ATMOS 2007

Robust Algorithms and Price of Robustness in Shunting Problems

Serafino Cicerone, Gianlorenzo D'Angelo, Gabriele Di Stefano, Daniele Frigioni

Dipartimento di Ingegneria Elettrica e dell'Informazione
Università delL'Aquila, Italy

Alfredo Navarra

Dipartimento di Matematica e Informatica
Università di Perugia, Italy

Outline

- Robustness Definition
- Shunting Definition
  - Considered Cases
- Robust Shunting for Bounded Disruptions
- Conclusion & Future Work

Robustness [based on ARRIVAL Deliverable D1.2L]

- A plan/solution for an optimization problem is called robust against
  - a specified model of imperfect knowledge

  and for
  - a specified family of algorithms/policies that can be applied for recovering purposes

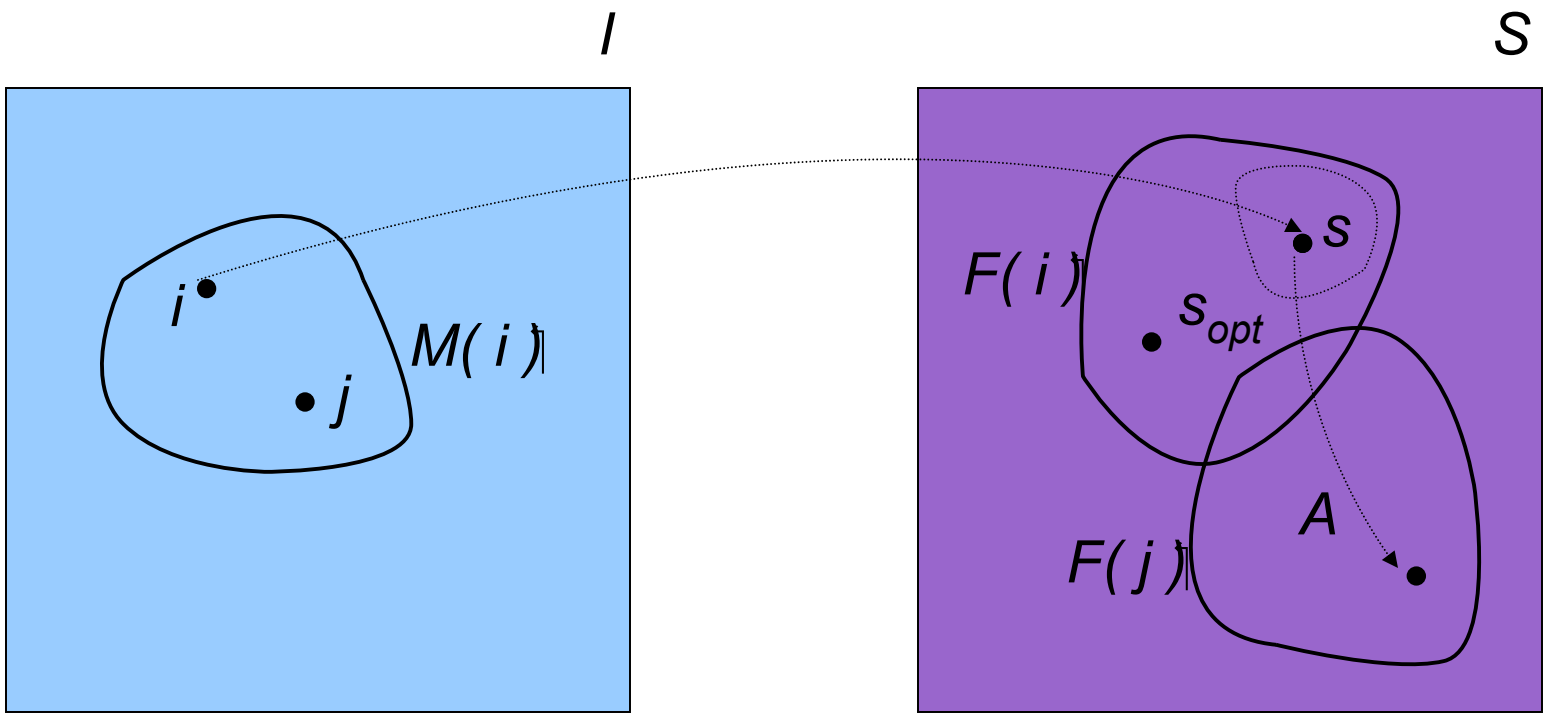
  if it is able to provide a solution after/while the knowledge becomes perfect

