Decomposition methods for large-scale programming IV: 
a challenge to you

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

ÖGOR Workshop for PhD students 2022
Krems an der Donau — August 2, 2022
Meta–Outline

- Part I: basic (old) theory
- Part II: some (new) tools
- Part III: advanced (not-so-old) theory
- Part IV: a challenge to you
Outline – Part IV

1. A small challenge to you

2. A bigger challenge to you

3. Conclusions (for good)
Constrained SP \equiv (\text{single-commodity}) \text{ shortest path} \cap \text{binary knapsack}

\begin{align*}
\text{min} & \sum_{(i,j) \in A} c_{ij} x_{ij} \\
\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} & = \begin{cases} 
-1 & \text{if } i = s \\
1 & \text{if } i = t \\
0 & \text{otherwise}
\end{cases} \quad i \in N \\
\sum_{(i,j) \in A} w_{ij} x_{ij} & \leq W \\
x_{ij} & \in \{0, 1\} \quad (i, j) \in A
\end{align*}

Perhaps simplest significant optimization problem with “split personality”

Many variants with more complex constraints (\textit{simple path}, \ldots)

Fundamental for D-W/CG of many problems, e.g. VRP
A small challenge to you: Constrained Shortest Path I

- Constrained SP ≡ (single-commodity) shortest path ∩ binary knapsack

\[
\begin{align*}
\min & \sum_{(i,j) \in A} C_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = \begin{cases} 
-1 & \text{if } i = s \\
1 & \text{if } i = t \\
0 & \text{otherwise} 
\end{cases} \quad i \in N \quad (2) \\
\sum_{(i,j) \in A} w_{ij} x_{ij} & \leq W \quad \text{for } (i, j) \in A \quad \text{(3)} \\
x_{ij} & \in \{0, 1\} \quad \text{for } (i, j) \in A \quad \text{(4)}
\end{align*}
\]

- Perhaps simplest significant optimization problem with “split personality”

- Many variants with more complex constraints (simple path, ...)

- Fundamental for D-W/CG of many problems, e.g. VRP

**Simple Challenge, part I:**
which Lagrangian dual should be solved?
Simple Challenge, part II:
implement a Lagrangian approach to CSP
to evaluate the time/gap trade-offs
w.r.t. continuous relaxation
A small challenge to you: Constrained Shortest Path II

Simple Challenge, part II:
implement a Lagrangian approach to CSP
to evaluate the time/gap trade-offs
w.r.t. continuous relaxation

Suggestions:

- Use SMS++ (surprised?)

- Let’s first discuss the plan (there are several options)

- Look to tests/LagrangianDualSolver_* or
tests/compare_formulations for suitable main and configs

- Ask me for the ORLib instances, translated in netCDF if you prefer
1. A small challenge to you

2. A bigger challenge to you

3. Conclusions (for good)
Why I bother

- Mainly a selfish reason and an altruistic one

  Selfish reason: I’ve made decomposition stuff for many years, but (almost) nobody cares
  
  Altruistic version of the selfish reason: decomposition may be useful for many, but it’s too difficult to do in practice
  
  Altruistic reason: stop a lot of good programming going to waste

  Programming is hard, good programming is harder, optimization is hard, good programming for optimization is extremely hard

  Often done early in the career (you), pays too little career-wise (wrong)

  Many months/years of your life go down the drain because developing, maintaining and supporting a good package is not “cost-effective”

  Selfish version of the altruistic reason: if just a fraction of all this work became available many more decomposition approaches would be possible and my work would make an impact
Mainly a **selfish reason** and an **altruistic one**

**Selfish reason:** I’ve made decomposition stuff for many years, but (almost) nobody cares

**Altruistic reason:** stop a lot of good programming going to waste.

Programming is hard, good programming is harder, optimization is hard, good programming for optimization is extremely hard.

Often done early in the career (you), pays too little career-wise (wrong).

Many months/years of your life go down the drain because developing, maintaining and supporting a good package is not “cost-effective”.

**Selfish version of the altruistic reason:** if just a fraction of all this work became available many more decomposition approaches would be possible and my work would make an impact.
Why I bother

- Mainly a **selfish reason** and an **altruistic one**
- **Selfish reason**: I’ve made decomposition stuff for many years, but (almost) nobody cares
- **Altruistic version of the selfish reason**: decomposition may be useful for many, but it’s too difficult to do in practice

A. Frangioni (DI — UniPi)
Why I bother

- Mainly a selfish reason and an altruistic one

- **Selfish reason:** I’ve made decomposition stuff for many years, but (almost) nobody cares

- **Altruistic version of the selfish reason:** decomposition may be useful for many, but it’s too difficult to do in practice

- **Altruistic reason:** stop a lot of good programming going to waste
Why I bother

- Mainly a **selfish reason** and an **altruistic one**
  
  **Selfish reason:** I’ve made decomposition stuff for many years, but (almost) **nobody cares**
  
  **Altruistic version of the selfish reason:** decomposition may be useful for many, but it’s too difficult to do in practice
  
  **Altruistic reason:** stop a lot of good programming going to waste
  
  Programming is hard, good programming is harder, optimization is hard, good programming for optimization is extremely hard
  
  Often done early in the career (you), pays too little career-wise (wrong)
  
  Many months/years of your life go down the drain because developing, maintaining and supporting a good package is not “cost-effective”
Why I bother

