A Theory and Algorithms for Combinatorial Reoptimization[⋆]

Hadas Shachnai[1][⋆⋆], Gal Tamir[1], and Tami Tamir[2]

[1] Computer Science Department, Technion, Haifa 32000, Israel.
E-mail: {hadas,galtamir}@cs.technion.ac.il.
[2] School of Computer science, The Interdisciplinary Center, Herzliya, Israel.
E-mail: tami@idc.ac.il

Abstract. Many real-life applications involve systems that change dynamically over time. Thus, throughout the continuous operation of such a 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 have the solution for the new instance close to the original one (under a certain distance measure). In this paper we develop a general model for combinatorial reoptimization, encompassing classical objective functions as well as the goal of minimizing the transition cost from one solution to the other. Formally, we say that $\mathcal{A}$ is an (r, ρ)-reapproximation algorithm if it achieves a ρ-approximation for the optimization problem, while paying a transition cost that is at most r times the minimum required for solving the problem optimally. When $\rho = 1$ we get an $(r, 1)$-reoptimization algorithm.
Using our model we derive reoptimization and reapproximation algorithms for several important classes of optimization problems. This includes *fully polynomial time reapproximation schemes* for DP-benevolent problems, a class introduced by Woeginger (*Proc. Tenth ACM-SIAM Symposium on Discrete Algorithms, 1999*), reapproximation algorithms for metric Facility Location problems, and $(1, 1)$-reoptimization algorithm for polynomially solvable subset-selection problems.
Thus, we distinguish here for the first time between classes of reoptimization problems, by their hardness status with respect to minimizing transition costs while guaranteeing a good approximation for the underlying optimization problem.

1 Introduction

Traditional combinatorial optimization problems require finding solutions for a single instance. However, many of the real-life scenarios motivating these problems involve systems that change dynamically over time. Thus, throughout the

⋆ This research is supported by the Israel Science Foundation (grant number 1574/10), and by the Ministry of Trade and Industry MAGNET program through the NEGEV Consortium (www.negev-initiative.org).

⋆⋆ Work partially supported by the Technion V.P.R. Fund, and by the Smoler Research Fund.

continuous operation of such a system, it is required to compute solutions for new problem instances, derived from previous instances. Moreover, since there is some cost associated with the transition from one solution to another, the solution for the new instance must be *close* to the former solution (under certain distance measure).

For example, in a video-on-demand (VoD) system, movie popularities tend to change frequently. In order to satisfy new client requests, the content of the storage system needs to be modified. The new storage allocation needs to satisfy the current demand; also, due to the cost of file migrations, this should be achieved by using a minimum number of reassignments of file copies to servers [20]. In communication networks, the set of demands to connect sources to destinations changes over time. Rerouting incurs the cost of acquiring additional bandwidth for some links that were not used in the previous routing. The goal is to optimally handle new demands while minimizing the total cost incurred due to these routing changes.

Solving the above *reoptimization* problems involves two challenges: (*i*) *Computing* an optimal (or close to the optimal) solution for the new instance, and (*ii*) Efficiently *converting* the current solution to the new one.

In this paper we develop a general model for combinatorial reoptimization, encompassing objective functions that combine these two challenges. Our study differs from previous work in two aspects. One aspect is the generality of our approach. To the best of our knowledge, previous studies consider specific reoptimization problems. Consequently, known algorithms rely on techniques tailored for these problems (see Section 1.1). We are not aware of general theoretical results, or algorithmic techniques developed for certain *classes* of combinatorial reoptimization problems. This is the focus of our work. The other aspect is our performance measure, which combines two objective functions.[3] The vast majority of previous research refers to the computational complexity of solving an optimization problem once an initial input has been modified, i.e., the first of the above-mentioned challenges (see, e.g., the results for reoptimization of the *traveling salesman problem (TSP)* [4, 6]).

One consequence of these differences between our study and previous work is in the spirit of our results. Indeed, in solving a reoptimization problem, we usually expect that starting off with a solution for an initial instance of a problem should help us obtain a solution at least as good (in terms of approximation ratio) for a modified instance, with better running time. Yet, our results show that reoptimization with transition costs may be harder than solving the underlying optimization problem. This is inherent in the reoptimization problems motivating our study, rather than the model we use to tackle them. Indeed, due to the transition costs, we seek for the modified instance an efficient solution which can be reached at low cost. In that sense, the given initial solution plays a restrictive role, rather than serve as guidance to the algorithm.[4]

[3] As discussed in Section 1.1, this is different than multiobjective optimization.
[4] This is similar in nature, e.g., to *incremental optimization* studied in [16].

