Optimization of two-phase methods using simple feedback mechanisms

W. Kool

VU University
Amsterdam

ORTEC
Zoetermeer

Agenda

Agenda

Introduction

Two-phase methods

- First phase solves part of the problem
- Second phase solves other part of the problem, given solution from first phase
- Overall solution is at most conditionally optimal given decisions in first phase

Introduction

Two-phase methods

- First phase solves part of the problem
- Second phase solves other part of the problem, given solution from first phase
- Overall solution is at most conditionally optimal given decisions in first phase

Examples

- First create clusters/regions/areas, then create routes
- Create timetable/routes, then assign drivers/crews
- More than two phases can be regarded as nested two-phase methods

Introduction

Idea

- There must be some 'cost metric' c_1 for first phase which, if optimized, results in intermediate solution that leads to global optimum
- Start with a priori belief $\tilde{c}_1$ of c_1 and update $\tilde{c}_1$ based on observations
- More precisely, let $\tilde{c}_1$ exactly represent last observed solution
- Stop if $\tilde{c}_1$ is no longer updated (so fixed point iteration)

Introduction

Instance x, intermediate solution $y = F_1(x, c_1)$, solution $z = F_2(x, y)$

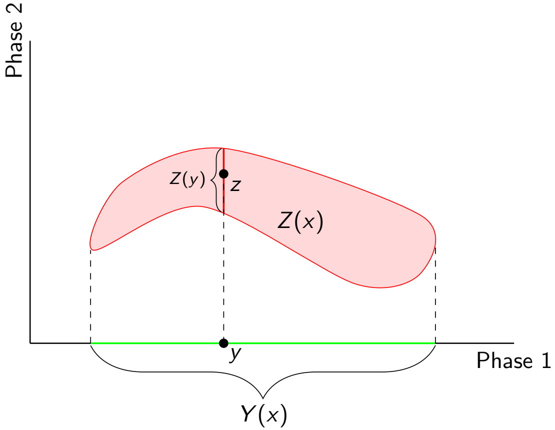


Figure: Visualization of domains

Agenda

Framework

Notation

- Instance x, intermediate solution $y \in Y(x)$, solution $z \in Z(y) \subseteq Z(x)$
- Cost metric c_1 and $c_2 = c$ for phase 1 and 2
- Optimization methods $F_1(x, c_1)$ and $F_2(x, y, c_2)$
- $c_2 = c$ fixed, so $F_2(x, y) = F_2(x, y, c_2)$

Framework

Notation

- Instance x, intermediate solution $y \in Y(x)$, solution $z \in Z(y) \subseteq Z(x)$
- Cost metric c_1 and $c_2 = c$ for phase 1 and 2
- Optimization methods $F_1(x, c_1)$ and $F_2(x, y, c_2)$
- $c_2 = c$ fixed, so $F_2(x, y) = F_2(x, y, c_2)$

Assumptions

- $F_1(x, c_1) = \arg\min_{y \in Y(x)} c_1(x, y)$
- $F_2(x, y) = \arg\min_{z \in Z(y)} c(x, z)$

Framework

Optimality

- Let z^* be the global optimum, and y^* the corresponding intermediate solution
- To find global optimum, c_1 should be such that $y^* = F_1(x, c_1)$
- Multiple options, for example $c_1(x, y) = \mathbb{1}_{\{y \neq y^*\}}$ and $c_1(x, y) = c(x, F_2(x, y)) = \min_{z \in Z(y)} c(x, z)$

Framework

Optimality

- Let z^* be the global optimum, and y^* the corresponding intermediate solution
- To find global optimum, c_1 should be such that $y^* = F_1(x, c_1)$
- Multiple options, for example $c_1(x, y) = \mathbb{1}_{\{y \neq y^*\}}$ and $c_1(x, y) = c(x, F_2(x, y)) = \min_{z \in Z(y)} c(x, z)$

Idea

- Problem: c_1 is not well behaved and hard to evaluate
- Therefore, replace c_1 by 'belief' $\tilde{c}_1$
- Update $\tilde{c}_1$ based on observation and repeat
- Convergence if $\tilde{c}_1$ no longer changes

Framework

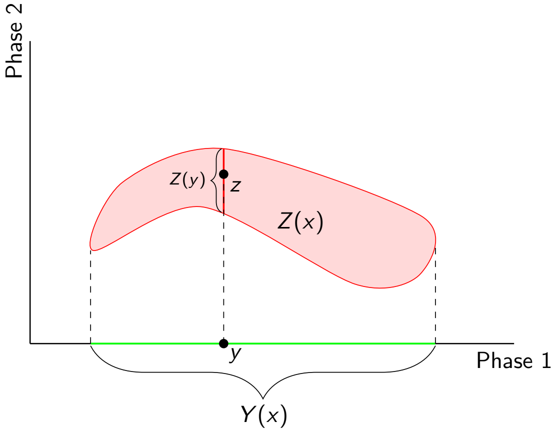


Figure: Visualization of domains

Framework

The feedback mechanism

- Update $\tilde{c}_1$ such that $\tilde{c}_1(x, \hat{y}) = c(x, \hat{z})$, where $\hat{y}, \hat{z}$ is the (intermediate) solution from the last iteration

Framework

The feedback mechanism

- Update $\tilde{c}_1$ such that $\tilde{c}_1(x, \hat{y}) = c(x, \hat{z})$, where $\hat{y}, \hat{z}$ is the (intermediate) solution from the last iteration

Considerations

- May invalidate previous equalities, but convergence if same solution is found
- Optionally require that the equality is *also* satisfied for best solution so far (the incumbent)
- Important: the *a priori* belief should be *optimistic* to avoid immediate convergence in local optimum

Framework

Algorithm 1 Simple feedback mechanism.

```
procedure SimpleFeedback(x, F_1, F_2, c̃_1, c)
    while c̃_1 has not converged do
        ŷ ← F_1(x, c̃_1);
        ẑ ← F_2(x, ŷ);
        c̃_1 ← Update(c̃_1, ŷ, ẑ, c);      ▷ Update c̃_1(x, ŷ) := c(x, ẑ)
    end while
end procedure
```

