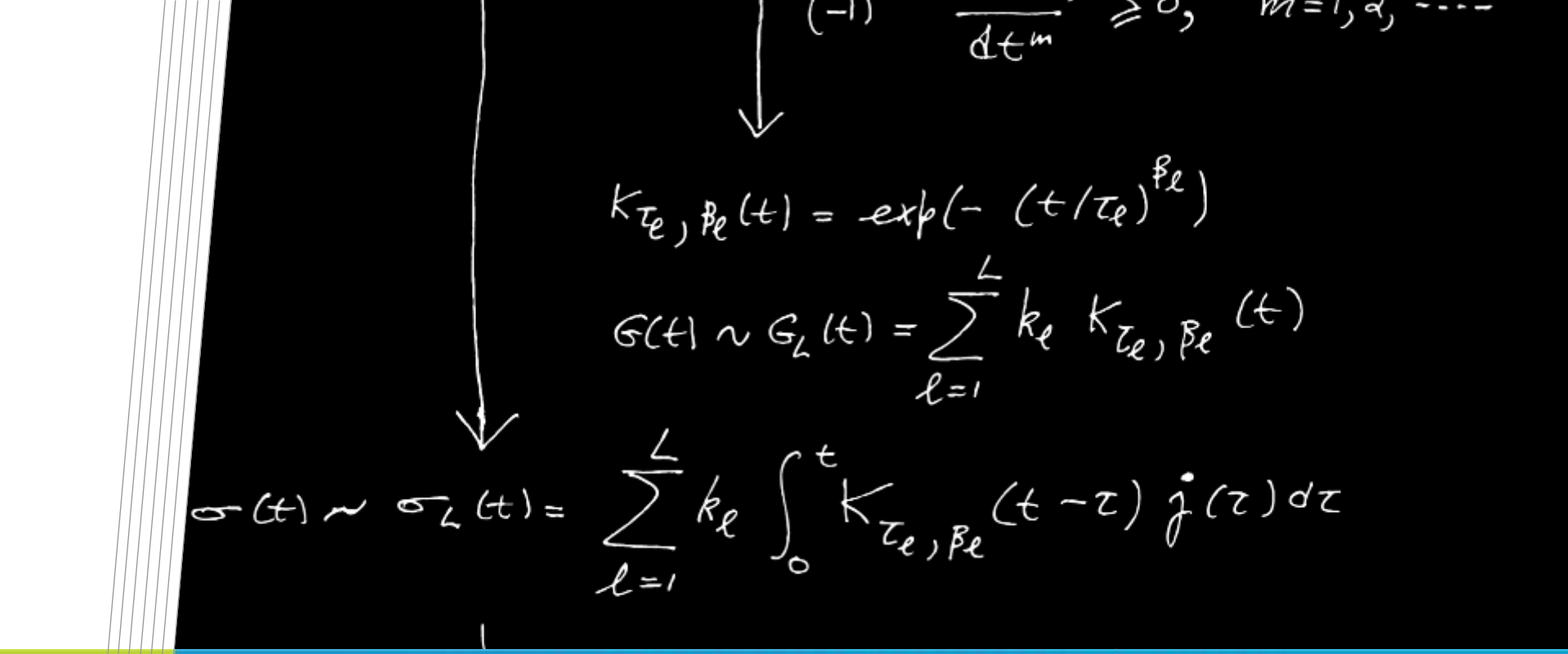


Shift Minimisation Personnel Task Scheduling Problem

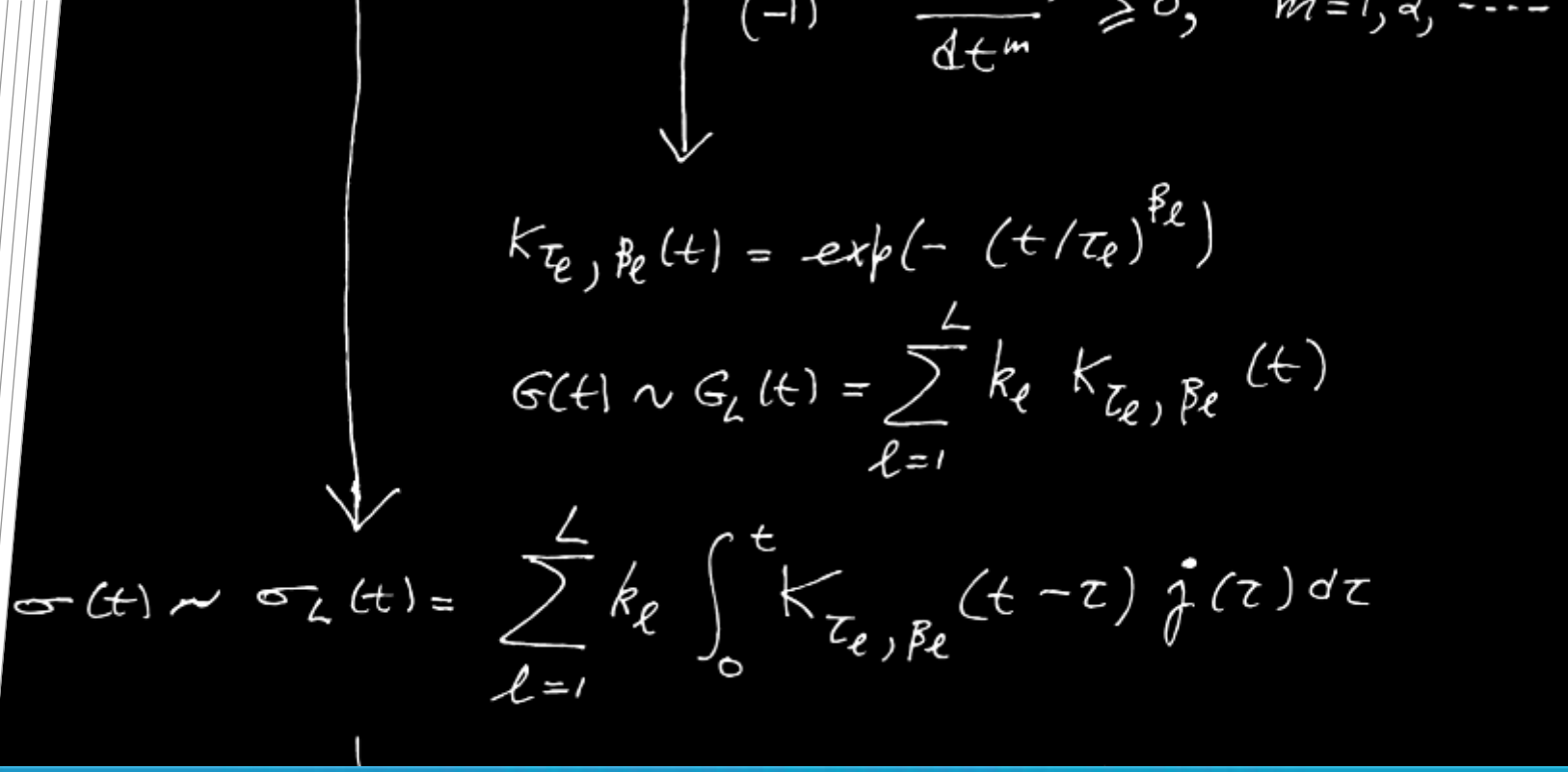
Andreas Ernst, CSIRO
Mohan Krishnamoorthy & Davaatseren Bataar, Monash University
Integer Programming Down Under, July 2011

