Chapter 9

On-line Generalized Steiner Problem

Baruch Awerbuch[*] *Yossi Azar*[†] *Yair Bartal*[‡]

Abstract

The *Generalized Steiner Problem* (GSP) is defined as follows. We are given a graph with non-negative weights and a set of pairs of vertices. The algorithm has to construct minimum weight subgraph such that the two nodes of each pair are connected by a path.

Off-line generalized Steiner problem approximation algorithms were given in [AKR91, GW92].

We consider the on-line generalized Steiner problem, in which pairs of vertices arrive on-line and are needed to be connected immediately.

We give a simple $O(\log^2 n)$ competitive deterministic on-line algorithm. The previous best algorithm was $O(\sqrt{n}\log n)$ competitive [WY93].

We also consider the *network connectivity leasing* problem which is a generalization of the GSP. Here edges of the graph can be either bought or leased for different costs. We provide simple randomized $O(\log^2 n)$ competitive algorithm based on the on-line generalized Steiner problem result.

1 Introduction

1.1 On-line Generalized Steiner Problem

Off-line version of the problem. The *Generalized Steiner Problem* (GSP) is defined as follows. We are given a graph with non-negative weights and a set of pairs of vertices. The algorithm has to construct minimum weight subgraph such that the two nodes of each pair are connected by a path. This problem [AKR91, GW92] has recently received a lot of attention in combinatorial optimization, networking, and distributed computing communities.

Agrawal et al and Goemans et al [AKR91, GW92] have shown a polynomial-time $2(1-\frac{1}{n})$-approximation algorithm. However, these algorithms are inapplicable in either on-line or distributed environments.

The special case of the GSP problem where all pairs of some subset of vertices have to be connected is the *Steiner Tree* problem. It is one of the most notorious NP-hard problems [Kar72, Win92]. The problem has been studies in a series of papers including [IW91, CV93, AA93, ABF93, WY93].

On-line version of the problem. On-line Steiner tree problem comes up in the context of network synchronization [AP90], mobile users tracking [AP91], distributed paging and file allocation [BFR92, ABF93, WY93, LRWY94], etc.

On-line *Generalized* Steiner Problem (in contrast to on-line Steiner Tree) problem [WY93] captures more refined communication requirements, e.g., situations where only partial (rather than global) synchronization is necessary. As pointed out in [AKR91], the on-line generalized Steiner problem can be viewed as the problem of minimizing the cost of building a network satisfying certain connectivity requirements, where new such requirements appear over time. It also captures the aspect of *communication aggregation*, namely the fact that in many situations, the cost of communication protocol is measured by the num-

[*] Johns Hopkins University and Lab. for Computer Science, MIT. Supported by Air Force Contract TNDGAFOSR-86-0078, ARO contract DAAL03-86-K-0171, NSF contract 9114440-CCR, DARPA contract N00014-J-92-1799, and a special grant from IBM. E-Mail: baruch@theory.lcs.mit.edu.

[†] Department of Computer Science, Tel Aviv University, Tel-Aviv 69978, Israel. Research supported in part by Allon Fellowship and by the Israel Science Foundation administered by the Israel Academy of Sciences. E-Mail: azar@math.tau.ac.il.

[‡] Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Research supported in part by Ben Gurion Fellowship, the Ministry of Science and Arts. E-Mail: yairb@math.tau.ac.il.

ber of edges used, rather than by the number of bits sent, which is certainly the case with long-term trunk reservation of telephone network. More formally, the problem can be defined as follows

Input: We consider an undirected weighted graph $G(V, E, w)$ with $|V| = n$ vertices and a *weight* function $w : E \rightarrow \mathcal{R}^+$, assigning an arbitrary non-negative weight $w(e)$ to each edge $e \in E$. Pairs of vertices of G, $p = \{q, \bar{q}\}$ appear on-line.

Output: The algorithm has to construct subgraph H such that for each pair q is connected to $\bar{q}$ (i.e., there is a path between $q, \bar{q}$). The goal is to construct H of minimum weight.

It is easy to see that H ought to be a forest of trees.

We comment that on-line GSP problem is also equivalent to the following problem. Pairs (j, v) arrives on-line where j is index of a set and v is a vertex in G. The algorithm has to add v to the j'th group, so that all the vertices which belong to group j are connected.