Robustness Problem

- Given an optimization problem *P* defined by
  - *I*, instances of P
  - *F*, the function that for each instance *i* outputs the set of feasible solutions for *i*
  - *f*: $S \rightarrow R$ objective function where $S = U_{i \text{ in } I} F(i)$
- A robustness problem R_P is defined by
  - *P*, optimization problem
  - *M*: $I \rightarrow 2^I$, modification function
  - A_{rec}, class of recovery algorithms for *P*
    - Each element goes from $I \times S \times I \rightarrow S$
- Given an instance *i* in *I* of *P*, *s* in *F(i)* is a robust solution for *i* with respect to R_P iff

$$\exists \mathrm{A} \in \mathrm{A}_{\mathrm{rec}} : \forall j \in M(i),\ \mathrm{A}(i,s,j) \in F(j)$$

Robustness Problem



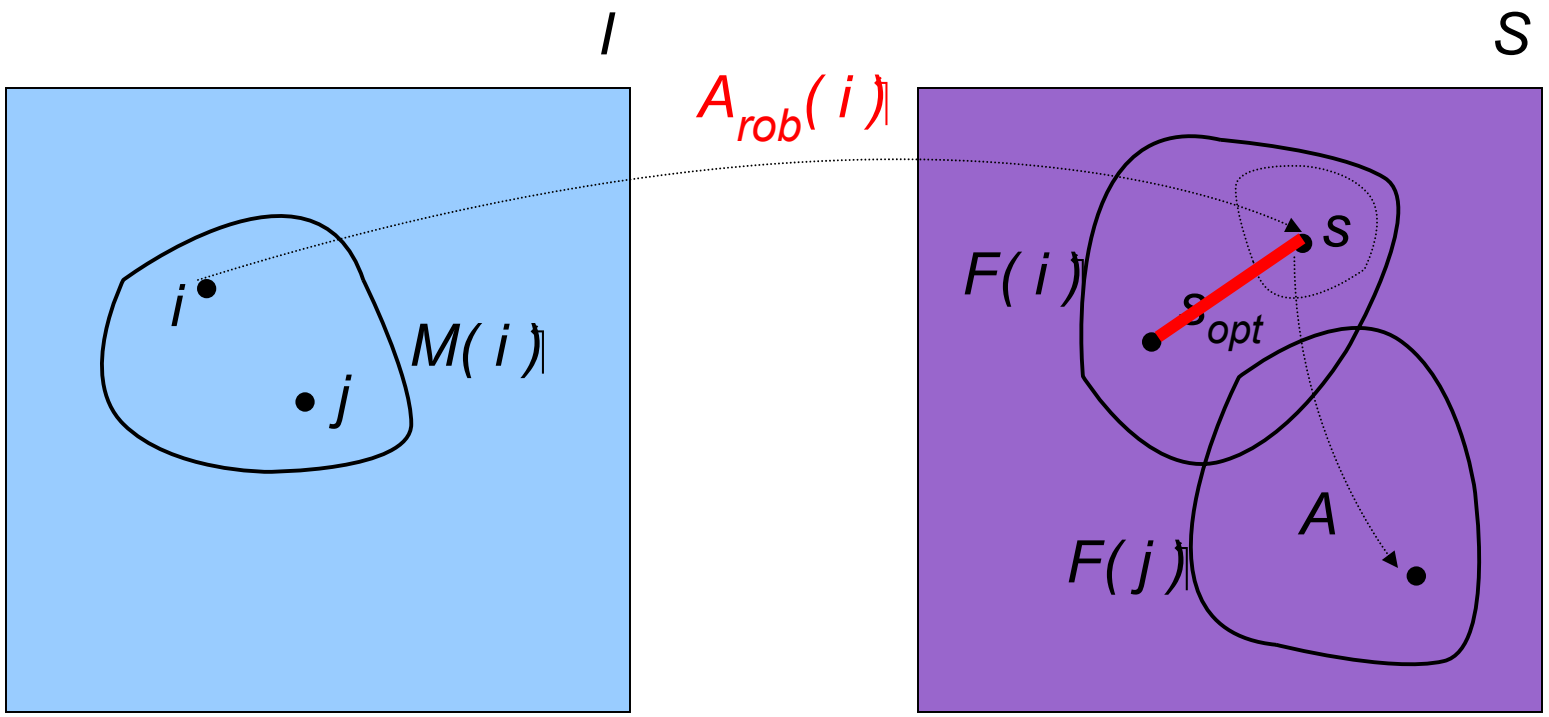
Robustness Definitions