- Mainly a **selfish reason** and an **altruistic one**
  - **Selfish reason**: I’ve made decomposition stuff for many years, but (almost) nobody cares
  - **Altruistic version of the selfish reason**: decomposition may be useful for many, but it’s too difficult to do in practice
  - **Altruistic reason**: stop a lot of good programming going to waste
  - Programming is hard, good programming is harder, optimization is hard, good programming for optimization is extremely hard
  - Often done early in the career (you), pays too little career-wise (wrong)
  - Many months/years of your life go down the drain because developing, maintaining and supporting a good package is not “cost-effective”
  - **Selfish version of the altruistic reason**: if just a fraction of all this work became available many more decomposition approaches would be possible and my work would make an impact
A bigger challenge to you: make an impact

- You are at the beginning of your career, be it academic or industrial
- Your choices about your research may be somewhat limited (or not)

A. Frangioni (DI — UniPi)
A bigger challenge to you: make an impact

- You are at the beginning of your career, be it academic or industrial
- Your choices about your research may be somewhat limited (or not)
- In the academia, “publish or perish” will push you to produce anything
- In the industria, the boss and money will push you to produce anything

A. Frangioni (DI — UniPi)
A bigger challenge to you: make an impact

- You are at the beginning of your career, be it academic or industrial
- Your choices about your research may be somewhat limited (or not)
- In the academia, “publish or perish” will push you to produce anything
- In the industria, the boss and money will push you to produce anything

You are better than that, so resist
A bigger challenge to you: make an impact

- You are at the beginning of your career, be it academic or industrial
- Your choices about your research may be somewhat limited (or not)
- In the academia, “publish or perish” will push you to produce anything
- In the industria, the boss and money will push you to produce anything

**You are better than that, so resist**

- Try to ask yourself what impact your work will have, who will benefit
- If it’s “none but myself”, perhaps you could reconsider
A bigger challenge to you: make an impact

- You are at the beginning of your career, be it academic or industrial
- Your choices about your research may be somewhat limited (or not)
- In the academia, “publish or perish” will push you to produce anything
- In the industria, the boss and money will push you to produce anything

You are better than that, so resist

- Try to ask yourself what impact your work will have, who will benefit
- If it’s “none but myself”, perhaps you could reconsider
- But you have to eat/get a career: one occasional bad paper/work is OK
- Just try not to make it an habit: there is too much of that already
Conclusions

(Part IV)
Conclusions and a lot of future work, for you and me

- Decomposition a large set of different techniques focusing on **structure**
- Not useful in all cases, somewhat useful often, necessary at times
- Theory (more or less) established, **software support** always been the issue
Conclusions and a lot of future work, for you and me

- Decomposition a large set of different techniques focusing on **structure**
- Not useful in all cases, somewhat useful often, necessary at times
- Theory (more or less) established, **software support** always been the issue
- SMS++ trying to improve on that, some steps of a long and winding road
- Perhaps useful already for some fringe huge-scale applications
- **Could** become really useful after having attracted mindshare (very hard)
Conclusions and a lot of future work, for you and me

- Decomposition a large set of different techniques focusing on structure
- Not useful in all cases, somewhat useful often, necessary at times
- Theory (more or less) established, software support always been the issue
- SMS++ trying to improve on that, some steps of a long and winding road
- Perhaps useful already for some fringe huge-scale applications
- Could become really useful after having attracted mindshare (very hard)
- Yet, I think it (imperfectly) tries to address some real needs:
  - improve collaboration and code reuse, reduce huge code waste
  - make decomposition approaches much more easily available
  - help a much-needed higher uptake of parallel methods
  - ...many other personal feverish dreams\textsuperscript{[1]} you don’t want to hear
Conclusions and a lot of future work, for you and me

- Decomposition a large set of different techniques focusing on structure
- Not useful in all cases, somewhat useful often, necessary at times
- Theory (more or less) established, software support always been the issue
- SMS++ trying to improve on that, some steps of a long and winding road
- Perhaps useful already for some fringe huge-scale applications
- Could become really useful after having attracted mindshare (very hard)
- Yet, I think it (imperfectly) tries to address some real needs:
  - improve collaboration and code reuse, reduce huge code waste
  - make decomposition approaches much more easily available
  - help a much-needed higher uptake of parallel methods
  - . . . many other personal feverish dreams[1] you don’t want to hear

Aiming for a big slice of the cake no-one else seems to want, one should seriously consider if they are the ones who are sane.
Conclusions and a lot of future work, for you and me

- Decomposition a large set of different techniques focusing on **structure**
- Not useful in all cases, somewhat useful often, necessary at times
- Theory (more or less) established, **software support** always been the issue
- **SMS++** trying to improve on that, some steps of a long and winding road
- Perhaps useful already for some fringe huge-scale applications
- **Could** become really useful _after having attracted mindshare_ (very hard)
- Yet, I think it (imperfectly) tries to address some real needs:
  - improve collaboration and code reuse, reduce huge code waste
  - make decomposition approaches much more easily available
  - help a much-needed higher uptake of **parallel methods**
  - ... many other personal feverish dreams[^1] you don’t want to hear

- Aiming for a big slice of the cake no-one else seems to want, one should seriously consider if they are the ones who are sane.
- It’s **not a lie**, but perhaps it’s **poisoned**? A bite, anybody?