www.csiro.au

CSIRO

Overview

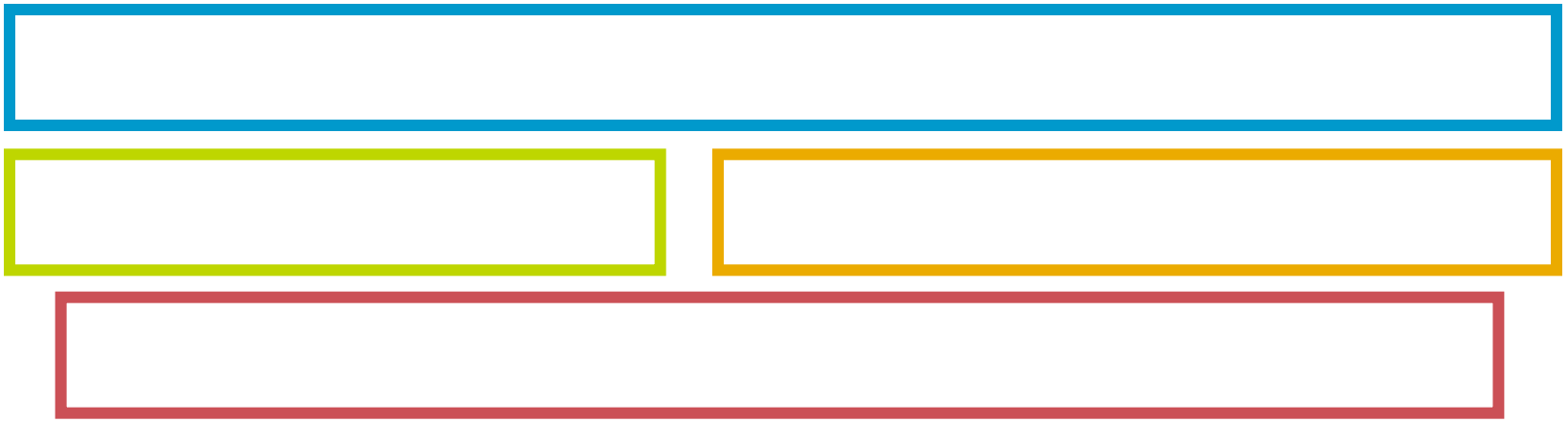
- Problem Definition
- Motivation
  - Planes, Trains & Automobiles
- ILP formulation and exact algorithms
- Lagrangian Heuristics
- Results
- Conclusions



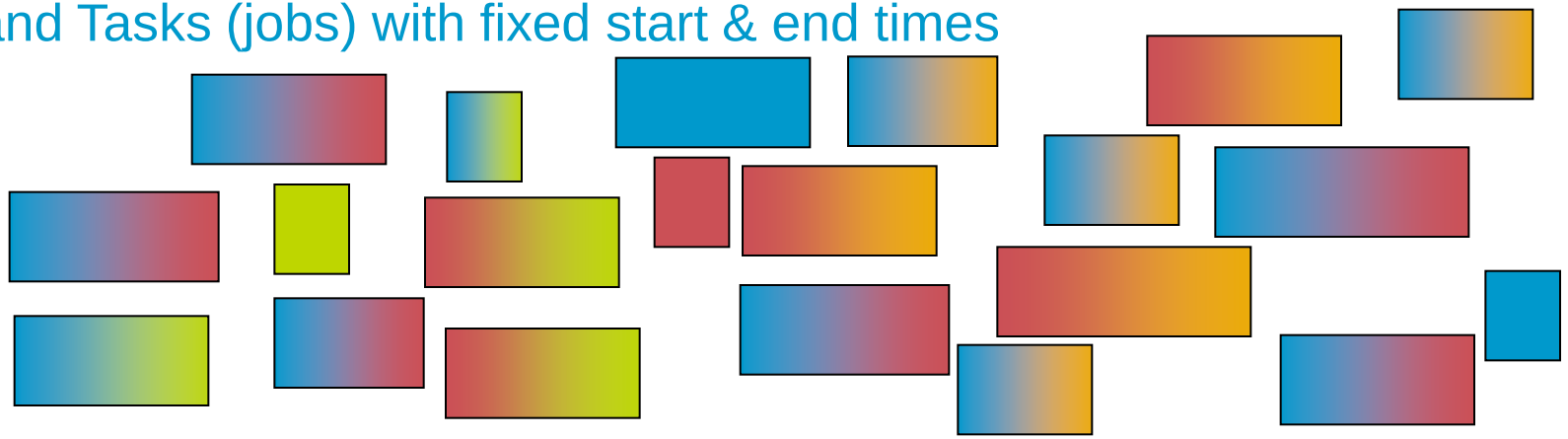
Background

Personnel Task Scheduling Problems

Given a set of shifts/workers:



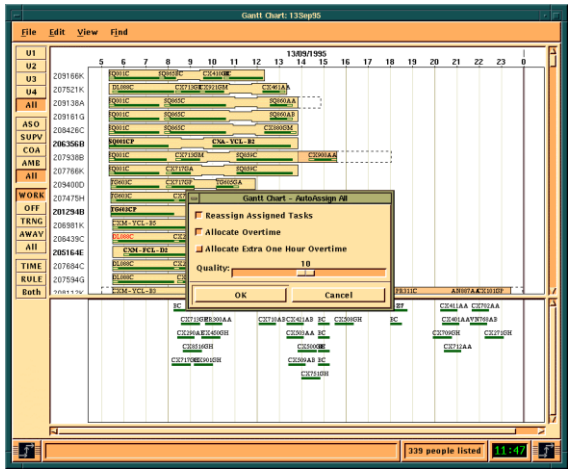
and Tasks (jobs) with fixed start & end times



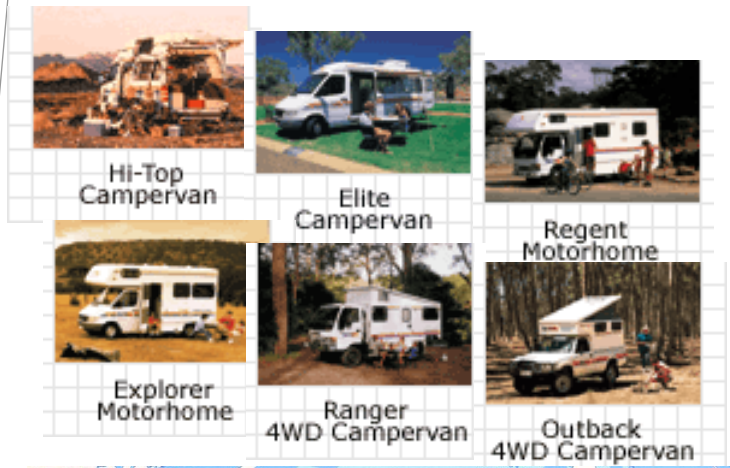
Assign all tasks to people that are qualified for it (max tasks assigned, minimum shifts used)

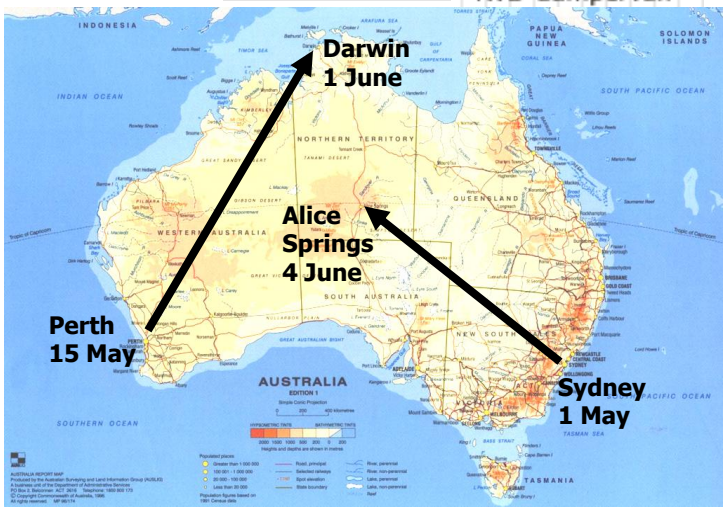
Scheduling of tasks for airport ground staff

- Ground staff at an airport
- Various tasks relating to servicing aircraft and passengers
- Task times driven by airline schedules
- Different qualifications for task by staff.
- Want to assign all tasks to staff each day, minimising casual staff/overtime required or tasks that are under-staffed.



