Integrating Algebraic Dynamic Programming in Combinatorial Optimization

Christopher Bacher, Günther Raidl
bacher@ac.tuwien.ac.at
www.ac.tuwien.ac.at/people/bacher
Algorithms and Complexity Group
TU Wien
December 2\textsuperscript{nd}, 2016
Dynamic Programming & Metaheuristics

Hybrid Metaheuristics often depend on Dynamic Programming for ...

- ... solving **subproblems** e.g. packing, shortest path
- ... enhancing **neighbourhood search** e.g. Dynasearch
- ... improving **recombination** operators in GAs e.g. memetic algorithms
- ... **decoding** solutions e.g. permutation encodings
Motivation

Some observations on Dynamic Programming ...

- DP is often encountered as solution technique
- DP has usually a problem-specific implementation
Motivation

Some observations on Dynamic Programming ...

- DP is often encountered as solution technique
- DP has usually a problem-specific implementation
- DP is often easier to describe ...
- ... than to implement
Some observations on Dynamic Programming ...

- DP is often encountered as solution technique
- DP has usually a problem-specific implementation
- DP is often easier to describe ...
- ... than to implement

Is something wrong with Dynamic Programming?
Algebraic Dynamic Programming (ADP)

Alternative view on Dynamic Programming (Giegerich et al., 2002)

- Formal grammar defines the search space by \textit{decomposition}
- Separates evaluation from search space declaration
- Works for sequence data (strings)—originally intended for bioinformatics
- Extension for set/general data structures available (Siederdissen et al., 2014/15)
Algebraic Dynamic Programming (ADP)

Alternative view on Dynamic Programming (Giegerich et al., 2002)

- Formal grammar defines the search space by **decomposition**
- Separates evaluation from search space declaration
- Works for sequence data (strings)—originally intended for bioinformatics
- Extension for set/general data structures available (Siederdissen et al., 2014/15)

**Whistle: a new solver framework for ADP**

- targeted for general combinatorial problems
- intended for integration in heuristics
Parts of an Algebraic Dynamic Program

- **Set of indexed terminal symbols**
  - Represents atomic objects of a solution

- **Set of indexed non-terminal symbols**
  - Each non-terminal is a **DP table**
    → addressed by the indices
  - Each indexed non-terminal represents a **state/compound object**

- **Set of production/decomposition rules**
  - Describes the **search space**
  - **Quantifiable**
  - Different types of **constraints**
Motivating Example: Knapsack

Given set of items $i \in \mathcal{I}$ and knapsack of max. weight $Q$

- $S$ ... Optimally packed knapsack
- $B_{i,q}$ ... Knapsack of weight at most $q$ with item $i$ considered last
  - $i$ ... integer
  - $q$ ... real-valued
- $\pi_i$ ... Item $i$
Motivating Example: Knapsack

Given set of items $i \in \mathcal{I}$ and knapsack of max. weight $Q$

- $S$ ... Optimally packed knapsack
- $B_{i,q}$ ... Knapsack of weight at most $q$ with item $i$ considered last
  - $i$ ... integer
  - $q$ ... real-valued
- $\pi_i$ ... Item $i$

**Decomposition Grammar**

$$S \rightarrow \pi_i B_{i,Q-w_i \geq 0}$$

$$B_{i,q} \rightarrow \pi_j B_{j,q-w_j \geq 0}$$

| $\epsilon$ | $\forall i \in \mathcal{I}$

| $\forall j \in \mathcal{I}, [i < j]$ |
Motivating Example: Knapsack

Given set of items \( i \in \mathcal{I} \) and knapsack of max. weight \( Q \)

- \( S \) ... Optimally packed knapsack
- \( B_{i,q} \) ... Knapsack of weight at most \( q \) with item \( i \) considered last
  - \( i \) ... integer
  - \( q \) ... real-valued
- \( \pi_i \) ... Item \( i \)

Decomposition Grammar

\[
S \rightarrow \pi_i B_{i,Q-w_i \geq 0} \\
B_{i,q} \rightarrow \pi_j B_{j,q-w_j \geq 0} \quad \forall j \in \mathcal{I}, [i < j]
\]

| \( i \) |
\hline
| \( j \) |
\hline
Dominance:

\[ A \prec B \equiv \sigma_{\text{value}}(A) < \sigma_{\text{value}}(B) \]
Heuristic Extensions
Search engines

Original ADP approach uses a fixed search order ...

- ... for solving “standalone” DP problems (bioinformatics)
- ... for proven-optimality nothing else is needed
Search engines

Original ADP approach uses a fixed search order …

- … for solving “standalone” DP problems (bioinformatics)
- … for proven-optimality nothing else is needed

Flexible search orders …

- … separate search from search space declaration
- … may find optimal solutions faster
- … have complexity benefits for some problems
Search engines

Original ADP approach uses a fixed search order ...
- ... for solving “standalone” DP problems (bioinformatics)
- ... for proven-optimality nothing else is needed

Flexible search orders ...
- ... separate search from search space declaration
- ... may find optimal solutions faster
- ... have complexity benefits for some problems

Whistle supports different search engines
- Depth-First Search
- Greedy Search
- A*
- ...
Index propagation

**Original ADP**
- Not explicit indices (by default)
- Automatic deduction
- Restricted to sequence/set data
- No index errors

**Whistle ADP**
- Explicit indices (by default)
- No automatic deduction
- Index propagators:
  - Sequence data
  - Cyclic permutations
  - Resource usage
  - ...
- Less index errors
- More flexibility

Christopher Bacher (AC @ TU Wien) Integrating Algebraic Dynamic Programming December 2nd, 2016 9 / 20
Partial Invalidation

DP approaches can be embedded in heuristics …

**Improvement heuristics …**

- …change parts of a solution
- …would require recalculation of the whole DP
Partial Invalidation

DP approaches can be embedded in heuristics ...