Agenda

The pallet matching problem

The problem

- Given a set of request and a set of pallets
- Match the requests in pairs (stacks) of two
- To each request, match one fulfilling pallet
- Costs depend on resulting pairs of pallets

The pallet matching problem

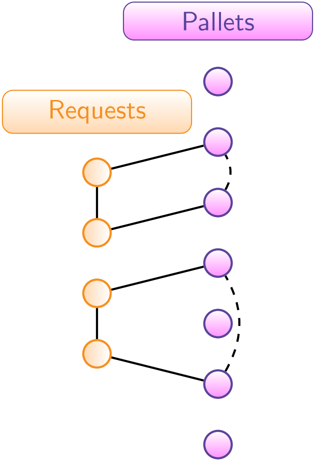


Figure: Visualization of the pallet matching problem

The pallet matching problem

Two phase decomposition

- Phase 1: (non-bipartite) matching of requests
- Phase 2: bipartite matching of requests to pallets
  - Note: *not* a pure bipartite matching problem because of cost function!

The pallet matching problem

Ingredients

- y: matching of requests

The pallet matching problem

Ingredients

- y: matching of requests
- $c_1(x, y)$: represented by cost matrix D

The pallet matching problem

Ingredients

- y: matching of requests
- $c_1(x, y)$: represented by cost matrix D
- $F_1(x, c_1)$: minimum cost perfect matching using cost matrix D

The pallet matching problem

Ingredients

- y: matching of requests
- $c_1(x, y)$: represented by cost matrix D
- $F_1(x, c_1)$: minimum cost perfect matching using cost matrix D
- $F_2(x, y)$: matching of pallets to request using MIP

The pallet matching problem

Ingredients

- y: matching of requests
- $c_1(x, y)$: represented by cost matrix D
- $F_1(x, c_1)$: minimum cost perfect matching using cost matrix D
- $F_2(x, y)$: matching of pallets to request using MIP
- $\tilde{c}_1$: entry $\tilde{d}_{k\ell}$ of $\tilde{D}$ represents lower bound of costs (costs if ideal pallets are matched)

The pallet matching problem

Ingredients

- y: matching of requests
- $c_1(x, y)$: represented by cost matrix D
- $F_1(x, c_1)$: minimum cost perfect matching using cost matrix D
- $F_2(x, y)$: matching of pallets to request using MIP
- $\tilde{c}_1$: entry $\tilde{d}_{k\ell}$ of $\tilde{D}$ represents lower bound of costs (costs if ideal pallets are matched)
- Updating mechanism: set $\tilde{d}_{k\ell}$ to actual costs of pallets matched to requests k and ℓ

The pallet matching problem

	Requests				Pallets						
Requests	0	1	0	0	0	0	1	0	1	0	0
	1	0	1	1	1	0	1	1	0	0	1
	0	1	0	1	1	1	0	1	1	0	1
	0	1	1	0	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

	Requests				Pallets						
Requests	0	1	0	0	0	0	1	0	1	0	0
	1	0	1	1	1	0	1	1	0	0	1
	0	1	0	1	1	1	0	1	1	0	1
	0	1	1	0	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

	Requests				Pallets						
Requests	0	1	0	0	0	0	1	0	1	0	0
	1	0	1	1	1	0	1	1	0	0	1
	0	1	0	1	1	1	0	1	1	0	1
	0	1	1	0	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

	Requests				Pallets						
Requests	0	1	0	0	0	0	1	0	1	0	0
	1	0	1	1	1	0	1	1	0	0	1
	0	1	0	1	1	1	0	1	1	0	1
	0	1	1	0	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

	Requests				Pallets						
Requests	0	1	0	0	0	0	1	0	1	0	0
	1	0	1	1	1	0	1	1	0	0	1
	0	1	0	1	1	1	0	1	1	0	1
	0	1	1	0	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

	Requests				Pallets						
Requests	0	1	0	0	0	0	1	0	1	0	0
	1	0	1	1	1	0	1	1	0	0	1
	0	1	0	1	1	1	0	1	1	0	1
	0	1	1	0	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

		Requests				Pallets						
Requests		0	1	0	0	0	0	1	0	1	0	0
		1	0	1	1	1	0	1	1	0	0	1
		0	1	0	1	1	1	0	1	1	0	1
		0	1	1	0	1	0	1	1	0	0	0
Pallets		0	1	1	1	∞	52	54	6	57	27	56
		0	0	1	0	52	∞	91	63	57	5	47
		1	1	0	1	54	91	∞	27	39	84	42
		0	1	1	1	6	63	27	∞	69	10	21
		1	0	1	0	57	57	39	69	∞	73	24
		0	0	0	0	27	5	84	10	73	∞	31
		0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

	Requests				Pallets						
Requests	∞	24	∞	∞	0	0	1	0	1	0	0
	24	∞	6	6	1	0	1	1	0	0	1
	∞	6	∞	6	1	1	0	1	1	0	1
	∞	6	6	∞	1	0	1	1	0	0	0
Pallets	0	1	1	1	∞	52	54	6	57	27	56
	0	0	1	0	52	∞	91	63	57	5	47
	1	1	0	1	54	91	∞	27	39	84	42
	0	1	1	1	6	63	27	∞	69	10	21
	1	0	1	0	57	57	39	69	∞	73	24
	0	0	0	0	27	5	84	10	73	∞	31
	0	1	1	0	56	47	42	21	24	31	∞

Figure: Calculating the lower bound matrix $\tilde{D}$ for $\tilde{c}_1$

The pallet matching problem

Algorithm 2 Simple feedback mechanism.

procedure SimpleFeedbackPalletMatching(x)
  $D \leftarrow$ calculateLowerBoundMatrix(x)
  while true **do**
    $\hat{y} \leftarrow$ matchRequests(x, D);
    $\hat{z} \leftarrow$ matchPallets($x, \hat{y}$);
    if $c_1(\hat{y}, D) == c(\hat{z})$ **then return**
    end if
    $D \leftarrow$ Update(D, $\hat{y}$, $\hat{z}$, c); ▷ Update $\tilde{c}_1(x, \hat{y}) := c(x, \hat{z})$
  end while
end procedure