Campervan Rentals Scheduling



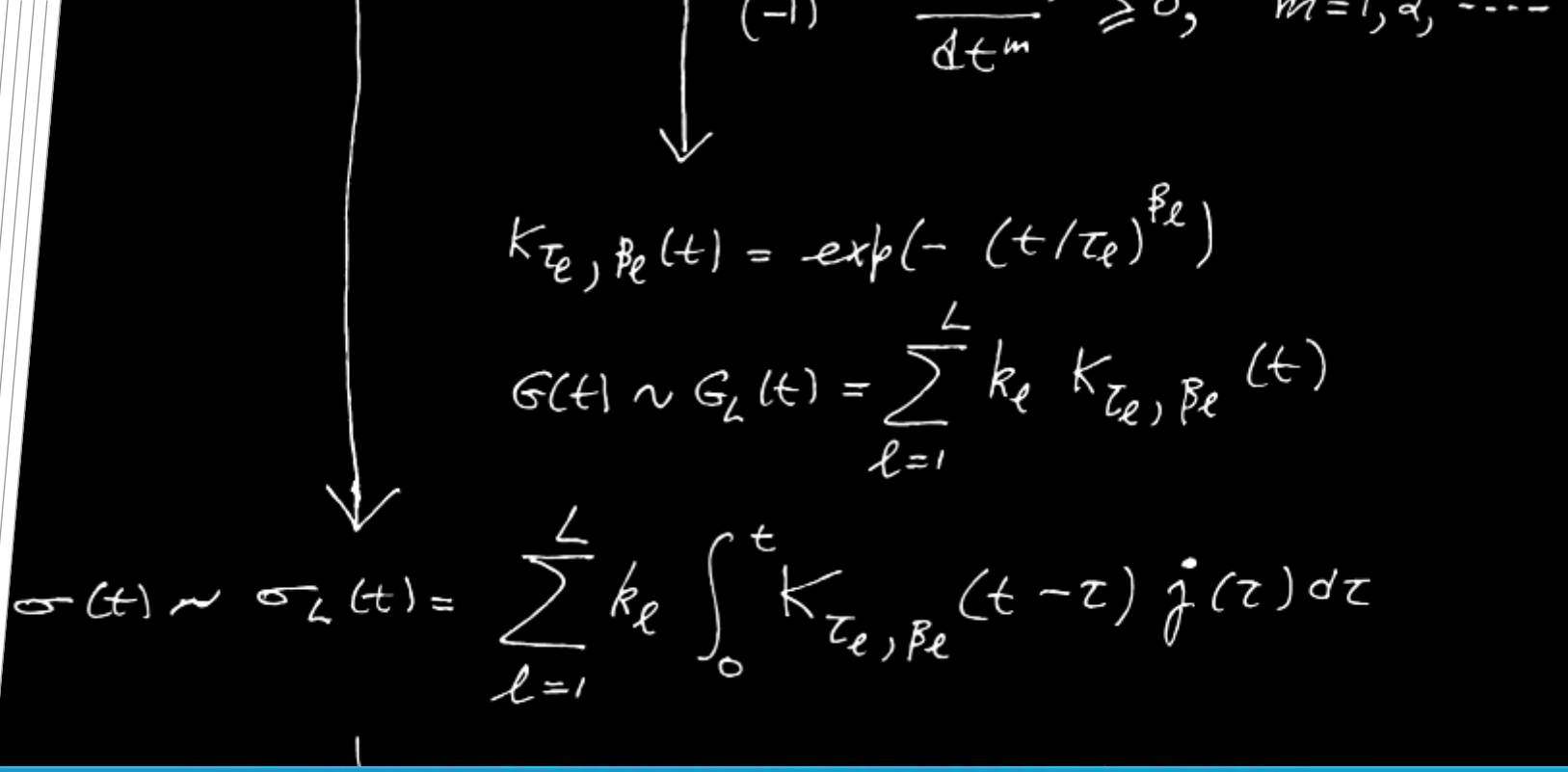


- Assign bookings (tasks with fixed start & end times) to vehicles (with qualification constraints for matching bookings based on vehicle type & initial location)
- Schedule relocations, upgrades, etc Australia wide over ~1 year
- System used to:
  - Create new schedule once or twice per day
  - Incrementally update schedule in response to changes
  - Check availability for any potential booking
  - Continuous re-optimisation
- Currently in used by 2 companies in Australia, NZ and USA.

Hunter Valley Rail Scheduling

- Create schedules for trains between coal-mines in the Hunter Valley & terminals in Newcastle
- Possible roundtrips with fixed start-end times created by train "paths"

  path = time slot on train line
- In addition to train "workers"
  - Loadpoints at the mines
  - Tracks where trains can wait
  - Dump stations
  - Terminal resources (eg stackers)
- Want to assign as many roundtrips as possible to the trains
  - subject to a range of additional side constraints
  - multiple types of soft constraints (preferences for a "good" schedule)



Exact Methods

Integer Linear Programming Formulation

$$\min \quad \sum_{w \in W} y_w + \sum_{j \in J} M\, u_j$$

$$\sum_{w \in P_j} x_{jw} + u_j = 1 \qquad \forall\ j \in J$$

Set of cliques of overlapping tasks for shift w

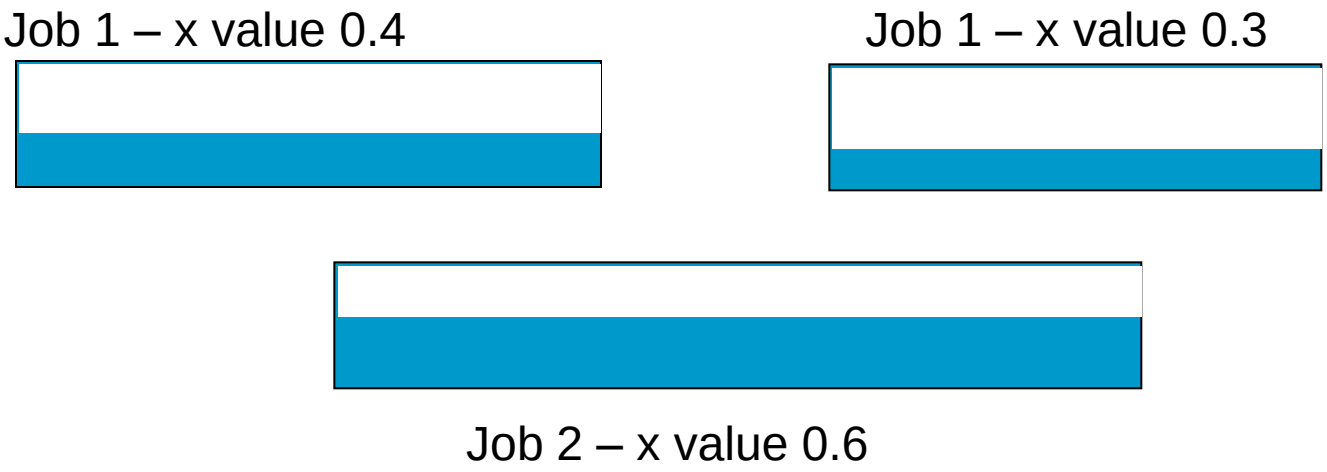
$$\sum_{j \in K} x_{jw} \leq y_w \qquad \forall\ w \in W,\ K \in C^w$$

Task j assigned shift w $\quad x_{jw} \in \{0,1\} \qquad \forall\ w \in W,\ j \in T_w$

Shift w used $\quad 0 \leq y_w \leq 1 \qquad \forall w \in W$

Task j unallocated $\quad 0 \leq u_j \leq 1 \qquad \forall j \in J$

Triplet based branching



Conflicting triplet:

3 jobs with one overlapping two otherwise compatible jobs

- Branch by selecting largest fractional variable in a conflicting triplet
- Any LP solution without conflicting triplets is integer

  (or can be turned into an equivalent integer solution)

Packing problem

- Find the maximum value set of tasks that can be completed by one (given) worker *w*

$$\max \quad \sum_{j \in T_w} \pi_j x_j$$

$$\text{s.t.} \quad \sum_{j \in K} x_j \leq 1 \qquad \forall\ K \in C^w$$

$$x_j \in \{0, 1\} \qquad \forall\ j \in T_w$$

- Can be solved in *O(n)* where *n* is number of tasks that *w* is qualified to carry out
- Simple shortest-path/dynamic programming algorithm

Preference Adjustment Heuristic

Start with preferences $\pi_{jw} = x_{jw}$ (fractional LP solution)

While not done:

Solve packing problems for all shifts w

If all jobs allocated, choose a shift to remove

Otherwise choose an unallocated task k

For each w, calculated the minimum change in π_{kw} that would result in j being allocated to w (from reduced costs)

Choose a $\underline{w}$ for which this change is smallest

Increase $\pi_{k\underline{w}}$ and decrease $\pi_{j\underline{w}}$ for all other tasks j