**Improvement heuristics ...**
- ...change parts of a solution
- ...would require recalculation of the whole DP

**Partial Invalidation ...**
- ... keeps track of dependencies of table cells
- ... allows for invalidation of parts of a table
  → on basis of changed terminal symbols
- ... can reuse remaining information
In Genetic Algorithms solution candidates …

- ... depend on their parents
- ... can reuse their information

**Shadowing** of table cells allows to redirect table access
→ less recomputation
Examples
Shortest Path

Given a graph $G = (V, A)$

- $S_{s,t}$ ... Shortest path from $s$ to $t$
- $P_{s,X,t}$ ... Path from $s$ to $t$ with unvisited nodes $X$
  - $s, t$ ... integer
  - $X$ ... set
- $a_{i,j}$ ... Arc from $i$ to $j$

$$S_{s,t} \rightarrow P_{s \setminus \{s,t\}, t}$$

$$P_{s,X,t} \rightarrow a_{s,x} P_{x, X \setminus x, t} \quad \forall x \in X, [(s, x) \in A]$$

$$\quad | a_{s,t} \quad [(s, t) \in A]$$
Given a graph $G = (V, A)$

- $S_{s,t}$ ... Shortest path from $s$ to $t$
- $P_{s,X,t}$ ... Path from $s$ to $t$ with unvisited nodes $X$
  - $s, t$ ... integer
  - $X$ ... set
- $a_{i,j}$ ... Arc from $i$ to $j$

\[
S_{s,t} \rightarrow P_{s \setminus \{s,t\},t}
\]
\[
P_{s,X,t} \rightarrow a_{s,x} P_{x,X \setminus x,t} \quad \forall x \in X, [(s, x) \in A]
\]
\[
| a_{s,t} \quad [(s, t) \in A]
\]

Shortest path is not expressibly without **set semantics**!
Shortest Path with Resource Constraints

Given graph $G = (V, A)$ and $k$ resource capacities $Q^{(k)}$

$$S_{s,t} \rightarrow P_{s,\forall \{s,t\},t,Q^{(k)}}$$

$$P_{s,x,t,q^{(k)}} \rightarrow a_{s,x}P_{x,x-x,t,q^{(k)}-r^{(k)}_{s,x}} \quad \forall x \in X, [(s, x) \in A] \forall k : q^{(k)} - r^{(k)}_{s,x} \geq 0$$

$$a_{s,t} \quad [(s, t) \in A] \forall k : q^{(k)} - r^{(k)}_{s,t} \geq 0$$
Traveling Salesman Problem

Given a graph $G = (V, A)$ visit all vertices in $V$ exactly once

**Formalization of the Bellman-Held-Karp algorithm**

$$S \rightarrow a_{1, i} P_i, \forall \{1, i, j\} \rightarrow a_{j, 1} \quad \forall i, j \in V, [1 \neq i \neq j][(1, i) \in A][(j, 1) \in A]$$

$$P_{i, x, j} \rightarrow a_{i, x} P_{x, x-x, j} \quad \forall x \in X, [(i, x) \in A]$$

$$[X = \emptyset][(i, j) \in A]$$
New Theoretical Insights

Considering the similarity of Shortest Path and TSP models ... Why is one significantly harder than the other?
New Theoretical Insights

Considering the similarity of Shortest Path and TSP models ...

Why is one significantly harder than the other?

In some cases ... 

- ... table indices can be **relaxed** ... *not symbol indices!* 
  → multiple indexed symbols map to the same table cell

- ... indices can be stored in an **amalgamated form**

- symbols with a higher **degree of freedom** are computed 
  → then update amalgamated index
Considering the similarity of Shortest Path and TSP models ...

Why is one significantly harder than the other?

In some cases ...

- ... table indices can be relaxed ... not symbol indices!
  → multiple indexed symbols map to the same table cell

- ... indices can be stored in an amalgamated form

- symbols with a higher degree of freedom are computed
  → then update amalgamated index

Preconditions are already formalized
New Theoretical Insights

Considering the similarity of Shortest Path and TSP models …

Why is one significantly harder than the other?

Two possibilities for **Shortest Path** …

- **Amalgamated set index**: less visited nodes $\implies$ higher degree of freedom

- **Completely relaxed set index**: requires heuristic search order $\rightarrow$ Djikstra’s algorithm
New Theoretical Insights

Considering the similarity of Shortest Path and TSP models ...

Why is one significantly harder than the other?

Two possibilities for Shortest Path ...

- **Amalgamated set index**: less visited nodes $\implies$ higher degree of freedom

- **Completely relaxed set index**: requires heuristic search order $\rightarrow$ Dijkstra’s algorithm

Not applicable for the TSP!
Conclusion
Whistle—ADP for combinatorial optimization

...no need to implement all this by yourself!

- Tailored for **combinatorial optimization** in general
- Written in **Rust** as compiler plugin—C ABI compatible
- Supports **integer, float, and set indices**
- Uses a new **compatibility and dominance** mechanism instead of objective functions
- Supports **Index Propagators** for advanced index deduction: Sequence data, Cyclic permutations, Resources, ...
- Different **evaluation algorithms**:
  Top-down, Bottom-up, Bidirectional (new)
- Supports **different search engines**:
  DFS (current), Greedy, A*, Beam-Search
- Supports **Partial Invalidation and Shadowing**
Thank you for your attention!