The pallet matching problem

Average gap with optimum: 2.88% (4.55% if heuristic used for second phase)

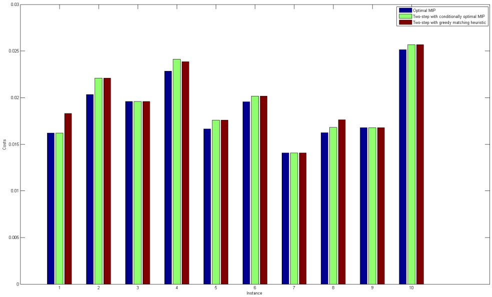


Figure: Results for 10 randomly generated instances

The pallet matching problem

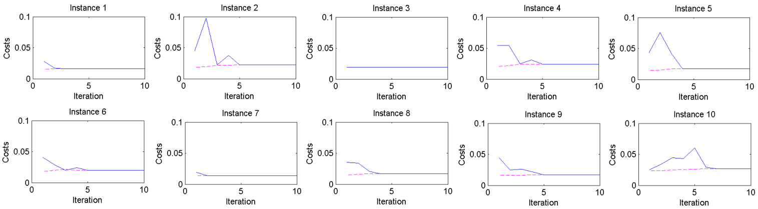


Figure: Solution quality vs. iterations

Agenda

The capacitated vehicle routing problem

The problem

- Given a depot, a set of customers, a demand for each customer, a vehicle capacity and a distance matrix
- Create routes with customers such that the total demand in each route does not exceed the capacity
- The total driving time or distance should be minimized

The capacitated vehicle routing problem

Figure: Visualization of the (capacitated) vehicle routing problem

The capacitated vehicle routing problem

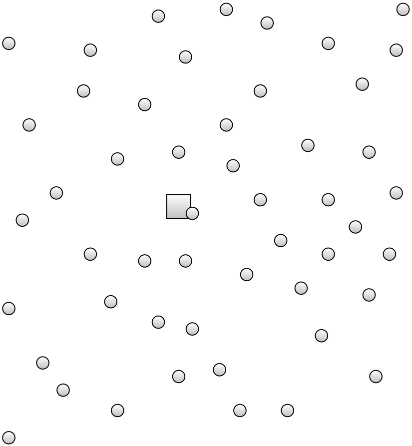


Figure: Visualization of the (capacitated) vehicle routing problem

The capacitated vehicle routing problem

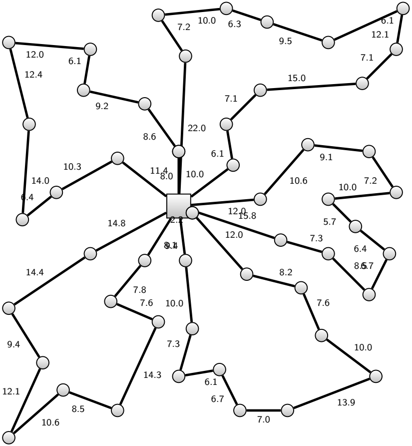


Figure: Visualization of the (capacitated) vehicle routing problem

The capacitated vehicle routing problem

Two phase decomposition

- Cluster-first-route-second method according to Bramel and Simchi-Levi (1995), inspired by Fisher and Jaikumar (1981)
- Phase 1: create clusters using capacitated contractor location problem (CCLP)
- Phase 2: create routes by solving travelling salesman problem (TSP) for each cluster