Column Generation

Price variables by solving packing problem using duals as preferences

1. Original Formulation: Add the x_{jw} that are used in the packing
2. Compact Formulation: Use binaries z_{wP} to select pattern P for shift w

$$\min \quad \sum_{w \in W} \sum_{P \in \mathcal{P}^w} z_{wP} + \sum_{j \in J} M u_j$$

$$\text{s.t.} \quad \sum_{w \in W} \sum_{\substack{P \in \mathcal{P}^w: \\ j \in P}} z_{wP} + u_j = \underline{1} \quad \forall\, j \in J$$

$$\sum_{k \in \mathcal{P}^w} z_{wk} \leq 1 \qquad \forall\, w \in W$$

$$z_{wP} \in \{0,1\} \qquad \forall\, w \in W,\ P \in \mathcal{P}^w$$

$$0 \leq u_j \leq 1 \qquad \forall j \in J$$

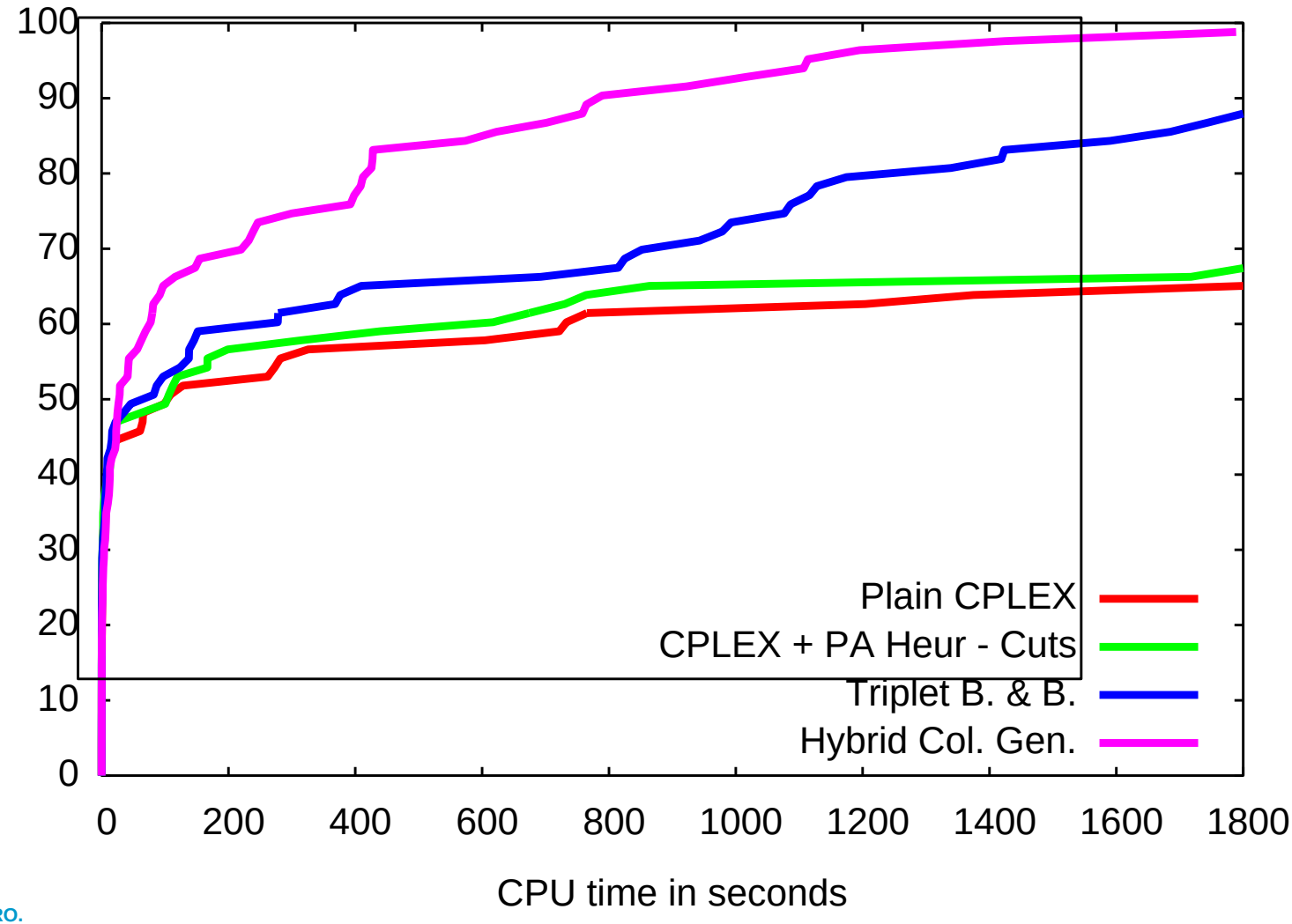
3. Hybrid: use Compact Formulation at root node and original formulation during B&B

   Since (2) is faster but (1) has generally less fractional solutions

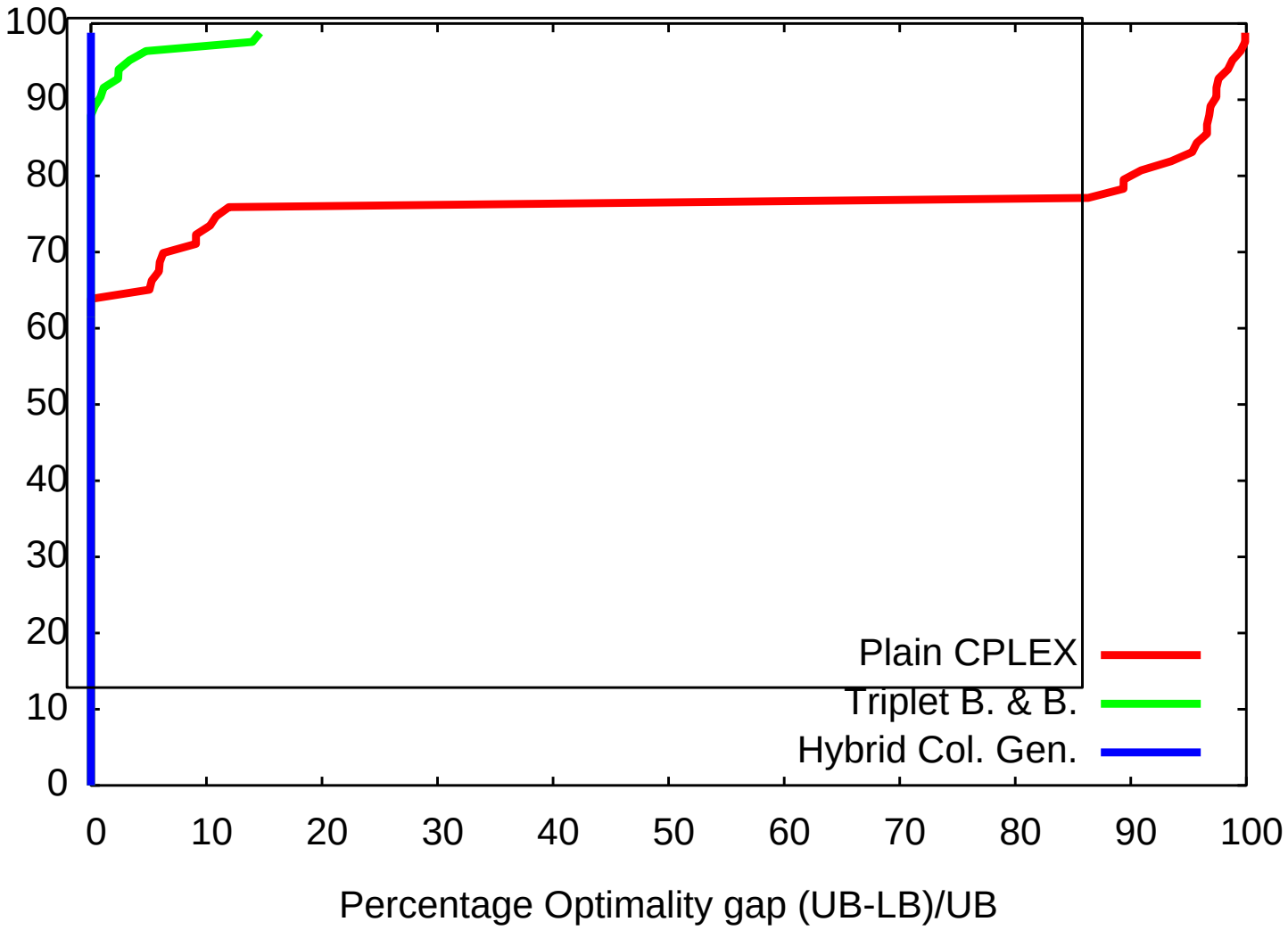
Computational Results

- Randomly generated instances
- About 80 instances with up to 380 shifts and 900 jobs used for exact method results
  - An additional 50 instances with up to 420 shifts and 2100 jobs used for the heuristic results
- Run on 4 * Quad-core Intel Xeon processors @ 2.93 GHz with 64Gb RAM

