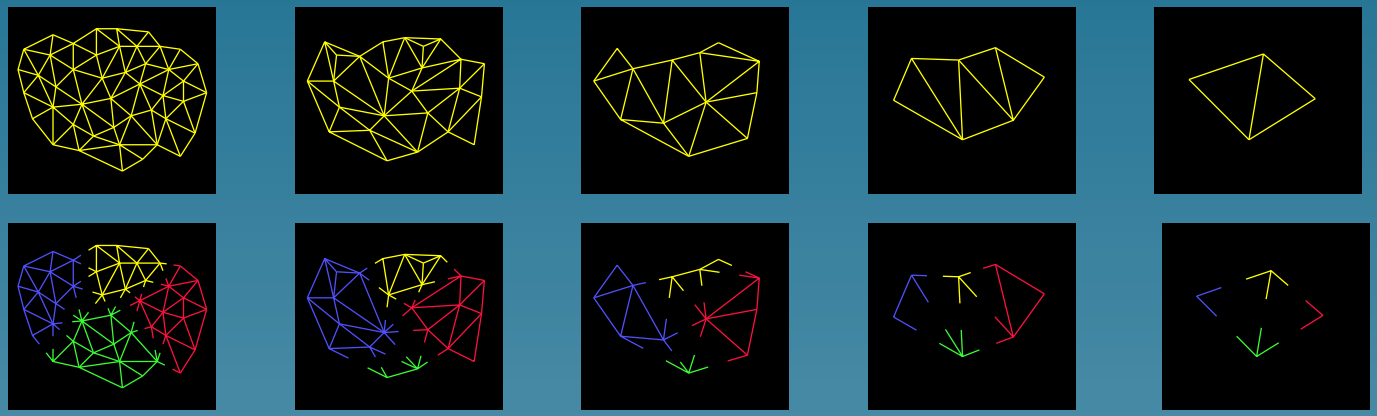
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*
- initial partition of coarsest graph *(very fast)*
- refine partition on each graph *(using local search)*

Multilevel partitioning



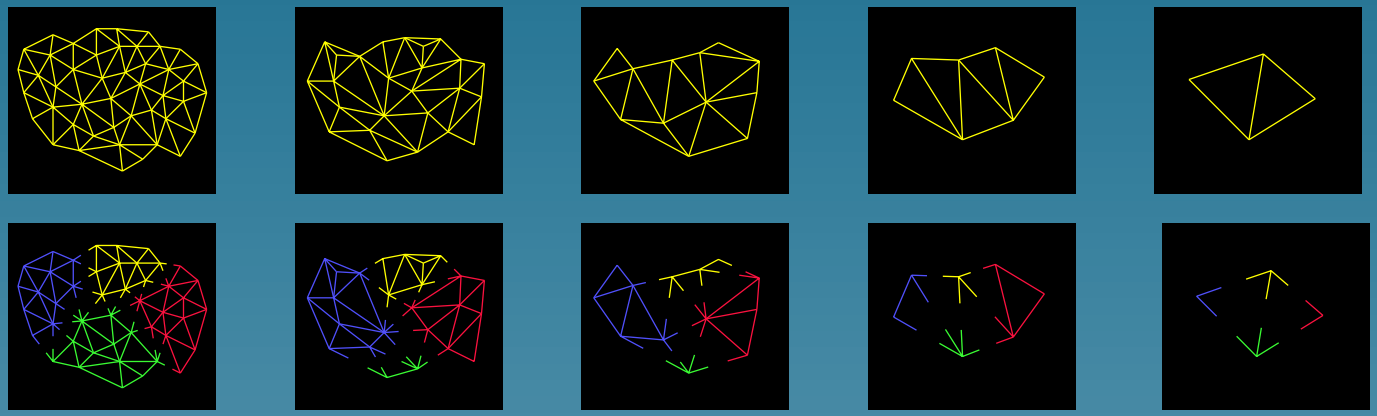
- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*
- initial partition of coarsest graph *(very fast)*
- refine partition on each graph *(using local search)*

Multilevel partitioning



- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*
- initial partition of coarsest graph *(very fast)*
- refine partition on each graph *(using local search)*

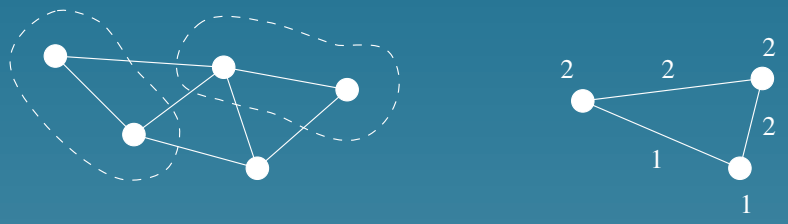
Multilevel partitioning



- coarsen problem $\Rightarrow$ hierarchy of graphs *(using vertex matching)*
- initial partition of coarsest graph *(very fast)*
- refine partition on each graph *(using local search)*