Applications: Reoptimization problems naturally arise in many real-life scenarios. Indeed, planned or unanticipated changes occur over time in almost any system. It is then required to respond to these changes quickly and efficiently. Ideally, the response should maintain high performance while affecting only a small portion of the system. In [21] we give a detailed description of some of the applications for which our reoptimization model fits well. This includes storage systems for VoD services, communication services and other network problems, stock trading, production planning and vehicle routing.

1.1 Related Work

The work on reoptimization problems started with the analysis of *dynamic graph* problems (see e.g. [12, 24] and a survey in [8]). These works focus on developing data structures supporting update and query operations on graphs. Reoptimization algorithms were developed also for some classic problems on graphs, such as shortest-path [18, 17] and minimum spanning tree [2]. Since all of these problems can be solved in polynomial time, even with no initial solution, the goal is to compute an optimal solution very efficiently, based on the local nature of the updates and on properties of optimal solutions.

A different line of research deals with the computation of a good solution for an NP-hard problem, given an optimal solution for a close instance. In general, NP-hardness of a problem implies that a solution for a locally modified instance cannot be found in polynomial time. However, it is an advantage to have a solution for a close instance, compared to not knowing it. In particular, for some problems it is possible to develop algorithms guaranteeing better approximation ratio for the reoptimization version than for the original problem. Among the problems studied in this setting are TSP, [4, 6], Steiner Tree on weighted graphs [11], Knapsack [1], and Pattern Matching problems [5]. A survey of other research in this direction is given in [3].

It is important to note that, unlike the present paper, in all of the above works, the goal is to compute an optimal (or approximate) solution for the modified instance. The resulting solution may be significantly different from the original one, since there is no cost associated with the transition among solutions. Reoptimization is also used as a technique in local-search algorithms. For example, in [25] reoptimization is used for efficient multiple sequence alignment – a fundamental problem in bioinformatics and computational biology. In [23], reoptimization in used to improve the performance of a branch-and-bound algorithm for the Knapsack problem.

Other related works consider *multiobjective* optimization problems. In these problems, there are several weight functions associated with the input elements. The goal is to find a solution whose quality is measured with respect to a combination of these weights (see e.g., [19, 14, 7]). Indeed, in alternative formulation of these problems, we can view one of the weight functions as the transition cost from one solution to another, thus, known results for multiobjective optimization carry over to *budgeted* reoptimization. However, in this paper we focus on minimizing the total transition cost required for achieving a good solution for the underlying optimization problem, rather than efficiently using a given budget. Indeed, in solving our reoptimization problems, it is natural to consider applying binary search, to find the reoptimization cost (i.e., the *budget*), and then use a multiobjective optimization algorithm as a black-box. However (as we show in Theorem 1), this cost cannot be found in polynomial time, unless $P = NP$. This leads us to use a different approach (and alternative measures) for obtaining reapproximation algorithms.

1.2 Our Contribution

We develop (in Section 2) a general model for combinatorial reoptimization that captures many real-life scenarios. Using our model, we derive reoptimization and reapproximation algorithms for several important classes of optimization problems. In particular, we consider (in Section 3) the class of DP-benevolent problems introduced by Woeginger [26]. The paper [26] gives an elaborate characterization of these problems, which is used to show that any problem in this class admits a *fully polynomial time approximation scheme (FPTAS)*.[5]

We introduce (in Definition 3) the notion of *fully polynomial time reapproximation scheme (FPTRS)*. Informally, such a scheme takes as input parameters $\varepsilon_1, \varepsilon_2 > 0$ and outputs a solution that approximates simultaneously the minimum reoptimization cost (within factor $1+\varepsilon_1$) and the objective function for Π (within factor $1+\varepsilon_2$), in time that is polynomial in the input size and in $1/\varepsilon_1$, $1/\varepsilon_2$. We show that the reoptimization variants of a non-trivial subclass of DP-benevolent problems admit fully polynomial time $(1+\varepsilon_1, 1+\varepsilon_2)$-reapproximation schemes, for any $\varepsilon_1, \varepsilon_2 > 0$. We note that this is the best possible, unless $P = NP$.

In Section 4 we show how α-approximation algorithms for metric Facility Location problems can be used to obtain $(1, 3\alpha)$-reapproximation algorithms for their reoptimization variants. In Section 5, we show that for any subset-selection problem Π over n elements, which can be optimally solved in time $T(n)$, there is a $(1, 1)$-reoptimization algorithm for the reoptimization version of Π, whose running time is $T(n')$, where n' is the size of the modified input. This yields a polynomial time $(1, 1)$-reoptimization algorithm for a large set of polynomially solvable problems, as well as for problems that are fixed parameter tractable.[6]

Thus, we distinguish here for the first time between classes of reoptimization problems by their hardness status with respect to the objective of minimizing transition costs, while guaranteeing a good approximation for the underlying optimization problem.