- Given $R_P = (P, M, A_{rec})$, a robust algorithm for R_P is any algorithm A_{rob} such that for every i in I, $A_{rob}(i)$ is robust with respect to R_P.
  - Strict robustness: $\forall (i,s) \in \mathrm{I\ x\ S}, \forall \mathrm{j} \in M(i), \mathrm{A(i,s,j)} = \mathrm{s}$

- The Price of Robustness of a robust algorithm A_{rob} for a robustness problem R_P is given by

$$PoR(R_P, A_{rob}) = \max_{i \in I} \left\{ \frac{f(A_{rob}(i))}{\min\{\mathrm{f(x)} \mid \mathrm{x} \in F(i)\}} \right\}$$

Price of Robustness



Robustness optimality

- The Price of Robustness of a robustness problem R_P is given by

$$PoR(R_P)=min\{PoR(R_P,A_{rob}) : A_{rob} \text{ is a robust algorithm for } R_P\}$$

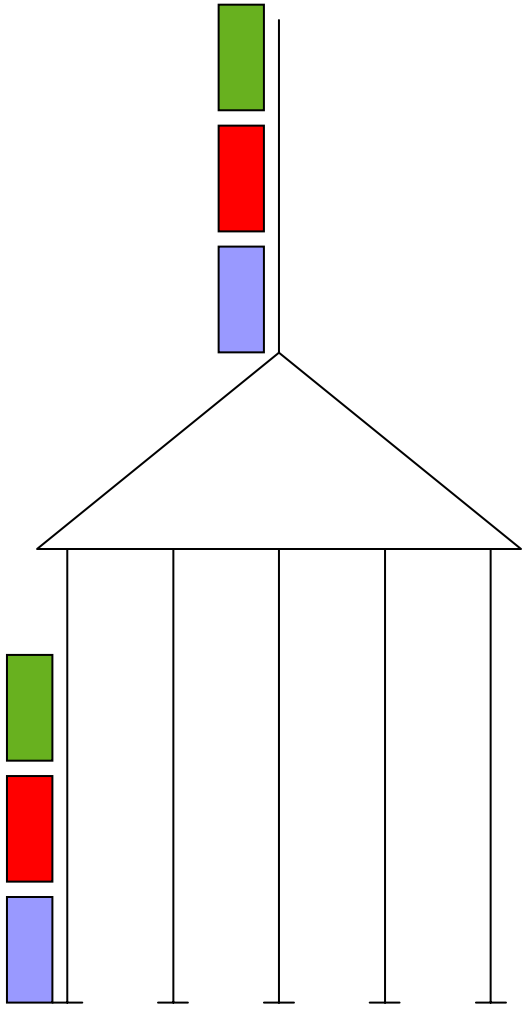
- A robust algorithm A_{rob} is exact for a robustness problem R_P if $PoR(R_P,A_{rob})=1$