[Bui & Jones, 1993; Hendrickson & Leland, 1995]

Graph coarsening



Visit randomly ordered vertex list and match with ‘best’ neighbour

Matching algorithms:

- random edge matching *[Hendrickson & Leland, 1995]*
- heavy edge matching *[Karypis & Kumar, 1998]*
- locally optimal (heavy edge) matching *[Monien* et al.*, 2000]*

Refinement

Typically a variant of Kernighan-Lin algorithm *[Kernighan & Lin, 1970]*

- local search technique
- limited ability to escape local minima

Refinement

Typically a variant of Kernighan-Lin algorithm *[Kernighan & Lin, 1970]*

- local search technique
- limited ability to escape local minima

In principle can use any iterative refinement method, e.g.

- multilevel simulated annealing *[Vanderstraeten et al., 1996]*
- multilevel tabu search *[Vanderstraeten et al., 1996]*
- multilevel reactive tabu search *[Battiti et al., 1999]*
- multilevel genetic algorithm *[Kaveh et al., 2000]*
- multilevel ant colony optimisation *[Langham & Grant, 1999]*

GPP/GCP test suites

sparse suite (commonly used for testing partitioning schemes)

- 16 graphs, mostly mesh-based
- $|V| = 2395, \ldots, 17758; \Delta < 1\%$
- `http://www.gre.ac.uk/~c.walshaw/partition`

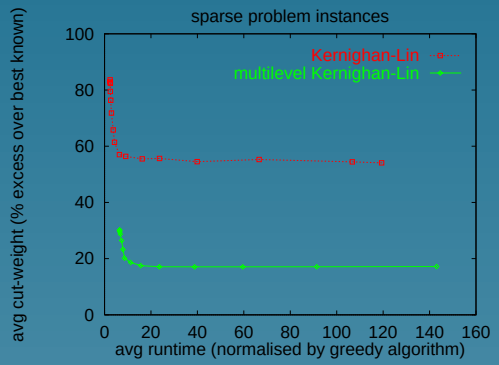
colouring suite (partially assembled for 2nd DIMACS challenge)

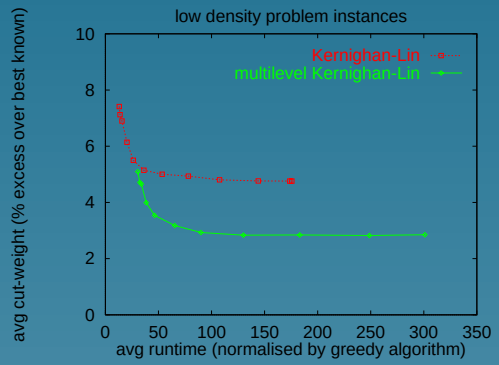
- 90 graphs for testing graph colouring schemes
- $|V| = 11, \ldots, 4000$
- 3 density-based subclasses

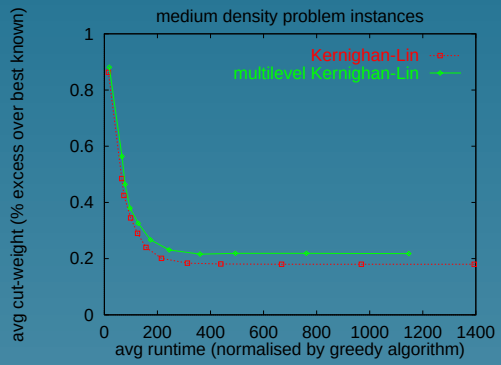
low:	0%	$\leq \Delta <$	33%;	58 graphs
medium:	33%	$\leq \Delta <$	66%;	23 graphs
high:	66%	$\leq \Delta \leq$	100%;	9 graphs

- `http://mat.gsia.cmu.edu/COLOR/instances.html`

GPP results







- spectacular success for sparse/low-densities
- not so successful for medium/high-densities

Variant problems

- partitioning for subdomain aspect ratio (shape)
- mapping to heterogeneous networks
  change in **local** *search gain function*
  $\Rightarrow$ **global** *changes to subdomain topology*
- dynamic load balancing
  tradeoff data migrated vs. solution quality
  using coarsening threshold

$\Rightarrow$ multilevel techniques add ‘global’ perspective
$\Rightarrow$ multilevel techniques are very flexible