We conclude (in Section 6) with a discussion of possible directions for future work. Due to space constraints, the proofs and implementation details are omitted. The detailed results appear in [21].

[5] A key property is that each problem in the class can be formulated via a dynamic program of certain structure, and the involved costs and transition functions satisfy certain arithmetic and structural conditions.

[6] For the recent theory of fixed-parameter algorithms and parameterized complexity, see, e.g., [9].

2 Combinatorial Reoptimization: Definitions and Notations

In the following we formally define our model for combinatorial reoptimization. Given an optimization problem Π , let I_0 be an input for Π, and let $\mathcal{C}_{\mathcal{I}0} = \{C^1_{I_0}, C^2_{I_0}, \ldots\}$ be the set of configurations corresponding to the solution space of Π for I_0.[7] Each configuration $C^j_{I_0} \in \mathcal{C}_{\mathcal{I}0}$ has some value $val(C^j_{I_0})$. In the reoptimization problem, $R(\Pi)$, we are given a configuration $C^j_{I_0} \in \mathcal{C}_{\mathcal{I}0}$ of an initial instance I_0, and a new instance I derived from I_0 by admissible operation, e.g, addition or removal of elements, changes in element parameters etc. For any element $i \in I$ and configuration $C^k_I \in \mathcal{C}_{\mathcal{I}}$, we are given the *transition cost* of i when moving from the initial configuration $C^j_{I_0}$ to the feasible configuration C^k_I of the new instance. We denote this transition cost by $\delta(i, C^j_{I_0}, C^k_I)$. Practically, the transition cost of i is not given as a function of two configurations, but as a function of i's state in the initial configuration and its possible states in any new configuration. This representation keeps the input description more compact. The primary goal is to find an optimal solution for I. Among all configurations with an optimal $val(C^k_I)$ value, we seek a configuration C^*_I for which the total transition cost, given by $\sum_{i\in I} \delta(i, C^j_{I_0}, C^*_I)$ is minimized.

For example, assume that Π is the *minimum spanning tree (MST)* problem. Let $G_0 = (V_0, E_0)$ be a weighted graph, and let $T_0 = (V_0, E_{T_0})$ be an MST for G_0. Let $G = (V, E)$ be a graph derived from G_0 by adding or removing vertices and/or edges, and by changing the weights of edges. Let $T = (V, E_T)$ be an MST for G. For every edge $e \in E_T \setminus E_{T_0}$, we are given the cost $\delta_{add}(e)$ of adding e to the new solution, and for every edge $e \in E \cap (E_{T_0} \setminus E_T)$ we are given the cost $\delta_{rem}(e)$ of removing e from the solution. The goal in the reoptimization problem $R(MST)$ is to find an MST of G with minimal total transition cost. As we show in Section 5, $R(MST)$ belongs to a class of subset-selection problems that are polynomially solvable.

The input for the reoptimization problem, I_R, contains both the new instance, I, and the transition costs δ (that may be encoded in different ways). Note that I_R does not include the initial configuration I_0 since, apart from determining the transition costs, it has no effect on the reoptimization problem.

2.1 Approximate Reoptimization

When the problem Π is NP-hard, or when the reoptimization problem $R(\Pi)$ is NP-hard,[8] we consider approximate solutions. The goal is to find a good solution for the new instance, while keeping a low transition cost from the initial configuration to the new one. Formally, denote by $\mathcal{O}(I)$ the optimal value of $\Pi(I)$ (i.e., the instance I of Π). A configuration $C^k_I \in \mathcal{C}_{\mathcal{I}}$ yields a ρ-approximation for $\Pi(I)$, for $\rho \geq 1$, if its value is within ratio ρ from $\mathcal{O}(I)$. That is, if Π is a minimization problem then $val(C^k_I) \leq \rho\mathcal{O}(I)$; if Π is a maximization problem

[7] A configuration can be any representation of a (partial) solution for Π.

[8] As we show below, it may be that none, both, or only $R(\Pi)$ is NP-hard.

then $val(C_I^k) \geq (1/\rho)\mathcal{O}(I)$. Given a reoptimization instance I_R, for any $\rho \geq 1$, denote by $\mathcal{O}_R(I_R, \rho)$ the minimal possible transition cost to a configuration $C_I^k \in \mathcal{C}_\mathcal{I}$ that yields a ρ-approximation for $\mathcal{O}(I)$, and by $\mathcal{O}_R(I_R)$ the minimal transition cost to an optimal configuration of I.