- A robust algorithm A_{rob} is optimal for a robustness problem R_P if $PoR(R_P,A_{rob})=PoR(R_P)$

Shunting yard

- Area where trains or single cars can be shoved or turned off or moved from one track to another

Definitions & Notation [Jacob '07]



- 1 access track
- *w*: number of classification tracks
- *c*: length of a track (contains c cars)
- *n*: number of cars composing a train
- *h*: number of track pulls

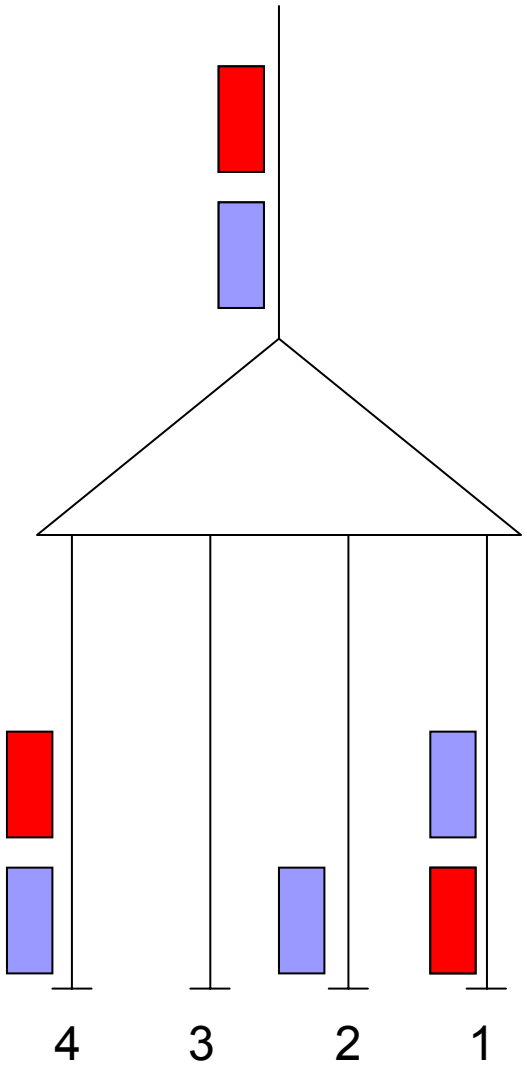
Track-pull operation

- Connect cars of one track into a pseudo-train
- Pull the pseudo-train over the hump
- Disconnect the cars of the pseudo-train
- Push the pseudo-train slowly over the hump, yielding single cars toward the tracks
- Control the switches such that every single car goes to a specified classification track

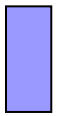
From Shunting to Coding

- In order to specify a shunting plan we need:
  - Track pulls sequence $T = \{1, \ldots, h\}$
    - Each track may appear many times in the sequence
    - h represents the number of logical tracks and determines the cost of a shunting plan
  - Cars allocation for each track pull
    - A binary string $b_1, \ldots, b_h$ for each car can be used to specify the sequence of tracks visited by the corresponding car

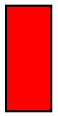
Example



Let T={1,2,3,2,1,4} → h=6



100011



110001

412321

Observations

- If two cars are associated with the same binary string, then they will appear in the outgoing train T_{out} in the same order in which they appeared in the incoming train T_{in}
- Two or more cars that appear in T_{out} in the same order as they arrive in T_{in} are called a *run*
- If the order of the n incoming cars of a train is reversed with respect to the outgoing train, then the number of runs $r = n$
- In a shunting plan, for each code x, a pure run is the maximal set of cars associated with x.