The capacitated vehicle routing problem

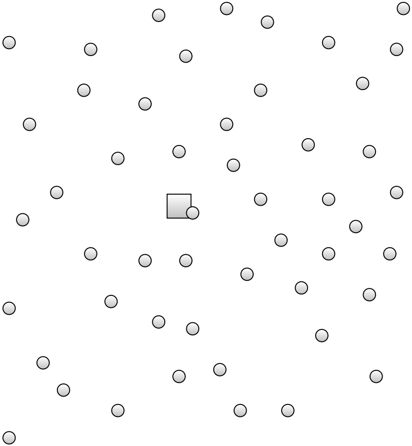


Figure: Visualization of the CCLP

The capacitated vehicle routing problem

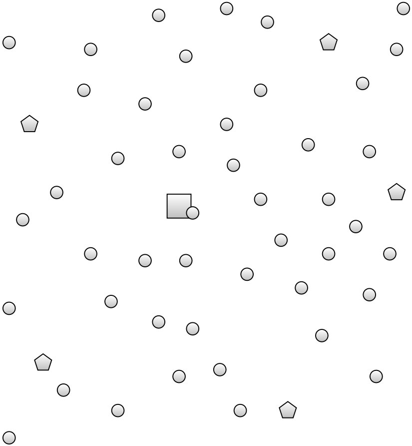


Figure: Visualization of the CCLP

The capacitated vehicle routing problem

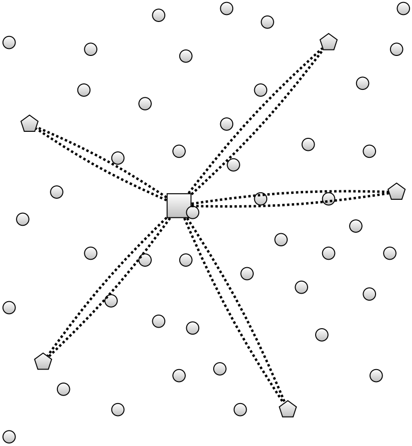


Figure: Visualization of the CCLP

The capacitated vehicle routing problem

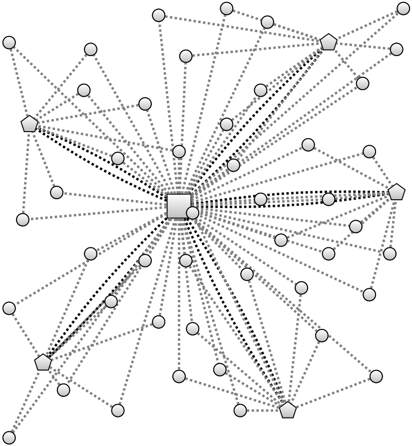


Figure: Visualization of the CCLP

The capacitated vehicle routing problem

Variables

- $y_{jj} = 1$ if j is a seed
- $y_{ij} = 1$ if i is connected to seed j

MIP formulation

$$
\begin{aligned}
\min \quad & c_1(x, y) = \sum_{i \in V \setminus \{0\}} \sum_{j \in V \setminus \{0\}} y_{ij} d_{ij} & & (1) \\
\text{s.t.} \quad & \sum_{j \in V \setminus \{0\}} y_{ij} = 1 & i \in V \setminus \{0\} & \quad (2) \\
& \sum_{i \in V \setminus \{0\}} a_i y_{ij} \leq B & j \in V \setminus \{0\} & \quad (3) \\
& y_{ij} \leq y_{jj} & i, j \in V \setminus \{0\}, i \neq j & \quad (4) \\
& y_{ij} \in \{0, 1\} & i, j \in V \setminus \{0\} & \quad (5)
\end{aligned}
$$

The capacitated vehicle routing problem

Ingredients

- y: clustering of customers
  - $y_{ij} = 1$ if i is connected to j, $y_{jj} = 1$ if j is a seed

The capacitated vehicle routing problem

Ingredients

- y: clustering of customers
  - $y_{ij} = 1$ if i is connected to j, $y_{jj} = 1$ if j is a seed
- $c_1(x, y)$: represented by cost matrix D
  - d_{ij} are the connection costs of location i to seed j
  - d_{jj} are the setup costs for a seed j

The capacitated vehicle routing problem

Ingredients

- y: clustering of customers
  - $y_{ij} = 1$ if i is connected to j, $y_{jj} = 1$ if j is a seed
- $c_1(x, y)$: represented by cost matrix D
  - d_{ij} are the connection costs of location i to seed j
  - d_{jj} are the setup costs for a seed j
- $F_1(x, c_1)$: solve the CCLP using the MIP formulation

The capacitated vehicle routing problem

Ingredients

- y: clustering of customers
  - $y_{ij} = 1$ if i is connected to j, $y_{jj} = 1$ if j is a seed
- $c_1(x, y)$: represented by cost matrix D
  - d_{ij} are the connection costs of location i to seed j
  - d_{jj} are the setup costs for a seed j
- $F_1(x, c_1)$: solve the CCLP using the MIP formulation
- $F_2(x, y)$: solve the TSP using MIP formulation with lazy constraint subtour elimination

The capacitated vehicle routing problem

Ingredients

- y: clustering of customers
  - $y_{ij} = 1$ if i is connected to j, $y_{jj} = 1$ if j is a seed
- $c_1(x, y)$: represented by cost matrix D
  - d_{ij} are the connection costs of location i to seed j
  - d_{jj} are the setup costs for a seed j
- $F_1(x, c_1)$: solve the CCLP using the MIP formulation
- $F_2(x, y)$: solve the TSP using MIP formulation with lazy constraint subtour elimination
- $\tilde{c}_1$: lower bound is hard, so use 'optimism parameter' γ (t_{ij} is distance, 0 is depot)
  - $d_{ij} = \gamma(t_{0i} + t_{ij} - t_{0j})$, $d_{jj} = 2t_{0j}$

The capacitated vehicle routing problem

Ingredients

- y: clustering of customers
  - $y_{ij} = 1$ if i is connected to j, $y_{jj} = 1$ if j is a seed
- $c_1(x, y)$: represented by cost matrix D
  - d_{ij} are the connection costs of location i to seed j
  - d_{jj} are the setup costs for a seed j
- $F_1(x, c_1)$: solve the CCLP using the MIP formulation
- $F_2(x, y)$: solve the TSP using MIP formulation with lazy constraint subtour elimination
- $\tilde{c}_1$: lower bound is hard, so use 'optimism parameter' γ (t_{ij} is distance, 0 is depot)
  - $d_{ij} = \gamma(t_{0i} + t_{ij} - t_{0j})$, $d_{jj} = 2t_{0j}$
- Updating mechanism: scale d_{ij} ($i \neq j$) according to observed route length
  - Both for last solution *and* incumbent; illustration on next slides