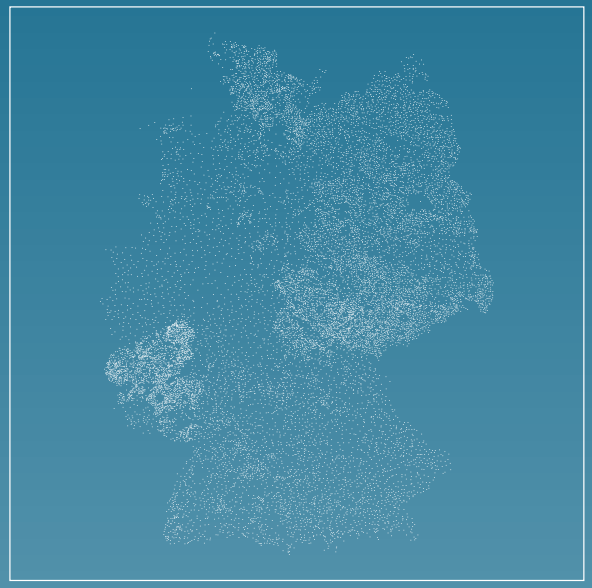
The travelling salesman problem (TSP)

find the shortest tour which visits a collection of 'cities' and returns to its starting point

- NP-hard
- good lower bound on solution quality *[Held & Karp, 1970]*
- applications in VLSI & logistics

- most studied combinatorial problem *[Johnson & McGeoch, 1997]*
- dominated by 'world champion' heuristics
  - Lin-Kernighan (1973-1989) *[Lin & Kernighan, 1973]*
  - chained Lin-Kernighan (1989—) *[Martin et al., 1991]*

d15112



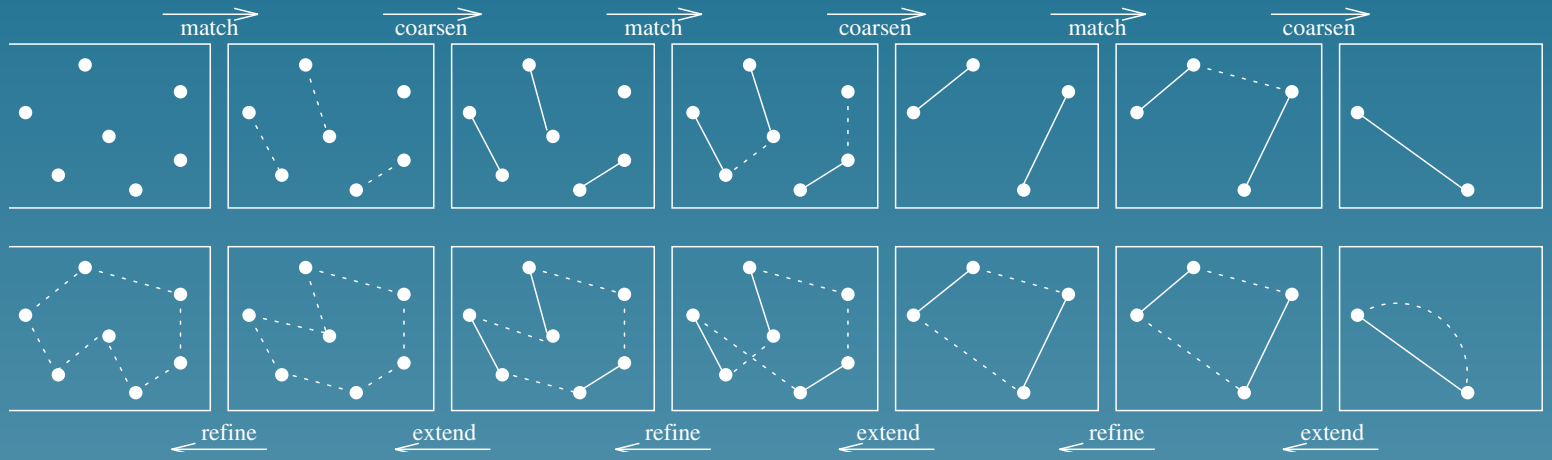
- largest solved TSP problem (April 2001)
- 15,112 cities (Germany)
- ∼22.6 years of CPU time (500 MHz)

d15112 tour

usa13509

- solved TSP problem (1998)
- 13,509 cities (USA, pop. > 500)

A multilevel algorithm for the TSP



- match locally — *increase locality after each coarsening*
- fix edges — *collapse chains of fixed edges*
- refinement: chained Lin-Kernighan

[Walshaw, 2000]