Ideally, in solving a reoptimization problem, we would like to find a solution whose total transition cost is close to the best possible, among all solutions with a given approximation guarantee, $\rho \geq 1$, for the underlying optimization problem. Formally,

Definition 1. *An algorithm $\mathcal{A}$ yields a* strong (r, ρ)-reapproximation *for $R(\Pi)$, for $\rho, r \geq 1$, if, for any reoptimization input I_R, $\mathcal{A}$ achieves a ρ-approximation for $\mathcal{O}(I)$, with transition cost at most $r \cdot \mathcal{O}_R(I_R, \rho)$.*

Unfortunately, for many NP-hard optimization problems, finding a strong (r, ρ)-reapproximation is NP-hard, for any $r, \rho \geq 1$. This follows from the fact that it is NP-hard to determine whether the initial configuration is a ρ-approximation for the optimal one (in which case, the transition cost to a ρ-approximate solution is equal to zero). We demonstrate this hardness for the Knapsack problem.

Theorem 1. *For any $r, \rho \geq 1$, obtaining a strong (r, ρ)-reapproximation for Knapsack is NP-hard.*

Thus, for such problems, we use an alternative measure, which compares the total transition cost of the algorithm to the best possible, when the underlying optimization problem is solved optimally. This alternative measure in fact helps us achieve our preliminary goal, namely, finding a good approximation for the optimization problem; to that end, we compare the incurred reoptimization cost with a higher optimum. Formally,

Definition 2. *An algorithm $\mathcal{A}$ yields an (r, ρ)-reapproximation for $R(\Pi)$, for $\rho, r \geq 1$, if, for any reoptimization input I_R, $\mathcal{A}$ achieves a ρ-approximation for $\mathcal{O}(I)$, with transition cost at most $r \cdot \mathcal{O}_R(I_R)$.*

Clearly, any strong (r, ρ)-reapproximation is also an (r, ρ)-reapproximation. For $\rho = 1$, we say that an $(r, 1)$-reapproximation algorithm is also an $(r, 1)$-reoptimization algorithm (as it yields an optimal solution). In this case, Definitions 1 and 2 coincide.

Our study encompasses a non-trivial subclass of optimization problems that admit FPTAS. Approximating the reoptimization versions of these problems involves two error parameters, $\varepsilon_1, \varepsilon_2$. This leads to the following extension for the classic definition of FPTAS.

Definition 3. *A fully polynomial time reapproximation scheme (FPTRS) for $R(\Pi)$ is an algorithm that gets an input for $R(\Pi)$ and the parameters $\varepsilon_1, \varepsilon_2 > 0$ and yields a $(1 + \varepsilon_1, 1 + \varepsilon_2)$-reapproximation for $R(\Pi)$, in time polynomial in $|I_R|, 1/\varepsilon_1$ and $1/\varepsilon_2$.*

Budgeted Reoptimization: The budgeted reoptimization problem $R(\Pi, m)$ is a restricted version of $R(\Pi)$, in which we add the constraint that the transition cost is at most m, for some budget $m \geq 0$. Its optimal solution for the input I_R is denoted $\mathcal{O}(I_R, m)$. Note that $\mathcal{O}(I_R, m)$ is the value of the best configuration that can be produced from the initial configuration with transition cost at most m.

Definition 4. *An algorithm $\mathcal{A}$ yields a ρ-approximation for $R(\Pi, m)$ if, for any reoptimization input I_R, $\mathcal{A}$ yields a ρ-approximation for $\mathcal{O}(I_R, m)$, with transition cost at most m.*

Note that the optimal value of $\mathcal{O}(I_R, m)$ may be far from $\mathcal{O}(I)$; thus, it is reasonable to evaluate algorithms for $R(\Pi, m)$ by comparison to $\mathcal{O}(I_R, m)$ and not to $\mathcal{O}(I)$.

3 Reoptimization of DP-benevolent Problems

In this section we consider the class of DP-benevolent problems introduced in [26]. For short, we call the class *DP-B*. The input for any problem Π in this class consists of a set of vectors $\bar{X}_i \in \mathbb{N}^\alpha$, $1 \leq i \leq n$, where $\alpha \geq 1$ is a fixed constant. Each problem $\Pi \in$ *DP-B* can be solved by using a dynamic program that is characterized by a set of states and a finite set, $\mathcal{F}$, of mappings; each such mapping determines the new state after a transition, which occurs during the execution of the dynamic program for Π. The paper [26] defines also the class of DP-simple problems. Such problems can be expressed via a simple dynamic program (which satisfies certain structural properties). The class of *DP-B* problems contains a large set of problems that admit an *FPTAS* via dynamic programming. We show that a non-trivial subclass of *DP-B* problems admit FPTRS. We review the description of *DP-B* problems and give the detailed proofs of our results in [21].