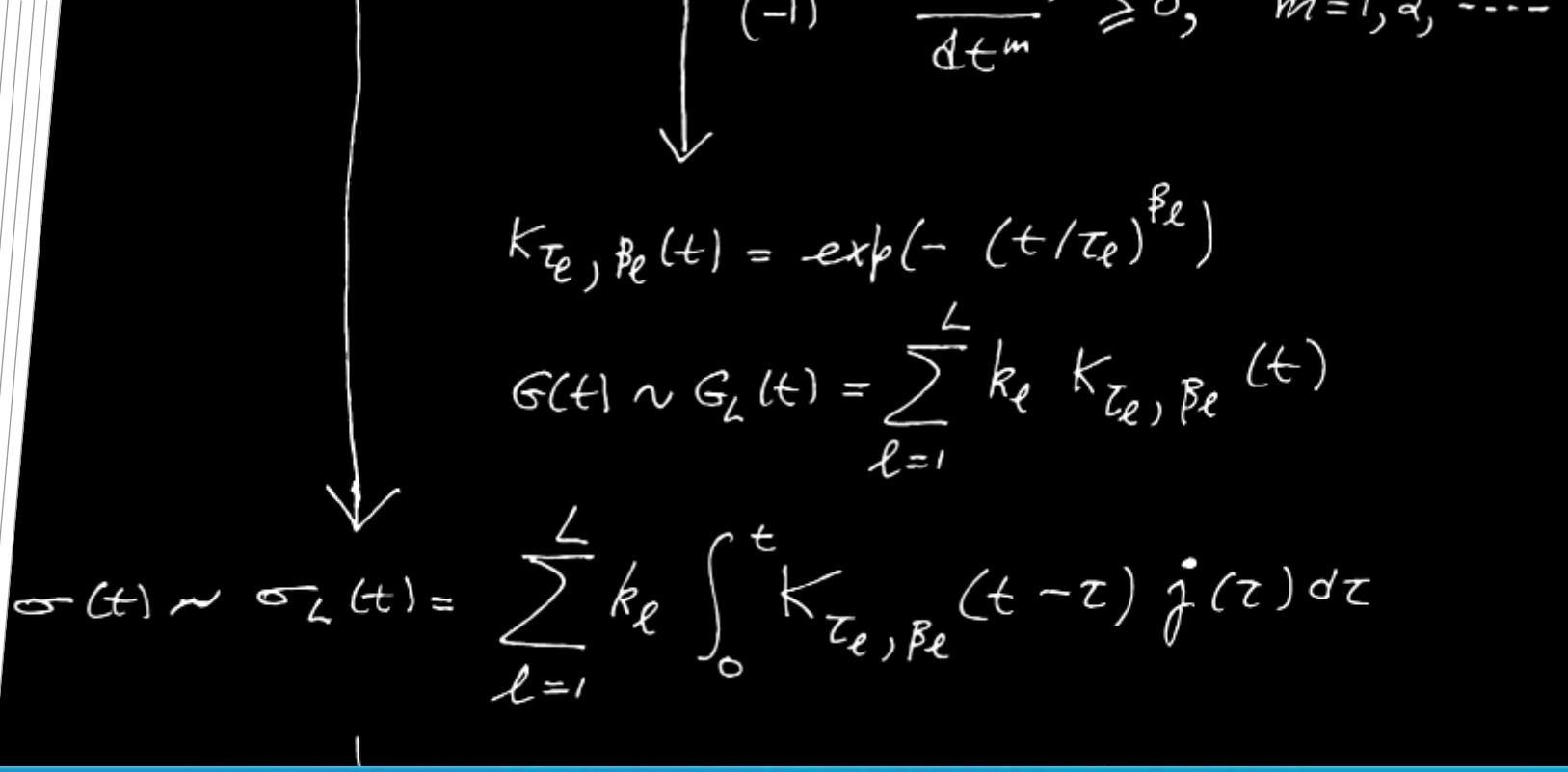
Problems solved within 1800 CPU seconds



Optimality Gaps



- All problems solved with column generation method



Lagrangian Heuristics

Lagrangian Approach

$$\min \quad \sum_{w \in W} y_w + \sum_{j \in J} M\, u_j$$

$$\sum_{w \in P_j} x_{jw} + u_j \quad \geq \quad 1$$

Relax with dual variables π_j

$$\sum_{j \in K} x_{jw} \quad \leq \quad y_w \qquad \forall\ w \in W,\ K \in C^w$$

$$x_{jw} \quad \in \quad \{0,1\} \quad \forall\ w \in W,\ j \in T_w$$

$$0 \ \leq \ y_w \ \leq \ 1 \qquad \forall w \in W$$

$$0 \ \leq \ u_j \ \leq \ 1 \qquad \forall j \in J$$

Lagrangian Method

- Basic approach:
  - Start with some dual value, solve relaxed problem, update dual vector, iterate
- Pros:
  - Fast
  - Reasonable lower bounds
- Cons:
  - Tends to zig-zag
  - No guarantee of primal feasible solutions
  - The solution for the relaxed problem may not be primal feasible for any value of the dual vector

Volume Algorithm

- **Choose an initial** $\pi_0 \in \mathbb{R}^m$ **compute** $x_0 \in \text{Argmin}\, L(x, \pi_0)$
  Let $v_0 := Ax_0 - b$ **Violation**
  $z_1 = x_0$ **estimate of primal solution**
  $w_1 := v_0$ **search direction**
  $\hat{\pi}_1 = \pi_0$ **stability center (current best dual)**

  ***t* = minor iter.**
  k = major iter.
  t = k = 1
- **Given** $\hat{\pi}_k$ **& step length** $s_t > 0$ **Let** $\pi_t = \hat{\pi}_k + s_t w_t$
- **Solve Lagrangean subproblem**
  $x_t \in \text{Argmin}\, L(x, \pi_t) \qquad v_t := Ax_t - b$
- **If** $\theta(\pi_t) > \theta(\hat{\pi}_k)$ **AND** $\langle w_t, v_t \rangle > 0$ **then**
  do a *serious step,* update stability center $\hat{\pi}_{k+1} = \pi_t$
  k++, $t_k := t$ **referred to as a *green iteration***
  else: *null step,* do nothing (aka *yellow* or *red* iteration)
- **Compute step size** s_{t+1} **and for some** $0 \le \alpha_t \le 1$
  $z_{t+1} := \alpha_t x_t + (1 - \alpha_t) z_t$
  $w_{t+1} := \alpha_t v_t + (1 - \alpha_t) w_t$
  $s_t = \mu \dfrac{\text{UB} - \theta(\pi_t)}{\|w_t\|^2}$
- **Set** *t*++**, go to step 1**

Volume Algorithm

- “Standard” Lagrangian relaxation approach
- Approximates primal solution
- Stabilises search direction by updating using a combination of previous search direction and current sub-gradient

$$z_{t+1} := \alpha_t x_t + (1 - \alpha_t) z_t$$

$$w_{t+1} := \alpha_t v_t + (1 - \alpha_t) w_t$$

$$s_t = \mu \frac{\mathrm{UB} - \theta(\pi_t)}{\|w_t\|^2}$$

Wedelin's Lagrangian Method

- Heuristic that tries to achieve primal feasibility by "splitting" Lagrangian multipliers π_j into π_{jw} (ie different for each shift w) and adding a perturbation
- Performs co-ordinate ascent to find good Lagrangian multiplier values. Given an unassigned task j

$$\pi_{\bar{j}\bar{w}} = \frac{r^- + r^+}{2} + \delta$$

$$\pi_{\bar{j}w} = \max\{\frac{r^- + r^+}{2} - \delta,\ 0\} \quad \forall w \in P_{\bar{j}}, w \neq \bar{w}$$

- Here
  - δ is a "small" perturbation
  - r^- & r^+ are the smallest & second smallest reduced cost for the task to any shift
  - Should make reduced cost negative for x_{jw} but positive for all others

Combined Algorithm

- Start with Volume Algorithms
  - Gives strong lower bound and starting point for Wedelin
- Use Wedelin's method to generate a good upper bound
  - Fairly similar to preference adjustment heuristic used for the exact methods
- This works quite well but …

… what are the other 15 CPU cores going to do?