The capacitated vehicle routing problem

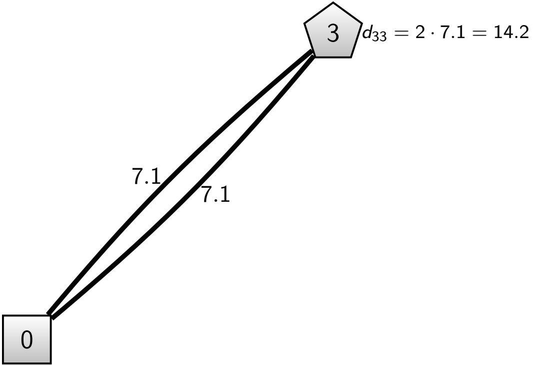


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

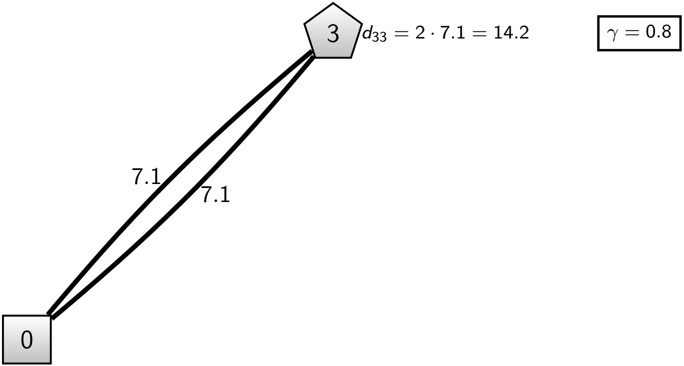


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

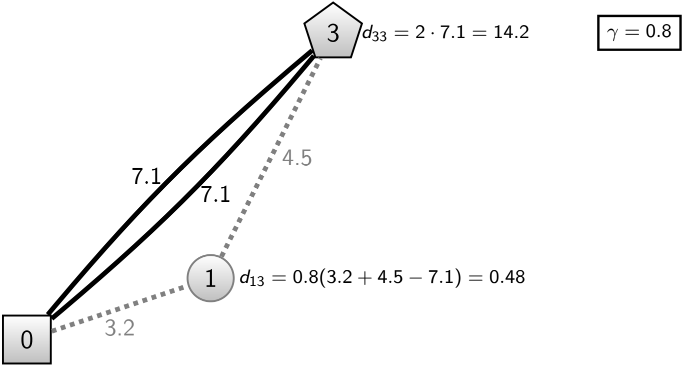


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

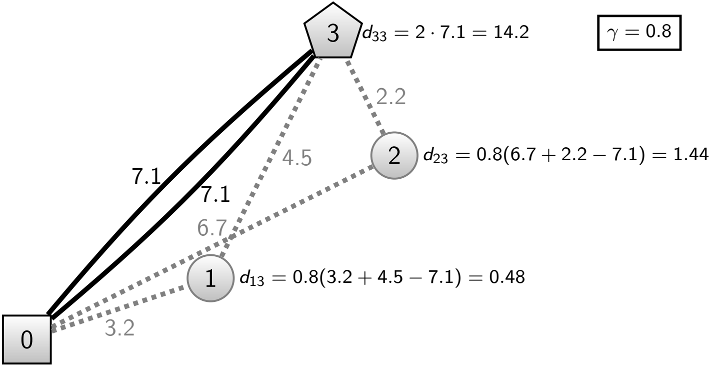


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

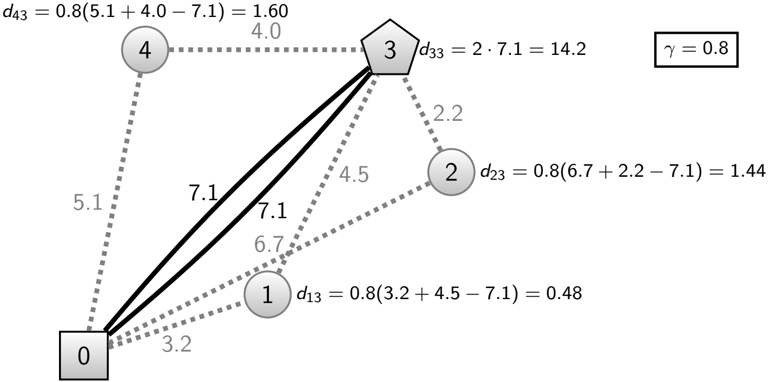


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

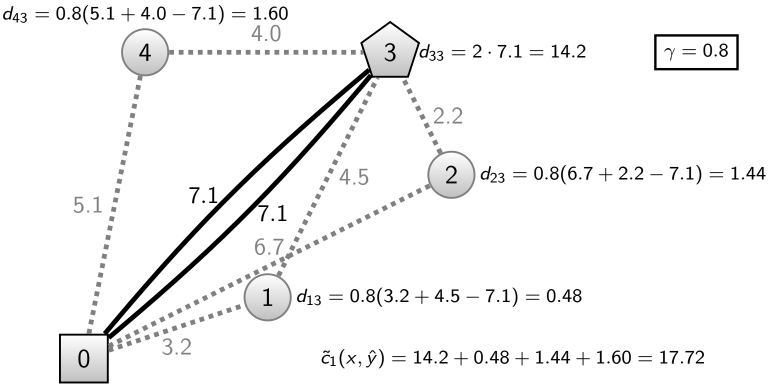


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

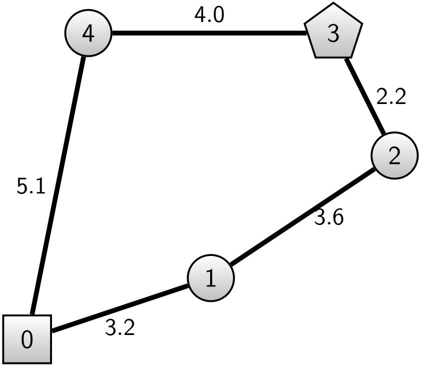
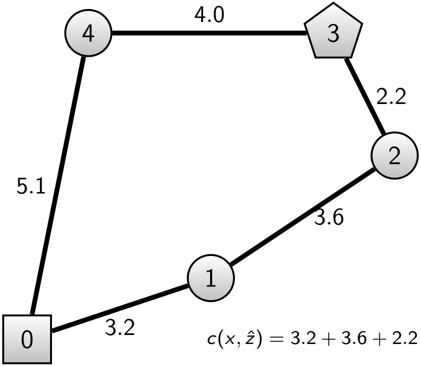


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem



$$c(x, \hat{z}) = 3.2 + 3.6 + 2.2 + 4.0 + 5.1 = 18.1$$

Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

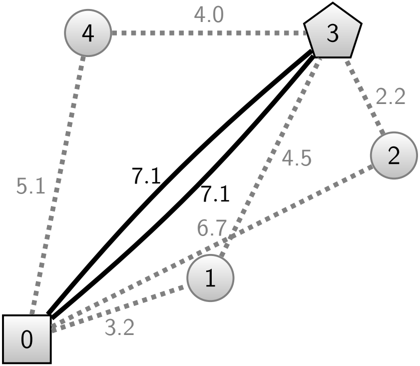
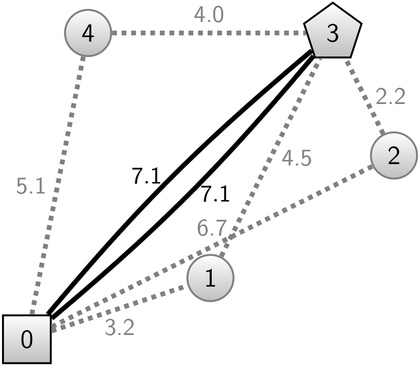