3.1 Polynomially Bounded Transition Costs

We first consider instances in which the transition costs are polynomially bounded in the input size. Let $\mathcal{F}$ be the set of mappings among states corresponding to (partial) solutions for a problem $\Pi \in$ DP-B. Our first main result is the following.

Theorem 2. *Let $R(\Pi)$ be the reoptimization version of a problem Π ∈DP-B, for which $|\mathcal{F}|$ is fixed, then there exists a fully polynomial time $(1, 1 + \varepsilon)$-reapproximation scheme for $R(\Pi)$.*

Recall that $R(\Pi, m)$ is a restricted version of $R(\Pi)$, in which the total transition cost is at most m, for some integer $m \geq 0$, and $\mathcal{O}(I_R, m)$ is the optimal value of $R(\Pi, m)$ for the input I_R. We show that $R(\Pi, m)$ is *DP*-benevolent in two steps. First, we show that $R(\Pi, m)$ is *DP*-simple, i.e., it can be expressed via a simple dynamic programming formulation (see in [21]). Next, we show that $R(\Pi, m)$ satisfies the properties of the class DP-B.

Theorem 3. *For any $\Pi \in$ DP-B, for which $|\mathcal{F}|$ is fixed, and any $m \in \mathbb{N}$, $R(\Pi, m) \in$ DP-B.*

Intuitively, budgeted Π is also in DP-B, since the budget induces a new 'knapsack-like dimension' on Π.

3.2 Arbitrary Transition Costs

Let I_R be an input for $R(\Pi, m)$, for some integer $m \geq 0$. Given the set of vectors $\bar{X} \in I$ containing the parameters of the input elements, we denote by $\bar{Y} = (\bar{X}; \bar{r})$ the vector corresponding to each vector $\bar{X}$ in the reoptimization instance I_R, where $\bar{r} = (r_{F_1}, r_{F_2}, \ldots)$ is the transition cost vector associated with $\bar{X}$ in I_R. To obtain approximate solutions for instances with arbitrary transition costs, we first apply a transformation on the cost vector $\bar{r}$.

Definition 5. *Given an input I_R for $R(\Pi)$, let γ be a rounding function that accepts as input the cost vector $\bar{r}$ and the parameters $n, m \in \mathbb{N}$ and $\varepsilon \in (0,1)$, then*

$$\gamma(\bar{r}, n, m, \varepsilon) = (\left\lfloor \frac{r_{F_1} \cdot n}{m \cdot \varepsilon} \right\rfloor, \left\lfloor \frac{r_{F_2} \cdot n}{m \cdot \varepsilon} \right\rfloor, \ldots).$$

Now, given an input I_R for $R(\Pi, m)$, we modify each element $\bar{Y} = (\bar{X}; \bar{r}) \in I_R$ to $\bar{Y}' = (\bar{X}; \gamma(\bar{r}, n, m, \varepsilon))$. Denote the rounded instance by $\hat{I}_{R,\varepsilon}$. Since the transition costs are rounded down, it holds that $\mathcal{O}(I_R, m) \leq \mathcal{O}(\hat{I}_{R,\varepsilon}, m)$.

Let DP, DP' be the dynamic programs for Π, $R(\Pi, m)$ $\in$DP-B, respectively. Let $\varepsilon_1, \varepsilon_2 > 0$ be two error parameters. Given the rounded values $\gamma(\bar{r}, n, m, \varepsilon_1/2)$, we can use binary search to find the minimum budget m such that the value obtained by DP' for the input I_R of $R(\Pi, m)$ is within factor $1 - \varepsilon_2$ from the best objective value obtained by DP for the input I of Π. The above is the main idea used to obtain our next result.

Theorem 4. *Let $R(\Pi)$ be the reoptimization version of a problem Π $\in$DP-B, for which $|\mathcal{F}|$ is fixed, then for any transition costs, there exists a fully polynomial time $(1 + \varepsilon_1, 1 + \varepsilon_2)$-reapproximation scheme for $R(\Pi)$.*

We note that the result in Theorem 4 is the best possible, unless $P = NP$. Indeed, there exist optimization problems Π that can be reduced to their reoptimization version, $R(\Pi)$. This includes, e.g., the subclass of minimization subset selection problems, in which we can use the costs in a given instance I as transition costs and assign to all elements initial cost 0. Thus, solving Π for I is equivalent to solving $R(\Pi)$ for I_R.