Can solve Lagrangian sub-problems in parallel but unless dealing with significantly larger problem instances than considered here these are too small to give good parallelisation

→ Hybridisation with Particle Swarm Optimisation

Particle Swarm Optimisation

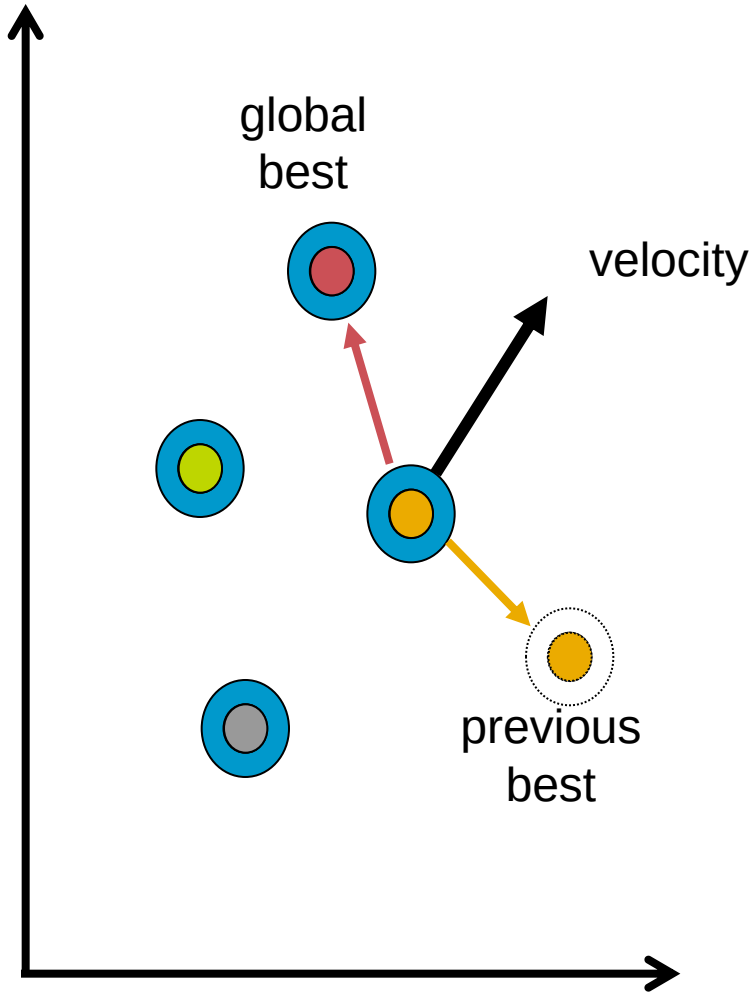
- Solution represented by
  - Continuous vector of variable values (***position***)
  - Search direction vector (***velocity***)
- Update velocity

  Velocity as weighted sum of
  1. previous velocity,
  2. direction to best solution found so far (*global optimisation*),
  3. direction to particle's best solution. (*local optimisation*)

  Update position based on velocity

$$\vec{v} \leftarrow \omega\vec{v} + \phi_p r_p(\vec{p} - \vec{x}) + \phi_g r_g(\vec{g} - \vec{x})$$

$$\vec{x} \leftarrow \vec{x} + \vec{v}$$



Hybrid Lagrangian Particle Swarm Opt.

- 2 Position vectors:
  - Lagrangian and perturbation (number of constraints/variables)
- 2 velocity vectors
- Follow basic pattern of PSO but
  - Replace update of Lagrangian velocity towards particle's best with update in direction of subgradient
  - Use Wedelin's method for determining local update direction of the perturbation velocity.
    - Add small constant for the "preferred" shift assignment of a job and subtract it for all others
- Note:
  - Added a repair heuristic to run occasionally
  - Can still obtain lower bounds (albeit weaker ones) even when there is a non-zero perturbation of the Lagrangian vector/matrix

Hybrid Lagrangian Particle Swarm Opt.

- 2 Position vectors:
  - Lagrangian and perturbation (number of constraints/variables)
- 2 velocity vectors
- Follow basic pattern of PSO but
  - Replace update of Lagrangian velocity towards particle's best with update in direction of subgradient
  - Use Wedelin's method for determining local update direction of the perturbation velocity.
    - Add small constant for the "preferred" shift assignment of a job and subtract it for all others
- One particle per CPU core, running in parallel with synchronisation after each iteration to update the global best
- Note:
  - Added a repair heuristic to run occasionally
  - Can still obtain lower bounds (albeit weaker ones) even when there is a non-zero perturbation of the Lagrangian vector/matrix

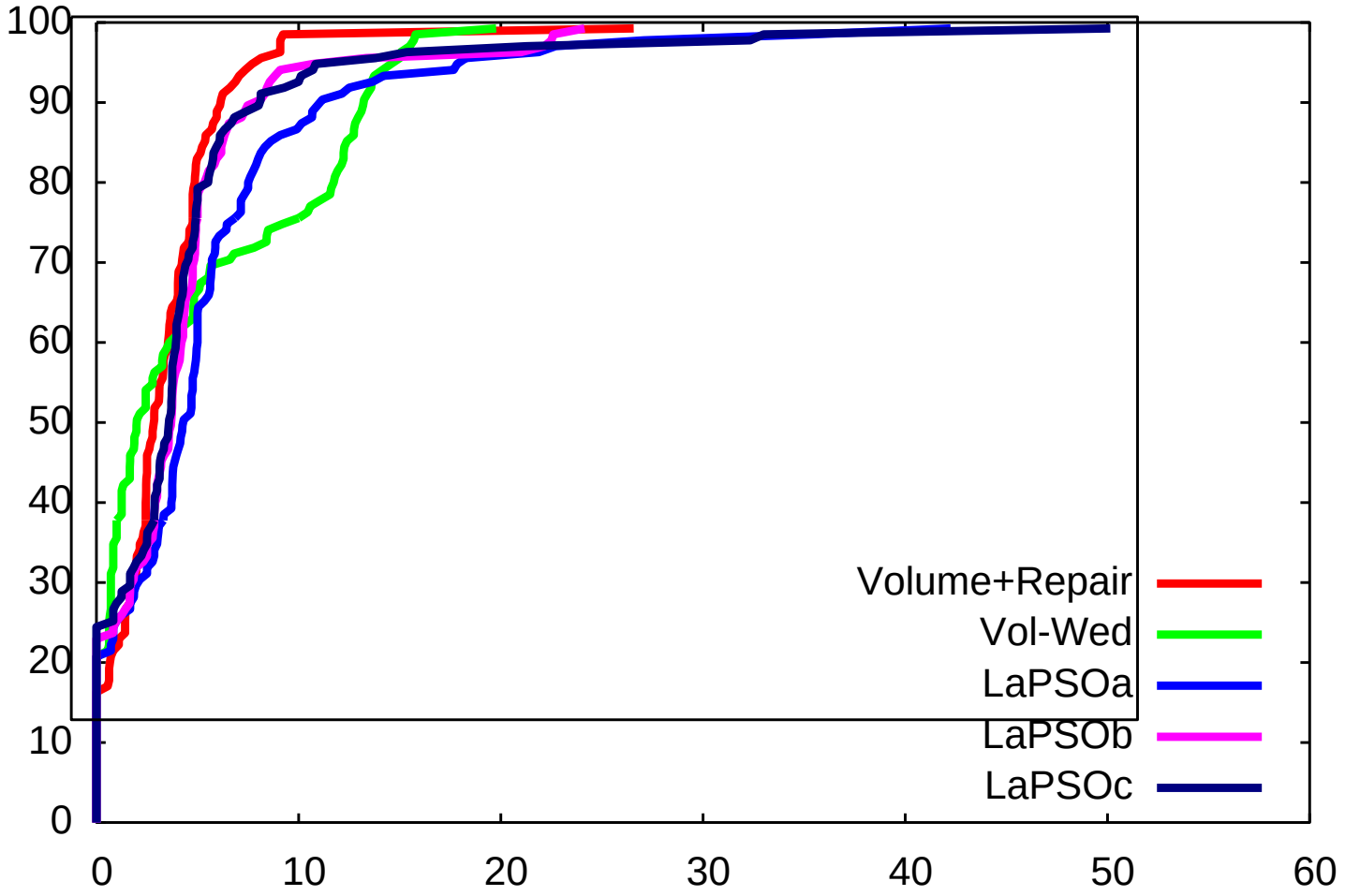
Methods compared

- Volume+Repair
  - Run volume algorithm only (no Wedelin)
  - Use a repair heuristic every few iterations
  - Lagrangian sub-problems solved in parallel
- Vol-Wed
  - First volume algorithm then Wedelin's method
  - No parallelisation
- LaPSOa – LaPSOc
  - Three runs of the Lagrangian Particle Swarm Opt. heuristic
  - First run has slightly different parameter settings to the other two
  - Shows sensitivity to randomisation & parameter settings
  - Uses 16 particles for 16 cores

