### A Theory and Algorithms for Combinatorial Reoptimization

Gal Tamir. Joint work with Hadas Shachnai and Tami Tamir

#### Combinatorial Reoptimization

Many real-life applications involve systems that change dynamically over time. Thus, throughout the continuous operation of such system, it is required to compute solutions for new problem instances, derived from previous instances. Since the transition from one solution to another incurs some cost, a natural goal is to solve for the new instance close to the original one.

Let \( \Pi \) denote an optimization problem. In the reoptimization problem \( R(\Pi) \) we want to optimally re-solve \( \Pi \) while minimizing the transition cost from an initial solution to a new one.

Solving \( R(\Pi) \) involves two challenges:
- Computing an optimal solution for the new instance.
- Efficiently converting the current solution to the new one.

#### MST - Reoptimization

<table>
<thead>
<tr>
<th>Old instance</th>
<th>New instance</th>
</tr>
</thead>
</table>

#### Reoptimization Algorithms

When \( \Pi \) is NP-hard, or when \( R(\Pi) \) is NP-hard, we consider approximate solutions.

Let \( O_\Pi(1) \) be the optimal value of \( \Pi(1) \) and \( O_{R(\Pi)}(1) \) be the optimal transition cost.

**Issue:** The transition cost depends on the quality of our approximation for \( \Pi \). Thus, for every \( \rho \geq 1 \), \( O_{R(\Pi)}(\rho) \) is the minimal possible transition cost to a configuration that yields a \( \rho \)-approximation for \( O_\Pi(1) \). Note that \( O_{R(\Pi)}(\rho) \) is a decreasing function of \( \rho \).

#### Strong Definition

An algorithm \( A \) yields an \((r,\rho)\)-reapproximation for \( R(\Pi) \), for \( r, \rho \geq 1 \), if, for any reoptimization input \( I_R \), \( A \) achieves a \( \rho \)-approximation for \( O_{R(I)}(\rho) \) with transition cost at most \( r \cdot O_{R(I)}(1) \).

**This definition is unusable and Hard to Use**

For some important classes of problems, it can be shown that if there exists an \((r,\rho)\)-reapproximation algorithm \( w.r.t. \) the strong definition then \( \Pi \) \( \in \) NP.

#### Weak Definition

An algorithm \( A \) yields an \((r,\rho)\)-reapproximation for \( R(\Pi) \), for \( r, \rho \geq 1 \), if, for any reoptimization input \( I_R \), \( A \) achieves a \( \rho \)-approximation for \( O_{R(I)}(\rho) \) with transition cost at most \( r \cdot O_{R(I)}(1) \).

In other words, compare the approximate solution to a \((1,1)\)-reapproximation for the input.

We can build an instance \( I'_R \). If the optimization value is larger than \( r \) the reoptimization-cost is 0; else this cost is strictly positive.

#### Applications

Reoptimization problems arise naturally in many real-life scenarios: storage systems for VoD services, communication services and other network problems, stock trading, production planning and vehicle routing.

**Storage Systems in Video on Demand Services**

**Online Service**

**Crew Scheduling**

An optimal solution for \( \Pi(I) \) with weights \( w_i \) is an optimal solution for \( \Pi(I) \) with weights \( w_i' \), and a minimal transition cost, i.e., it is a \((1,1)\)-reoptimization for \( R(\Pi) \).

#### Reoptimization of k-center selection

- The k-center problem admits a \((1,4)\)-reapproximation algorithm.

#### Reoptimization Algorithm for Subset-Selection

A \((1,1)\)-reoptimization algorithm for \( R(\Pi) \):

1. Let \( \Delta = \max_{i \in \Pi} \{ \delta_i + \delta_i' \} \).
2. Let \( \lambda = \Delta + 1 \).
3. Define for every \( i \in I \) a new weight, \( \delta_i' \), as follows: for every \( i \in I \), let \( \delta_i' = \lambda \delta_i + \delta_i' \).
4. Solve \( R(I) \) with weights \( \delta ' \).

**Our result**

- For all \( \Pi \in DP-B \) with bounded transition cost, \( R(\Pi) \) has a \((1,1+\varepsilon)\)-reapproximation algorithm.
- For all \( \Pi \in DP-B \), \( R(\Pi) \) has a \((1+\varepsilon_1,1+\varepsilon_2)\)-reapproximation algorithm.

For any maximization problem \( \Pi \) and \( \varepsilon \geq 0 \),

\[
\ell^* = \ell(R(\Pi), I_R, \varepsilon)
\]

\[
\equiv \min \{ m \mid \exists (R(\Pi), I_R, m, \varepsilon) \geq (1-\varepsilon)T(\Pi, I) \}
\]

1. Given an input \( I_R \) for \( R(\Pi) \) and \( \varepsilon_1, \varepsilon_2 > 0 \), compute the rounded instance \( I_{R, \varepsilon_2, \varepsilon_1} \).
2. Let \( \varepsilon_3 = \varepsilon_2 / 2 \).
3. Find \( \ell^* = \ell(R(\Pi), I_R, \varepsilon_1, \varepsilon_2) \).
4. Return a state in \( DP^* \) corresponding to a solution for \( R(\Pi) \) whose value is \( T(R(\Pi, \ell^*), I_R, \varepsilon_1, \varepsilon_2) \).

Develop a more complex DP and search for a most profitable solution with minimum reoptimization cost.

Run a generic algorithm for k-center selection and choose the centers by their reoptimization costs.