4 Reoptimization of Metric Facility Location

In this section we show how approximation algorithms for classic network design problems can be used to obtain reapproximation algorithms with similar performance ratios and the minimum reoptimization cost. We exemplify this on the Center Selection problem. The input is a set of n sites $s_1, \ldots, s_n$ in a metric space. The goal is to select the locations of k centers (on the plane) so that the maximum distance from a site to its nearest center is minimized. Let $\Pi(I_0)$ be the Center Selection problem over an instance I_0. In the reoptimization problem, the instance I_0 is modified. The change can involve insertion or deletion of sites, as well as changes in the distance function. Denote by I the modified instance. Given an approximate solution S_0 for $\Pi(I_0)$, the goal in the reoptimization problem is to find an approximate solution S for $\Pi(I)$, such that S has the minimal possible transition cost from S_0. Specifically, the opening cost of any center $i \in S_0$ is 0, while there is a uniform (positive) cost associated with opening a center at any other location. W.l.o.g, we assume a unit opening cost for any new center. Denote by $c(\ell) \geq 0$ the cost of opening a center in location ℓ; when ℓ is the location of a site, s_j, we may use the notation $c(s_j)$. The transition cost from S_0 to S is the sum of the costs of the new centers, i.e., $\sum_{\ell \in S \setminus S_0} c(\ell)$, where ℓ is a location in which we open a center. Suppose that for some $\alpha \geq 1$, we have an α-approximation algorithm for the Center Selection problem, denoted by $\mathcal{A}_{CS}$. We give below a reapproximation algorithm for $R(Center\_Selection)$. We measure the approximation ratio of our algorithm using Definition 2, as it is NP-hard to obtain any *strong* reapproximation algorithm.

Let $dist(m, \ell)$ denote the distance from location m to location ℓ (where *location* may also be a site). We say that a site s_j is covered if there is a center that is 'close enough' to s_j (see below). Initially, all sites are *uncovered.*

```
Algorithm $\tilde{\mathcal{A}}_{CS}$ for $R(Center\_Selection)$:

1. Preprocessing step: Use algorithm $\mathcal{A}_{CS}$ to obtain $\alpha$-approximation
   for the Center Selection problem with the input $I$. Let $\hat{d} = D(\mathcal{A}_{CS})$
   be the maximal distance from any site to the closest center output
   by $\mathcal{A}_{CS}$.
2. Let $S = \emptyset$ and $L = \{\ell \mid c(\ell) = 0\}$.
3. Let $U = \{s_1, \ldots, s_n\}$ be the set of uncovered sites.
4. While there exists ($\ell \in L$ and $s_j \in U$ with $dist(s_j \ell) \leq \hat{d}$) do
   (a) $S = S \cup \{\ell\}$.
   (b) For any site $s_j$ with $dist(j, \ell) \leq 3\hat{d}$ do $U = U \setminus \{s_j\}$.
5. $k' = 0$, and set $D(\mathcal{A}_{CS}) = \infty$.
   While $D(\mathcal{A}_{CS}) > \hat{d}$ do
   (a) $k' = k' + 1$
   (b) Run Algorithm $A_{CS}$ with the set of sites $U$ and parameter $k'$.
6. Let $S_{A_{CS}}$ be the set of centers opened by $A_{CS}$, then $S = S \cup S_{A_{CS}}$.
   Output $S$.
```

Theorem 5. *$\tilde{\mathcal{A}}_{CS}$ is a $(1, 3\alpha)$-reapproximation algorithm for $R(Center\_Selection)$.*

We note that for the case where $\alpha = 2$, we can obtain a better reapproximation ratio for the Center Selection problem. This can be done by modifying algorithm $\tilde{\mathcal{A}}_{CS}$ to use the generic 2-approximation algorithm for Center Selection (see in [15] and [10]). Thus, we have

Theorem 6. *There is a $(1, 4)$-reapproximation algorithm for $R(Center\_Selection)$.*

5 Optimal Reoptimization of Weighted Subset-Selection

In this section we show that for any subset-selection problem Π over n elements, that can be optimally solved in time $T(n)$, there is a $(1, 1)$-reoptimization algorithm for the reoptimization version of Π, whose running time is $T(n')$, where n' is the size of the modified input. In particular, if Π is solvable in polynomial time, then so is its reoptimization variant. This includes the minimum spanning tree problem, shortest path problems, maximum matching, maximum weighted independent set in interval graphs, and more. Similarly, if Π is fixed parameter tractable, then so is $R(\Pi)$. We describe the framework for maximization problems. With slight changes it fits also for minimization problems.