TSP test suite

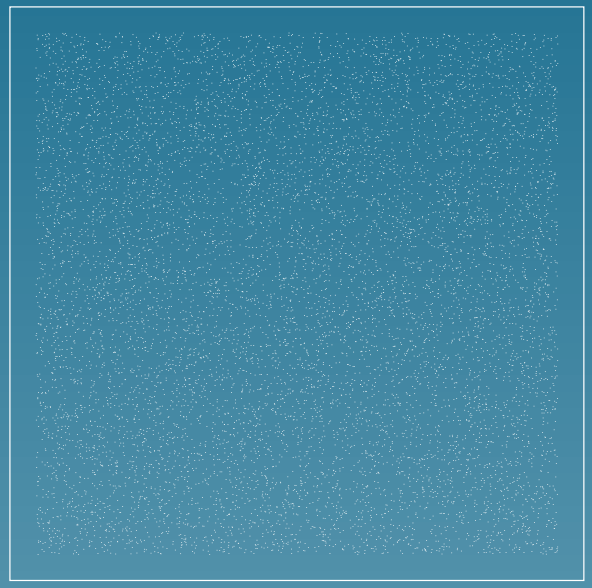
- 80 problems (subset of 90 from 8th DIMACS challenge)
- $|V|$ = 1,000, . . . , 1,000,000
- 3 subclasses
  - `TSPLIB` *(including real-life instances)*
  - random uniformly distributed instances
  - randomly clustered instances

    considered harder to solve than uniformly distributed
- `http://www.research.att.com/~dsj/chtsp/`

pla33810

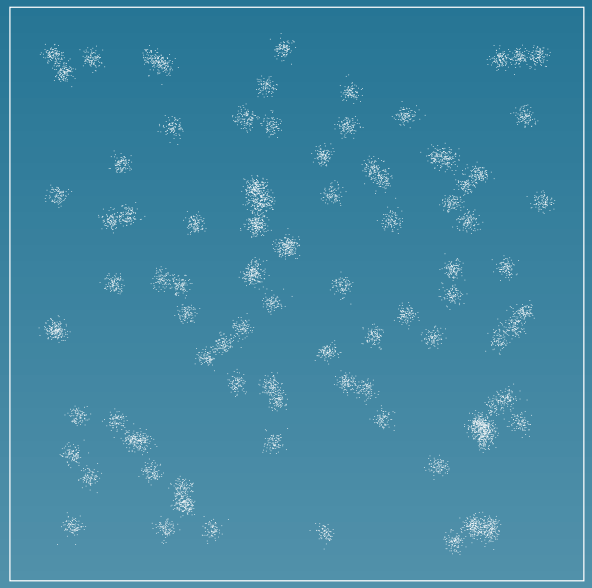
- unsolved TSP problem
- 33,810 vertices (programmed logic array)

E10k.0



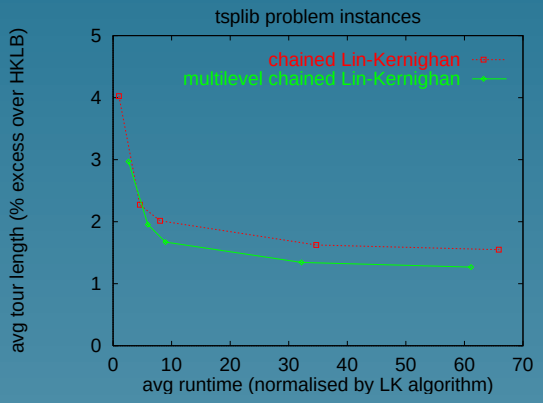
- 10,000 cities
- randomly generated
- uniform distribution

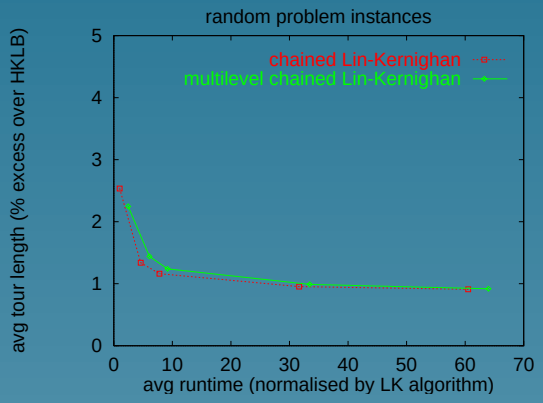
C10k.0

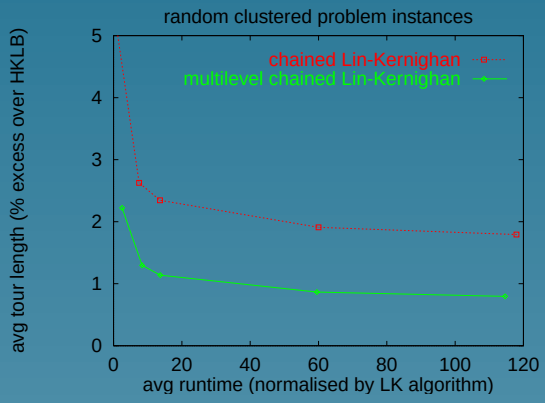


- 10,000 cities
- randomly generated
- clustered distribution

TSP results







- good for (real-life) `TSPLIB`
- OK for random uniform instances
- spectacular success for randomly clustered