Figure: Illustration of the updating mechanism

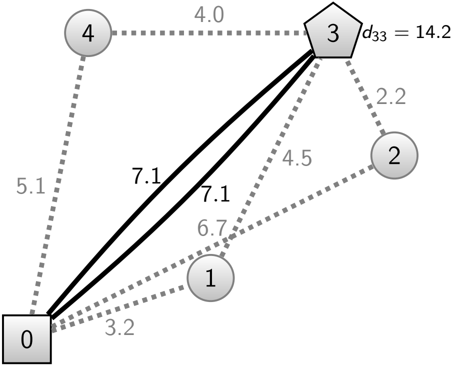
The capacitated vehicle routing problem



$$\frac{18.1-14.2}{17.72-14.2} = \frac{3.9}{3.52} \approx 1.1$$

Figure: Illustration of the updating mechanism

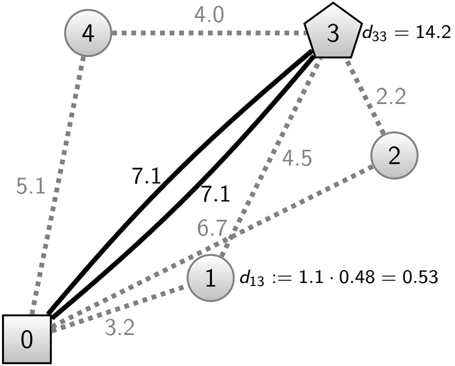
The capacitated vehicle routing problem



$$\boxed{\frac{18.1-14.2}{17.72-14.2} = \frac{3.9}{3.52} \approx 1.1}$$

Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem



$\boxed{\frac{18.1-14.2}{17.72-14.2} = \frac{3.9}{3.52} \approx 1.1}$

Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

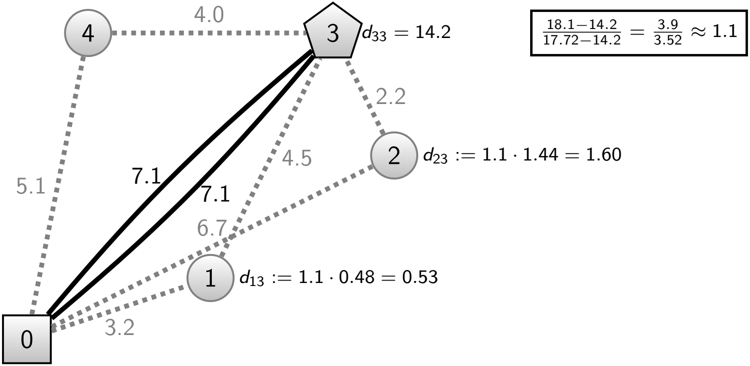


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

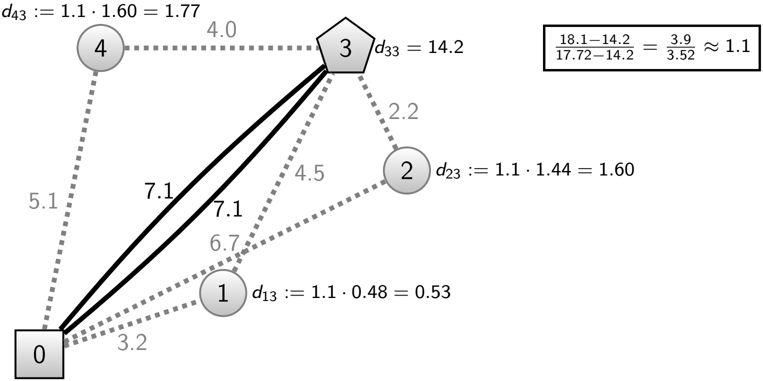


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

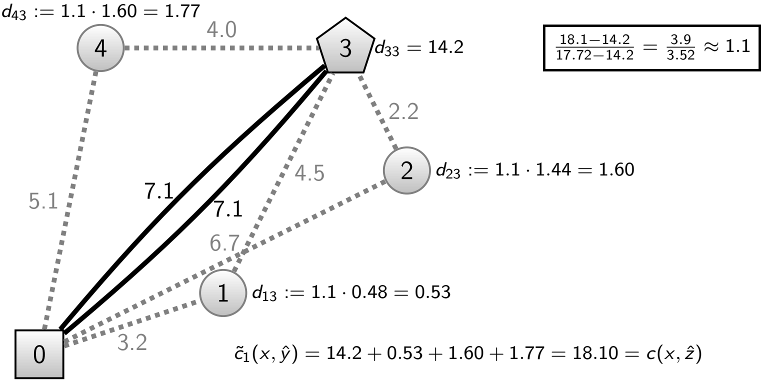


Figure: Illustration of the updating mechanism

The capacitated vehicle routing problem

Convergence

- If γ small, other options are explored
- Unfortunately diverges
- Therefore, keep incumbent as fall back

How to keep incumbent?

- Additionally ensure $\tilde{c}_1(x, y^*) = c(x, z^*)$
- If different seeds, updates have no effect
- If same seeds, apply update for locations in intersection of incumbent and last route
- Apply updates for connecting locations only in one of both to the intersection route

The capacitated vehicle routing problem

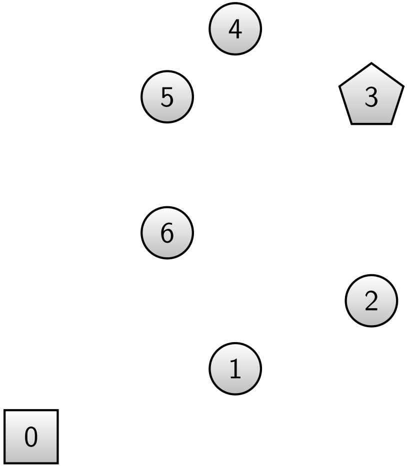


Figure: Illustration of the updating mechanism, if the last solution and incumbent share a seed customer