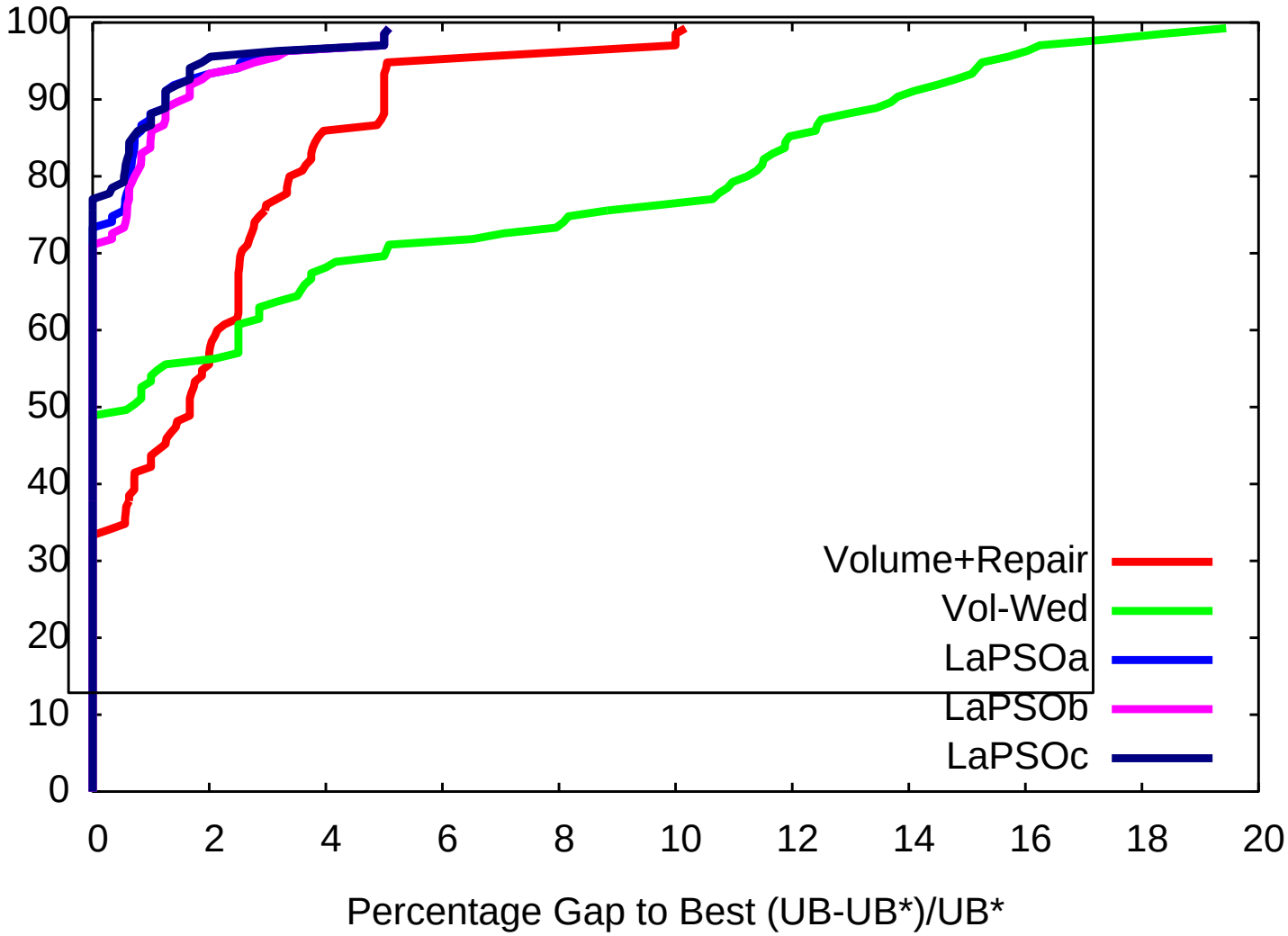
Optimality gap of heuristics

Compare: worst gap for B&B ~15%, Volume+Repair ~ 10%, Vol-Wed ~ 20%

but exact methods perform very poorly on larger problems even with 5000 CPU sec

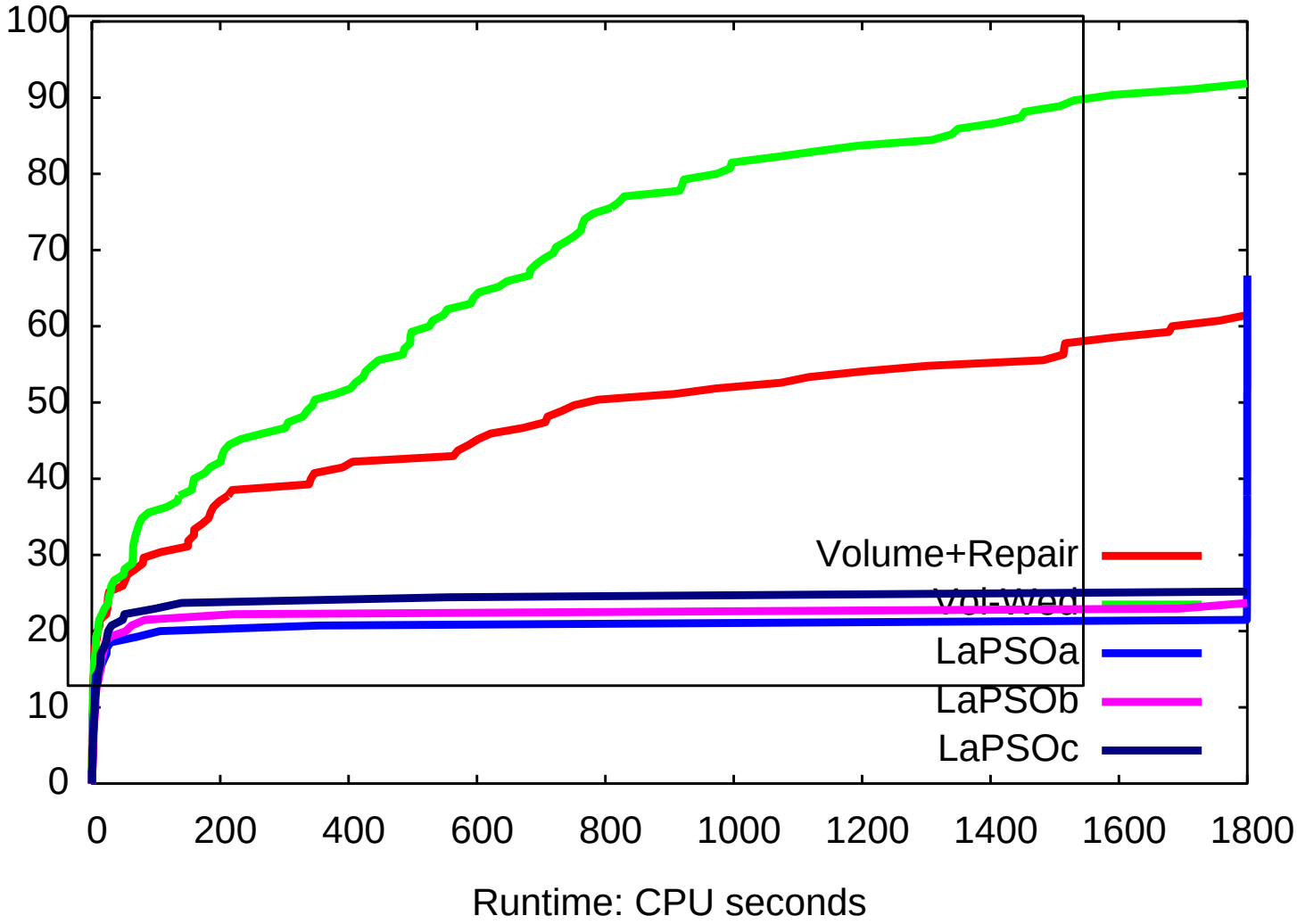


Gap to best solution found by any method



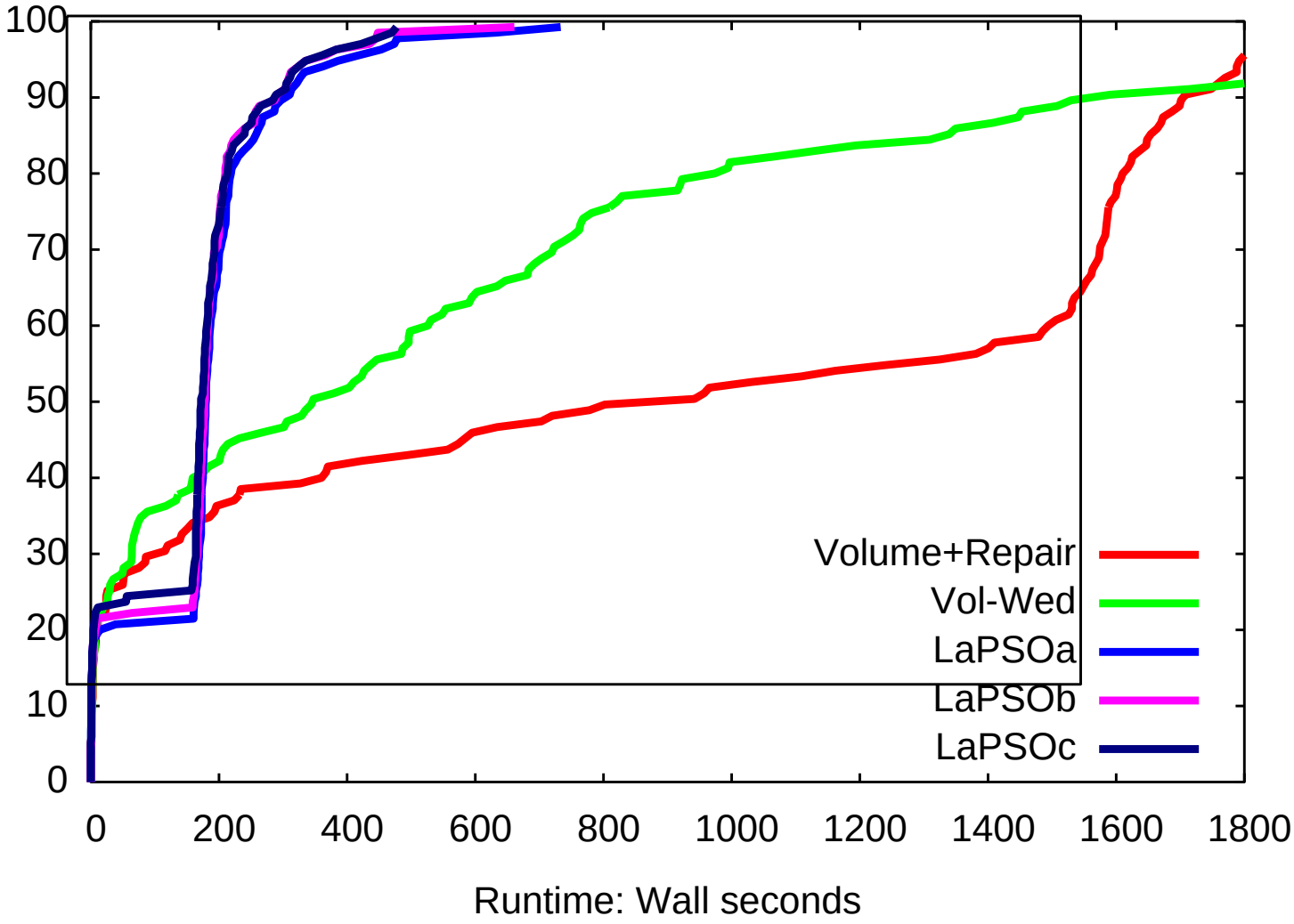
→ Main advantage of Volume-Wedelin method is in lower bounds

CPU time required

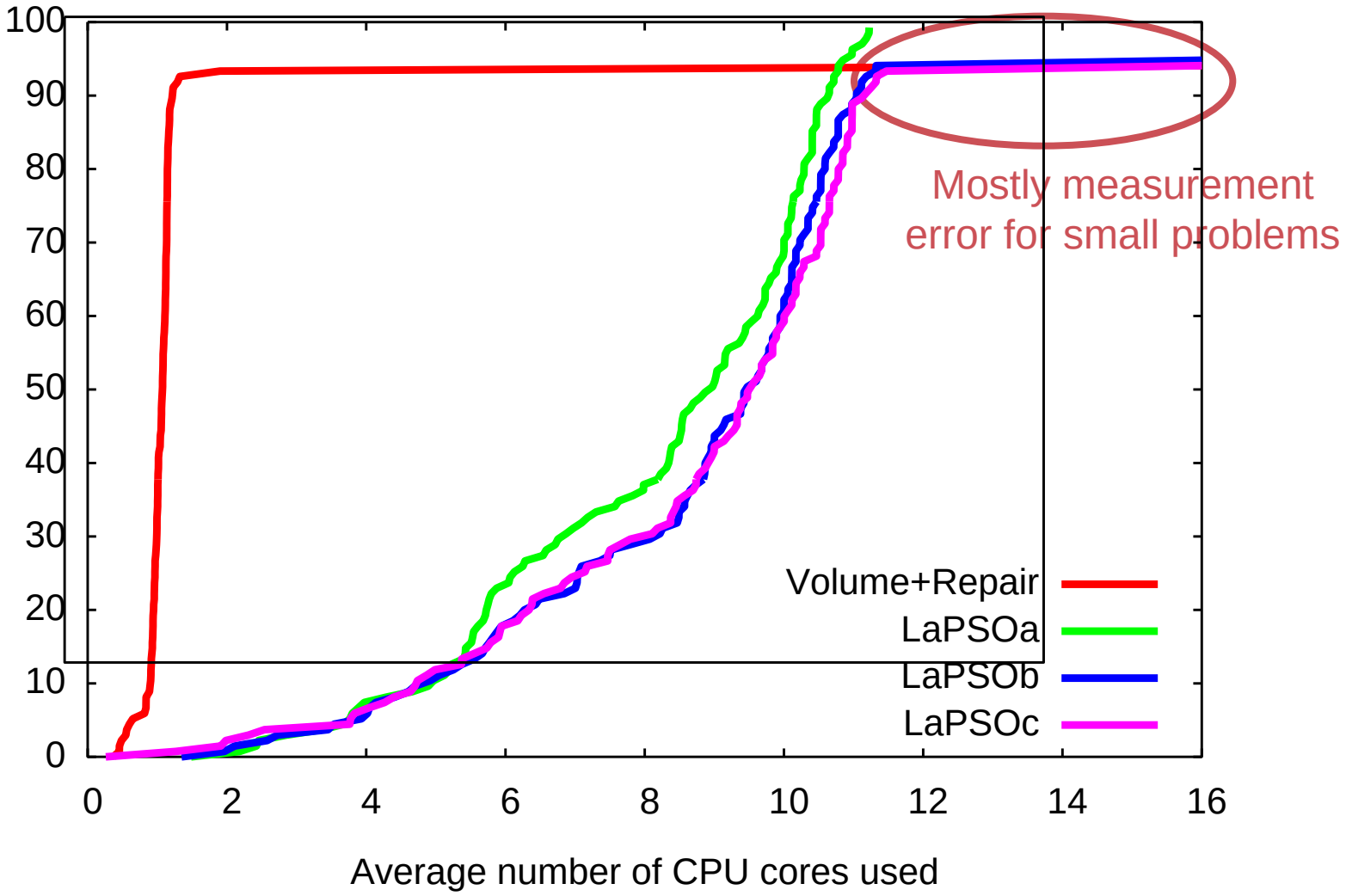


→ Particle Swarm Method doesn't know when to give up!

Elapsed (wall) time required



Average number of CPU cores used



Conclusions

- Some gains from column generation and custom branching scheme, root-node heuristic…
- but overall simplex based methods only effective if provable optimality is important and problems are small
- Lagrangian methods perform well
  - Need to exploit strengths of both Volume method and Wedelin's heuristic approach to get good solution
- Particle swarm optimisation provides good framework for parallelisation

Questions:

- How to increase CPU utilisation on a parallel machine
- What should I do with the 32 cores of our newest machine given that I can't even use 16 effectively?

Andreas Ernst
Operations Research Group
CSIRO Mathematics, Informatics and Statistics

Email: Andreas.Ernst@csiro.au

www.csiro.au

Thank you

Contact Us
Phone: 1300 363 400 or +61 3 9545 2176
Email: enquiries@csiro.au Web: www.csiro.au

CSIRO