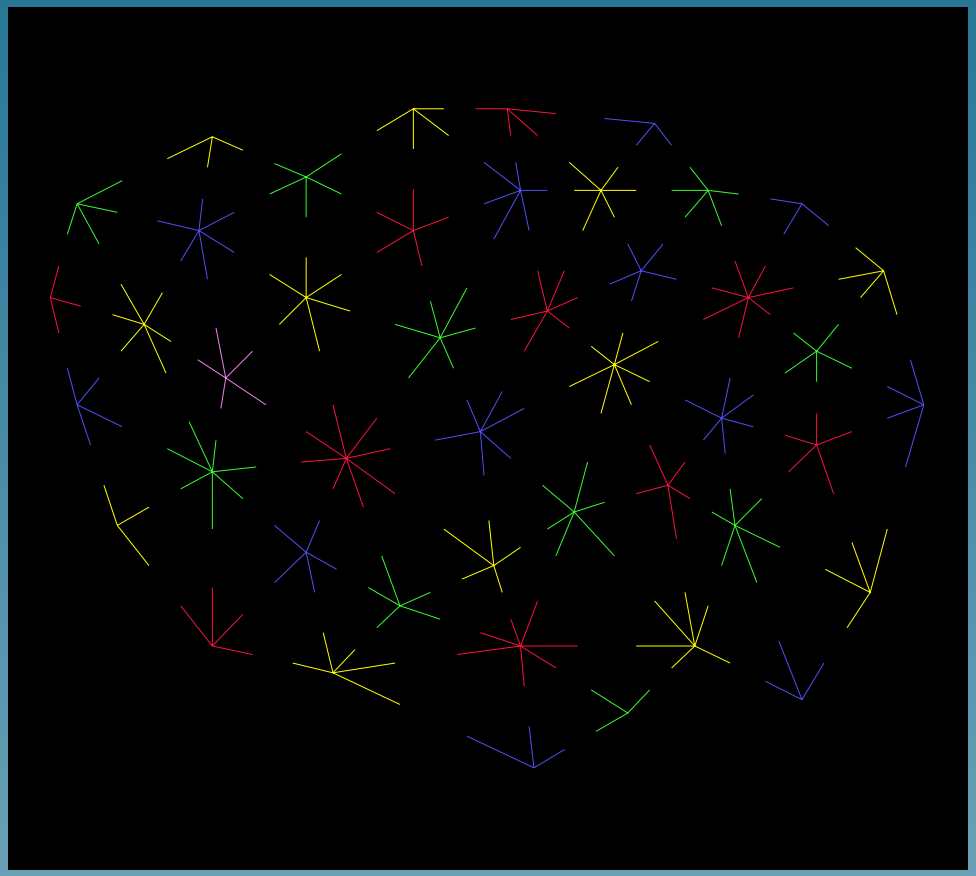
The graph colouring problem (GCP)

assign a colour to each vertex such that

- *– adjacent vertices have different colours*
- *– the number of colours is minimised*

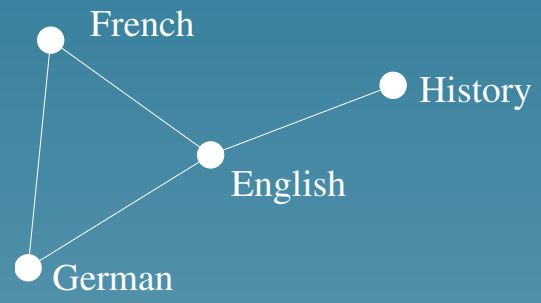
- NP-hard & considered very challenging
- difficult for local search *(large 'plateaus' in cost function)*
- applications in timetabling & scheduling