The capacitated vehicle routing problem

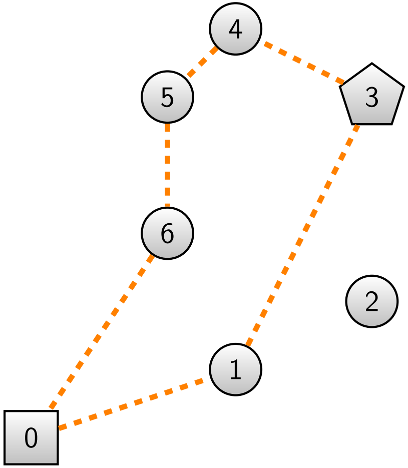


Figure: Illustration of the updating mechanism, if the last solution and incumbent share a seed customer

The capacitated vehicle routing problem

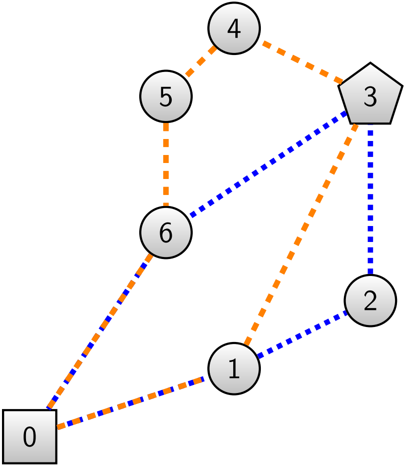


Figure: Illustration of the updating mechanism, if the last solution and incumbent share a seed customer

The capacitated vehicle routing problem

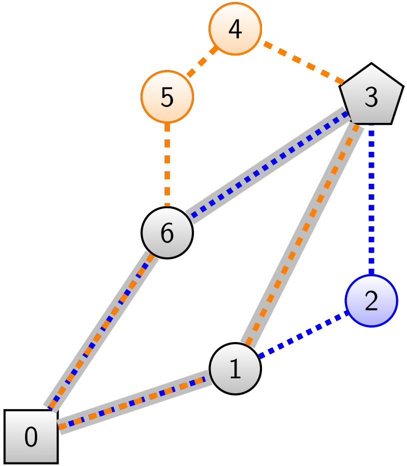


Figure: Illustration of the updating mechanism, if the last solution and incumbent share a seed customer

The capacitated vehicle routing problem

Results for instances from Christofides and Eilon (1969)

Instance	Initial solution		Incumbent		Convergence	
	Val	Gap	Val	Gap	It	Time
vrp1-50	539.22	2.78 %	539.22	2.78 %	2	0.4 s
vrp2-75	841.93	0.80 %	841.93	0.80 %	7	137.5 s
vrp3-100	858.54	3.92 %	832.43	0.76 %	8	169.7 s
vrp4-100	832.83	1.62 %	826.90	0.90 %	4	69.2 s
vrp5-120	1052.54	1.00 %	1045.42	0.32 %	7	428.5 s
vrp6-150	1076.66	4.69 %	1053.24	2.41 %	10	878.5 s

The capacitated vehicle routing problem

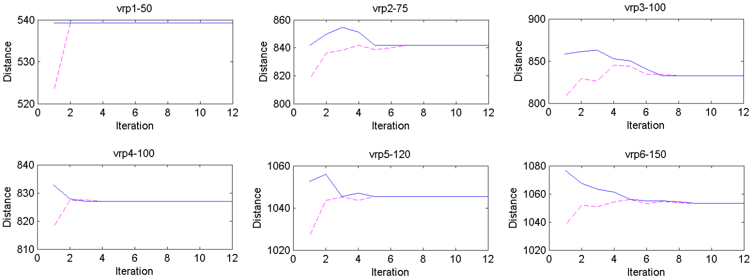


Figure: Solution quality vs. iterations

Agenda

The resource assignment problem

The problem

- Variant of the multiple trip vehicle routing problem (MTVRP)
- Given a depot, a set of customers, capacity constraints, time windows and driving time constraints
- For a fixed time horizon, create routes for each trailer, where each trailer can visit the depot multiple times
- Find an assignment of drivers (resources) to the trailers that satisfies driving time constraints
- Depending on how the trailer routes are constructed, assigning one driver to each trailer may be suboptimal or even infeasible

The resource assignment problem

Two phase decomposition

- Phase 1: creation of trailer routes using parallel cheapest insertion
- Phase 2: assignment of resources to trailers using column generation

The resource assignment problem

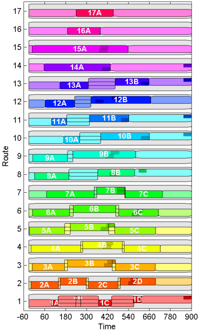


Figure: Trailer routes

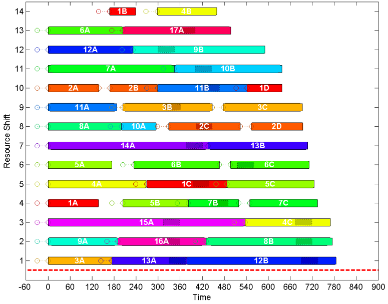


Figure: Resource assignments

The resource assignment problem

Ingredients

- y: the routes for the trailers

The resource assignment problem

Ingredients

- y: the routes for the trailers
- $c_1(x, y)$: non trivial!
  - Estimate 'topline' using a lower bound and a bias
  - Estimate no. drivers from topline height in critical interval

The resource assignment problem

Ingredients

- y: the routes for the trailers
- $c_1(x, y)$: non trivial!
  - Estimate 'topline' using a lower bound and a bias
  - Estimate no. drivers from topline height in critical interval
- $F_1(x, c_1)$: trailer routes using parallel cheapest insertion
  - Adapt insertion cost to find result with low topline
  - Repeat for a number of different weights, with last solution bias and incumbent bias

The resource assignment problem

Ingredients

- y: the routes for the trailers
- $c_1(x, y)$: non trivial!
  - Estimate 'topline' using a lower bound and a bias
  - Estimate no. drivers from topline height in critical interval
- $F_1(x, c_1)$: trailer routes using parallel cheapest insertion
  - Adapt insertion cost to find result with low topline
  - Repeat for a number of different weights, with last solution bias and incumbent bias
- $F_2(x, y)$: assign the resources to trailers using column generation