Let Π be a polynomially solvable subset-selection maximization problem over an instance I_0. The weight of an element $i \in I_0$ is given by an integer $w_i \geq 0$. The goal is to select a subset $S_0 \subseteq I_0$ satisfying various constraints, such that the total weight of the elements in S is maximized. In the reoptimization problem, the instance I_0 is modified. The change can involve insertion or deletion of elements, as well as changes in element weights. For example, in the maximum matching problem, possible changes are addition or deletion of vertices and edges, as well as changes in edge weights. Denote by I the modified instance. Let w'_i denote the modified weight of element i. For a given optimal solution S_0 of $\Pi(I_0)$, the goal in the reoptimization problem is to find an optimal solution S of $\Pi(I)$ with respect to the modified weights, such that S has the minimal possible transition cost from S_0. Specifically, every element $i \in I$, is associated with a removal-cost $\delta_{rem}(i)$ to be charged if $i \in S_0 \setminus S$, and an addition-cost $\delta_{add}(i)$ to be charged if $i \in S \setminus S_0$. The transition cost from S_0 to S is defined as the sum of the corresponding removal- and addition-costs. The following is a $(1, 1)$-reoptimization algorithm for $R(\Pi)$.

```
A (1,1)-reoptimization algorithm for R(Π):

  (i) Let $\Delta = \max(\max_{i \in S_0 \cap I} \delta_{rem}(i), \max_{i \in I \setminus S_0} \delta_{add}(i))$.
 (ii) Let $\lambda = 2|I|\Delta + 1$.
(iii) Define for every $i \in I$ a new weight, $\hat{w}_i$, as follows: for every $i \in S_0 \cap I$,
      let $\hat{w}_i = \lambda w'_i + \delta_{rem}(i)$. For every $i \in I \setminus S_0$, let $\hat{w}_i = \lambda w'_i - \delta_{add}(i)$.
 (iv) Solve $\Pi(I)$ with weights $\hat{w}$.
```

The proofs of the following theorems are given in the full version [21].

Theorem 7. *An optimal solution for $\Pi(I)$ with weights $\hat{w}$ is an optimal solution for $\Pi(I)$ with weights w', and a minimal transition cost, i.e., it is a $(1, 1)$-reoptimization for $R(\Pi)$.*

The above framework does not fit for problems in which the optimal algorithm is for the objective of finding a subset of minimum (maximum) *cardinality.* Moreover,

Theorem 8. *There are polynomially-solvable subset-selection problems whose reoptimization variants are NP-hard.*

6 Discussion

In this paper we developed a general model for combinatorial reoptimization. We defined the notion of *reapproximation* and showed that for many optimization problems, strong reapproximation algorithms are unlikely to exist, unless

P=NP. This led us to an alternative definition that is used to obtain reapproximation algorithms as well as FPTRS for a non-trivial subclass of DP-benevolent problems.

The theoretical model introduced in this paper serves as a first step in the study of the reoptimization variants of classical problems, which arise in many practical scenarios. Our results show how known techniques from combinatorial optimization can be enhanced to obtain efficient reapproximation algorithms (or reapproximation schemes). It is natural to try and develop a generic approach for obtaining reapproximation algorithms for a family of metric network design problems, based on known approximation algorithms for these problems. More generally, in the study of reoptimization variants of NP-hard problems, suppose that there exists an α-approximation algorithm for such optimization problem Π. Is there a polynomial time (r, α)-reapproximation algorithm for $R(\Pi)$, for some bounded value $r > 1$? We have shown that any (weighted) subset selection problem that is polynomially solvable admits a $(1, 1)$-reoptimization algorithm. The existence of such optimal algorithms for a wider class of problems remains open.

Finally, while our model captures well the transition from one solution to the other, namely, scenarios where an initial input I_0 changes to a new one, I, it is interesting to consider also scenarios in which a *sequence* of changes is applied to an initial input. Formally, in addition to the initial input I_0 and a solution S_0, the instance of the reoptimization problem consists also of a sequence $(I', I'', \ldots)$ of inputs. The goal is to find a solution for each of the inputs in the sequence optimizing the quality of the solutions and the total transition cost (with no transition costs such a problem is studied in [13]). An optimal solution for sequence reoptimization may be significantly different from the solution derived by combining the optimal transition for each pair of consecutive solutions in the sequence. It is natural to examine also the usage of the techniques developed for *incremental approximation* (see, e.g., [16]). Here, the algorithms gradually modify the solutions for a given sequence of inputs, while guaranteeing that, for any $i > 1$, the i-th solution contains the first $(i - 1)$ solutions.

Acknowledgments: We thank Baruch Schieber and Rohit Khandekar for helpful discussions.