GCP example



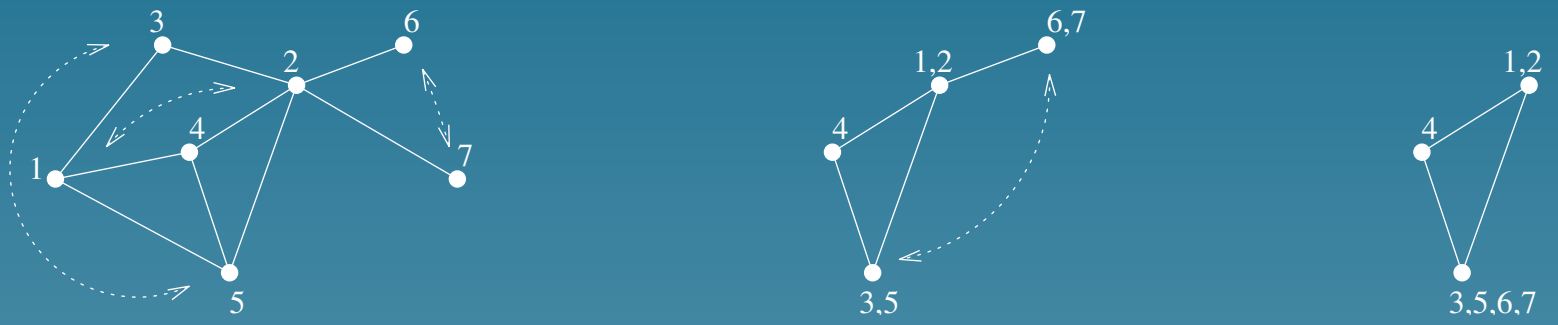
GCP example application: timetabling

	English	French	German	History
Andrew	X			X
Barbara		X	X	
Carol	X	X		
David	X		X	



- each vertex represents a class
- edges between classes which cannot run concurrently
- # sessions required = # colours in graph

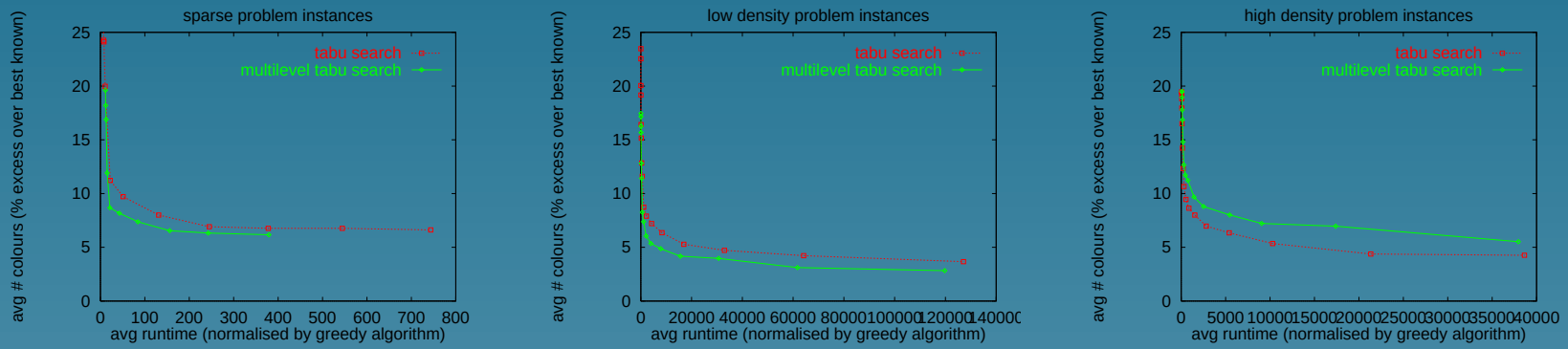
A multilevel algorithm for the GCP



- similar coarsening to multilevel partitioning
- match with vertices which are *not* adjacent
  can extend colouring without conflicts
- refinement: tabu search (or iterated greedy)

[Walshaw, 2001]

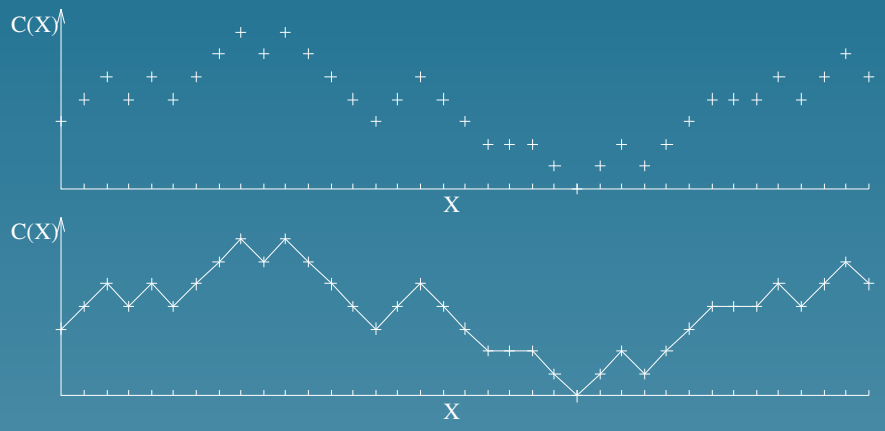
GCP results



- good for sparse/low densities
- not so successful for medium/high-densities

Why does it work?

Review of cost function/solution space



- Typical problem:
  minimise cost (objective) function $C(x)$ (subject to some constraints)
- Let $\mathcal{X} =$ set of all possible solutions *(solution space)*
- can plot $C(x)$ against x **but** $\mathcal{X}$ is unordered
- local search neighbourhood defines relations between solutions
- known as a *landscape* (valleys, mountains)

Why does it work?

- simple answer:
  each coarsened problem approximates its parent ($P_l \approx P_{l-1}$)

Why does it work?