The resource assignment problem

Ingredients

- y: the routes for the trailers
- $c_1(x, y)$: non trivial!
  - Estimate 'topline' using a lower bound and a bias
  - Estimate no. drivers from topline height in critical interval
- $F_1(x, c_1)$: trailer routes using parallel cheapest insertion
  - Adapt insertion cost to find result with low topline
  - Repeat for a number of different weights, with last solution bias and incumbent bias
- $F_2(x, y)$: assign the resources to trailers using column generation
- $\tilde{c}_1$: a priori we have no bias and use the default criterium without feedback

The resource assignment problem

Ingredients

- y: the routes for the trailers
- $c_1(x, y)$: non trivial!
  - Estimate 'topline' using a lower bound and a bias
  - Estimate no. drivers from topline height in critical interval
- $F_1(x, c_1)$: trailer routes using parallel cheapest insertion
  - Adapt insertion cost to find result with low topline
  - Repeat for a number of different weights, with last solution bias and incumbent bias
- $F_2(x, y)$: assign the resources to trailers using column generation
- $\tilde{c}_1$: a priori we have no bias and use the default criterium without feedback
- Updating mechanism: take bias from observed topline

The resource assignment problem

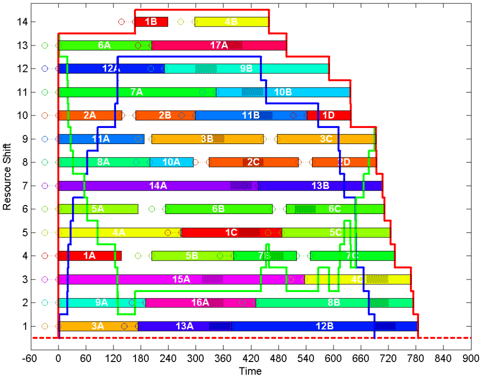


Figure: Illustration of the topline

The resource assignment problem

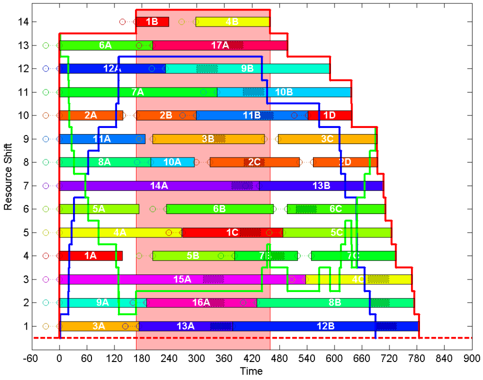


Figure: Illustration of the area under the topline at the critical interval

The resource assignment problem

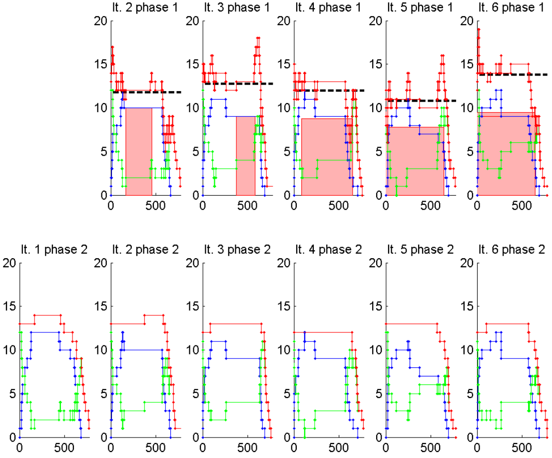


Figure: Illustration of the estimation using the topline

The resource assignment problem

Results

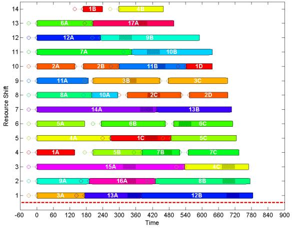


Figure: Initial: 17 trailers, 14 drivers

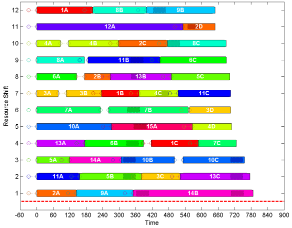


Figure: Final: 15 trailers, 12 drivers

The resource assignment problem

Instance	Initial solution	Stopping criteria				10 iterations		
	Drv (Trl)	Drv (Trl)	Impr.	Found	Stop	Drv (Trl)	Impr.	Found
C101	21 (23)	20 (23)	4.76 %	2	8	19 (22)	9.52 %	9
C102	19 (22)	17 (19)	10.53 %	3	4	17 (19)	10.53 %	3
C103	18 (21)	15 (18)	16.67 %	3	5	15 (16)	16.67 %	6
C104	14 (17)	12 (15)	14.29 %	4	6	12 (15)	14.29 %	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
RC205	21 (21)	18 (20)	14.29 %	3	5	18 (20)	14.29 %	3
RC206	20 (21)	20 (21)	0.00 %	1	2	17 (18)	15.00 %	3
RC207	17 (19)	17 (18)	0.00 %	2	4	17 (18)	0.00 %	2
RC208	15 (16)	15 (16)	0.00 %	1	3	15 (16)	0.00 %	1
Average			4.96 %	2.0	3.7		7.55 %	4.1

Table: Results for the resource assignment problem

Agenda

Discussion

Conclusions

- Feedback is very effective!
- Solution quality improves within few iterations
- Efforts needed to achieve convergence depend on difficulty of problem
- Structured way of exploring alternatives by repetition, not going into details

Discussion

Conditions

- The structural quality of $\tilde{c}_1$
- The quality of F_1 and F_2
- The effectiveness of the updating mechanism
- The level of optimism in the a priori belief $\tilde{c}_1$

Discussion

Further research

- Conditions and proofs for convergence
- Redefine $\tilde{c}_1$ to only describe *precedence relation* on y
- Formally consider multiple objectives
- Hybrid heuristic/exact approach for both phases