Considered Cases [Jacob '07]

- Case 1- *c* bounded
- Case 2- *w* bounded
- Case 3- *c* and *w* unbounded

	Case 1 ***c bounded***	***Case 2*** ***w bounded***	***Case 3*** ***c, w unbounded***
r=n	***opt***	***opt***	***opt***
any r	***NP-Hard, 2-apx***	***opt***	***opt***

Recovery algorithm classes A_{rec}

- $A_{rec}^{1}: \forall A \in A_{rec}^{1}, \forall (i,s) \in IxS, \forall j \in M(i), A(i,s,j) = s$
  there are no recovery strategies to apply (strict robustness)
- $A_{rec}^{2}: \forall A \in A_{rec}^{2}, \forall (i,s) \in IxS, \forall j \in M(i), A(i,s,j) = s'$
  where s' may differ from s by at most one code, i.e., at most one pure run may be assigned with a new code of the same length
- $A_{rec}^{3}: \forall A \in A_{rec}^{3}, \forall (i,s) \in IxS, \forall j \in M(i), A(i,s,j) = s'$
  where s' may differ from s by all the set of codes

Considered Changes (*M*)

- Car with unexpected incoming position
- New incoming car
- Missing car
- Unavailable track

Car with unexpected incoming position, Case 1, A^1_{rec}

- **Lemma 1**. Let *v* be a car arriving at the hump in a different position than expected. At most one additional pure run must be differently managed with respect to the expected case.
- **Lemma 2**. For every input train T_{in} and considering A^1_{rec}, any robust shunting algorithm A_{rob} must provide a unique code to each car of T_{in}.
- **Theorem 1**. Considering A^1_{rec} there exists an optimal robust shunting algorithm A_{rob} such that

$$PoR(R_P, A_{rob}) = \max_{i \in I} \frac{opt(n_i, c)}{opt(r_i, c)}$$

Car with unexpected incoming position, Case 1, A^2_{rec}

- **Theorem 2**. Considering A^2_{rec} there exists a polynomial robust shunting algorithm A_{rob} such that

$$PoR(R_P, A_{rob}) = \max_{i \in I} \frac{apx(r_i, c) + 1}{opt(r_i, c)} \leq 2 + \max_{i \in I} \frac{1}{opt(r_i, c)} = 3$$

Note: nothing better for A^3_{rec} has been found.

Car with unexpected incoming position, Cases 2 and 3, A^2_{rec}

- **Theorem 3**. In Case 2- (3 resp.), considering A^2_{rec}, there exists a polynomial robust shunting algorithm A_{rob} s.t.

$$PoR(R_P, A_{rob}) = \max_{i \in I} \frac{opt(r_i, w) + 1}{opt(r_i, w)} = 1 + \max_{i \in I} \frac{1}{opt(r_i, w)} = 2$$

$$\left(PoR(R_P, A_{rob}) = \max_{i \in I} \frac{opt(r_i) + 1}{opt(r_i)} = 1 + \max_{i \in I} \frac{1}{opt(r_i)} = 2 \quad \text{resp.} \right)$$

- **Theorem 4**. In all Cases, and considering A^2_{rec}, $PoR(R_P) \geq 2$

New Incoming Car

- **Theorem 5**. If we consider A^1_{rec}, no robust shunting algorithm *exists*.
- **Theorem 6**. In Case 1-, and considering A^2_{rec},

$$PoR(R_P, A_{rob}) = \max_{i \in I} \frac{opt(n_i + 1, c - 1) + 1}{opt(r_i, c)}$$

- **Theorem 6**. In Case 2- (3- resp.), and considering A^2_{rec},

$$PoR(R_P, A_{rob}) = \max_{i \in I} \frac{opt(n_i + 1, w) + 1}{opt(r_i, w)}$$

$$\left(PoR(R_P, A_{rob}) = \max_{i \in I} \frac{opt(n_i + 1) + 1}{opt(r_i)} \quad \text{resp.} \right)$$

New Incoming Car, Case 1, A^2_{rec}

Conclusion

- Robustness with recoverability has been defined for optimization plans
- In particular we have studied robustness of shunting plans
- Many scenarios arise from this study:
  - Other shunting yard types
  - Other optimization plans (schedules, …)
  - Other possible imperfections in knowledge (*M*)
  - …