- simple answer:
  each coarsened problem approximates its parent ($P_l \approx P_{l-1}$)

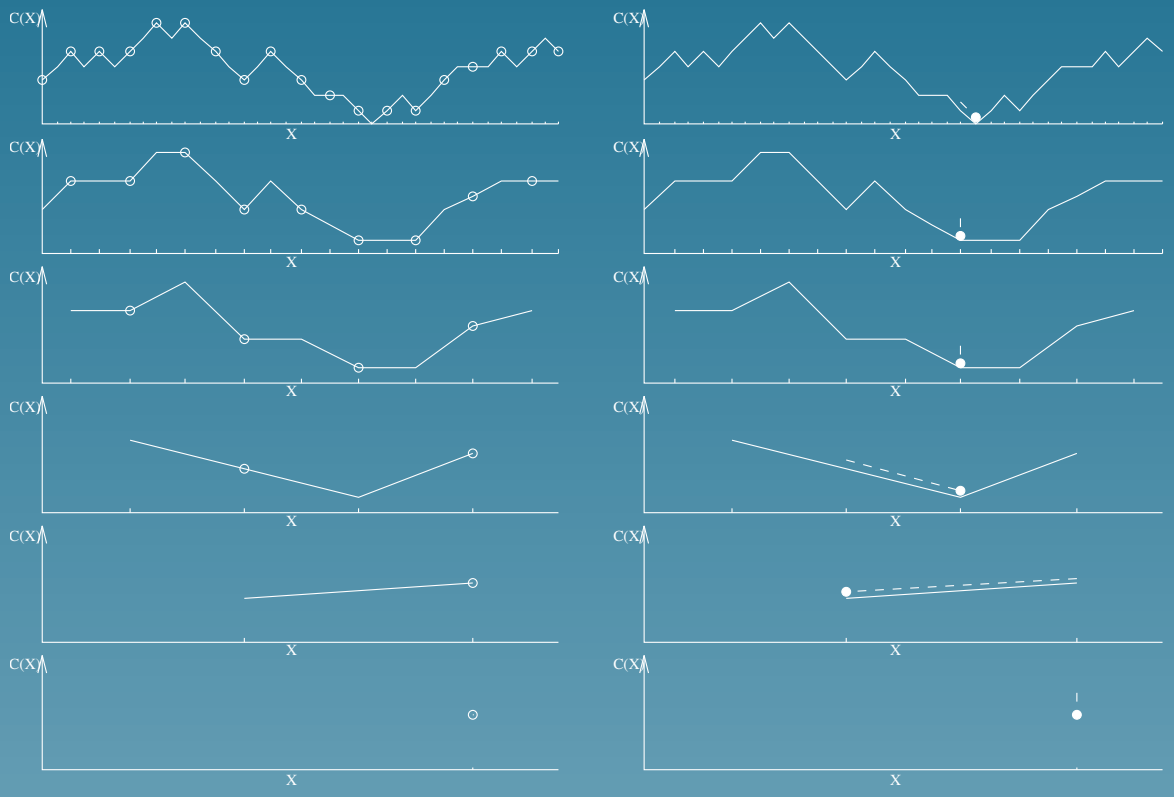
- in fact:
  coarsening $\Leftrightarrow$ sampling cost function

i.e. match v_1, v_2

$\Leftrightarrow$ *(GPP/GCP) restrict search space to solutions in which* v_1, v_2 *are in same set*

$\Leftrightarrow$ *(TSP) restrict search space to solutions where* v_1, v_2 *are neighbours*

Example cost function



sampling of solution space $\stackrel{?}{\Rightarrow}$ smoothing of cost function

the UNIVERSITY *of* GREENWICH

Generic multilevel refinement

Schematic

multilevel refinement(
`input` *problem instance* P_0 ,
`output` *solution* C_0)

l := 0
while (*coarsening*)
P_{l+1} = **coarsen**(P_l)
l := l+1
end
C_l = **initialise**(P_l)
while ($l > 0$)
l := l−1
C_l^0 = **extend**(C_{l+1}, P_l)
C_l = **refine**(C_l^0, P_l)
end

Generic multilevel refinement

4 components:

Generic multilevel refinement

4 components:

- coarsening algorithm

Generic multilevel refinement

4 components:

- coarsening algorithm

- initialisation algorithm
  typically trivial, e.g.
  GPP: distribute k vertices $\rightarrow$ k sets
  GCP: colour a complete graph
  TSP: construct a tour for 2 cities

Generic multilevel refinement

4 components:

- coarsening algorithm

- initialisation algorithm
  typically trivial, e.g.
  GPP: distribute k vertices $\rightarrow$ k sets
  GCP: colour a complete graph
  TSP: construct a tour for 2 cities
- extension algorithm
  trivial reversal of coarsening

Generic multilevel refinement

4 components:

- coarsening algorithm

- initialisation algorithm
  typically trivial, e.g.
  GPP: distribute k vertices $\rightarrow$ k sets
  GCP: colour a complete graph
  TSP: construct a tour for 2 cities
- extension algorithm
  trivial reversal of coarsening
- refinement algorithm *(may already exist)*
  local search (problem specific)
  metaheuristic: SA, TS, GA, AC, NN

Approximate run time

Assume:

problems size: $N, N/2, N/4, \ldots, 2$

refinement linear for G_l – i.e. $O(N_l)$

Let:

$T_r =$ time of refinement on graph size N

$T_c =$ total coarsening time, $T_c << T_r$

ML runtime $\approx T_r + T_r/2 + \ldots + 2T_r/N$

$\approx 2T_r$

$\Rightarrow$ complexity linear despite # graphs $= \log_2 N$

$\Rightarrow$ ML runtime $< 2T_r$ if refinement quadratic for G_l

Review of experimental results

Cost function

GPP/TSP:

$C(x)$ = sum of cut-edges/distances

(small change in solution changes cost)

$\Rightarrow$ $C(x)$ has *sensitive dependence on local conditions*

$\stackrel{?}{\Rightarrow}$ multilevel refinement allows smooth transition through the problem levels

GCP:

$C(x)$ = number of colour classes (sets)

(can change many elements of solution without changing cost)

$\stackrel{?}{\Rightarrow}$ effectiveness of multilevel refinement reduced

(solution on coarse levels may be far from that on fine levels)

Sparsity

- multilevel refinement better for sparse & low-density problems
  (fewer neighbours to choose a match with?)
- perhaps dependent on *matching affinities*
  (i.e. can pick matches with good probability that they will appear in the solution)
- sparsity is good indicator of affinities?

Iterated multilevel refinement

Solution based coarsening

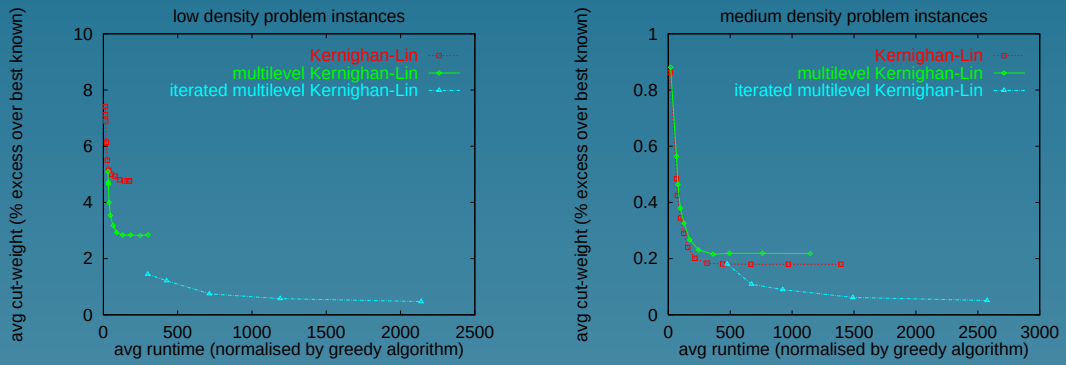
Given an existing solution with cost C

- restrict matching to solution
  GPP: match vertices in same set
  GCP: match vertices of same colour
  TSP: match vertices adjacent in tour
- guarantees initial solution to coarsest problem with cost C
  (although coarsening may terminate early)
- guarantees final solution to original problem with cost $C' \leq C$
  (assuming refinement algorithm provides same guarantees)

Iterated multilevel

- generate initial solution
  (e.g. with ML refinement)
- repeat:
  - use existing solution for coarsening
  - find new solution with ML refinement
- each iteration:
  – can guarantee to find solution with same or better cost
  – randomisation: each coarsening gives different hierarchy of problems
- similar to multigrid V-cycles

Iterated multilevel results for GPP



- compensates for medium-density GPP results
- not yet successful for TSP/GCP

Conclusions

Conclusions

- multilevel refinement can enhance local search/metaheuristics
  some results spectacular, but not a panacea
- (empirically) best for situations where:
  – cost function has sensitive dependence on local conditions
  – problem has inherent sparsity
- simple framework, flexible paradigm
- problem specific coarsening

$\stackrel{?}{\Rightarrow}$ “new” “metaheuristic” for combinatorial problems

Concluding remarks

Future directions:

- further testing on different problem classes
- sparsification
- advanced matching *(e.g. locally optimal, memory based)*
- vertical refinement *[Brandt, 1988]*

Technical report:

`www.gre.ac.uk/~c.walshaw/papers/ps/WalshawTR7301.ps.gz`

Papers:

`www.gre.ac.uk/~c.walshaw/papers`