References

1. C. Archetti, L. Bertazzi, M.G. Speranza, Reoptimizing the 0-1 knapsack problem, *Discrete applied mathematics*, vol. 158(17), 2010.
2. G. Amato, G. Cattaneo, G. F. Italiano. Experimental analysis of dynamic minimum spanning tree algorithms. *In Proc. of* 8th SODA, 1997.
3. G. Ausiello, V. Bonifaci, and B. Escoffier, Complexity and approximation in reoptimization. In *Computability in Context: Computation and Logic in the Real World*, B. Cooper, A. Sorbi Eds., Imperial College Press/World Scientific, 2011.
4. G. Ausiello, B. Escoffier, J. Monnot and V. Th. Paschos, Reoptimization of minimum and maximum traveling salesmans tours, *J. of Discrete Algorithms* 7(4):453-463, 2009.

5. D. Bilo, H. Böckenhauer, D. Komm, R. Královič, T. Mömke, S. Seibert, A. Zych. Reoptimization of the shortest common superstring problem. *Symp. on Combinatorial Pattern Matching*, (CPM) 2009.
6. H.J. Böckenhauer, L. Forlizzi, J. Hromkovič, J. Kneis, J. Kupke, G. Proietti, P. Widmayer: On the approximability of TSP on local modifications of optimally solved instances. *Algorithmic Operations Research* 2(2), 2007,
7. A. Berger, V. Bonifaci, F. Grandoni, G. Schäfer, Budgeted matching and budgeted matroid intersection via the gasoline puzzle. In proc. *IPCO 2008*, pp. 273–287, 2008.
8. C. Demetrescu, I. Finocchi, and G.F. Italiano. Dynamic graph algorithms. *Handbook of Graph Theory*, J. Yellen and J.L. Gross eds., CRC Press Series, in Discrete Math and Its Applications, 2003.
9. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York, 1999.
10. M. E. Dyer and A. M. Frieze. A simple heuristic for the p-center problem. Oper. Res. Lett., 3:285-288, 1985.
11. B. Escoffier, M. Milanič, V. Th. Paschos: Simple and fast reoptimizations for the Steiner tree problem. *DIMACS Technical Report* 2007-01.
12. D. Eppstein, Z. Galil and G.F. Italiano, Dynamic graph algorithms, Chapter 8. in *CRC Handbook of Algorithms and Theory of Computation*, ed. M. J. Atallah, 1999.
13. A. Frangioni and A. Manca. A Computational study of cost reoptimization for min-cost flow problems. *INFORMS Journal on Computing* vol. 18(1), 2006.
14. F. Grandoni, R. Zenklusen. Optimization with more than one budget. In *Proc. of ESA*, 2010.
15. D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. In *Math.of Operations Research*, 10:180-184, 1985.
16. G. Lin, C. Nagarajan, R. Rajaraman, and D. P. Williamson. A general approach for incremental approximation and hierarchical clustering. SIAM J. Comput. 39(8): 3633–3669, 2010.
17. E. Nardelli, G. Proietti, P. Widmayer: Swapping a failing edge of a single source shortest paths tree is good and fast. Algorithmica 35, 2003.
18. S. Pallottino and M. G. Scutella; A new algorithm for reoptimizing shortest paths when the arc costs change, *Operations Research Letters*, vol. 31, 2003.
19. R. Ravi, M. X. Goemans. The constrained minimum spanning tree problem, In *5th Workshop on Algorithm Theory*, 66–75, 1996.
20. H. Shachnai, G. Tamir and T. Tamir, Minimal cost reconfiguration of data placement in storage area network. , *In Proc. of* 7rd WAOA, 2009.
21. H. Shachnai, G. Tamir, and T. Tamir. A Theory and Algorithms for Combinatorial Reoptimization. full version. `http://www.cs.technion.ac.il/~hadas/PUB/reopt_full.pdf`.
22. M. Shindler, Approximation Algorithms for the Metric k-Median Problem. *Masters thesis*, Department of Computer Science, UCLA, 2008.
23. B. Thiongane, A. Nagih and G. Plateau, Lagrangian heuristics combined with reoptimization for the 0-1 bidimensional knapsack problem. *Discrete Appl. Math.*, vol. 154:15, 2006.
24. M. Thorup and D.R. Karger. Dynamic graph algorithms with applications. *In Proc of* 7th SWAT, 2000.
25. F. Yue, J. Tang, A new approach for tree alignment based on local re-optimization, In Proc. of *Intl Conf. on BioMedical Engineering and Informatics*, 2008.
26. G. J. Woeginger. When does a dynamic programming formulation Guarantee the Existence of an FPTAS? In Proc. of *SODA*, pp. 820–829, 1999.