Our result versus previous work. Westbrook and Yan [WY93] gave an algorithm for on-line GSP that achieves $O(\sqrt{n} \log n)$ competitive ratio.

In contrast, we show

Theorem 1.1. *The Min-Cost algorithm for the on-line generalized Steiner problem is $O(\log^2 n)$ competitive.*

An $\Omega(\log n)$ lower bound on the competitiveness of any on-line alorithm for the generalized Steiner problem follows from the lower bound on-line Steiner tree, shown by Imaze and Waxman [IW91].

1.2 Network Connectivity Leasing Problem

The GSP problem can be generalized to *Network Connectivity Leasing* problem below. Imagine we can either *buy* or *lease* network edges. The cost of purchasing an edge is F times more expensive than the cost of renting that edge. Once an edge is bought, it can be used for free to accommodate future requests. The special case of Connectivity Leasing problem, in which all edges bought must form a tree, is called *Tree Leasing Problem*. We comment this special case is essentially the file replication problem [BS89, BFR92, ABF93, WY93, Kog93, AK94, LRWY94].

Picking $F = 1$, the problem reduces to the generalized Steiner problem. Note that for a single link network, this is exactly the *ski rental* problem (due to L. Rudolph, see [Kar92]).

Our result versus previous work. Optimally-competitive algorithms are known for both ski rental problem [Kar92] and tree leasing problem [BFR92, ABF93]. This work does not apply to the general case of network connectivity leasing.

We give a randomized connectivity leasing algorithm that follows as application of Theorem 1.1, using a general technique ([Bar94]) that applies to a large set of on-line problems.

Theorem 1.2. *There exists an $O(\log^2 n)$ competitive randomized algorithm for the network connectivity leasing problem.*

2 On-line Generalized Steiner Problem Algorithm

The Minimum Cost GSP Algorithm: For two vertices u, v in a graph G let $\mathrm{dist}_G(u, v)$ denote the (weighted) length of a shortest path in G between those vertices, i.e., the cost of the cheapest path connecting them, where the cost of a path $(e_1, \ldots, e_s)$ is $\sum_{1 \le i \le s} w(e_i)$. (We sometimes omit the subscript G where no confusion arises.)

Given an algorithm for the generalized Steiner problem, at every stage we can associate a graph $\hat{G}(V, E, \hat{w})$ such that $\hat{w}$ differs from w in that every edge, e of the subgraph H constructed by the algorithm at that stage has new weight $\hat{w}(e) = 0$.

Min-Cost GSP Algorithm: For request $p = \{q, \bar{q}\}$ connect q to $\bar{q}$ thru the current minimum cost path in the graph $\hat{G}$.

Theorem 2.1. *The Min-Cost GSP algorithm is $O(\log^2 n)$ competitive.*

Proof. We denote by $\mathrm{Cost}(p)$ the on-line cost expended for adding a pair p. Clearly the off-line optimum solution consists of set of connected components. Each requested pair must be in the same component. Let C be some connected component and $weight(C)$ be the total weight of edges of C. Let $P(C)$ be the set of pairs of vertices that belong to C.

Let $P_\ell(C) = \{p \in P(C) | \text{Cost}(p) \geq \ell\}$. Our proof is based on the following lemma:

LEMMA 2.1. *Using the above notation for a given component C and for every $\ell > 0$,*

$$|P_\ell(C)| = O(\frac{weight(C) \cdot \log n}{\ell})$$

.

To complete the proof of Theorem 2.1 we sort the costs $\text{Cost}(p)$ of all pairs in $p \in P(C)$ in non-increasing order. The above lemma 2.1 implies that the i'th cost in the order is $O(weight(C) \log n/i)$. Hence, the cost that the on-line encountered for the set C is bounded as follows:

$$\begin{aligned} \sum_{p \in P(C)} \text{Cost}(p) &\leq \sum_{1 \leq i \leq n} O(\frac{weight(C)}{i} \log n) \\ &= O\quad (weight(C) \cdot \log^2 n) \end{aligned}$$

Summing up the above equation over all clusters implies the theorem.

To complete the proof, we need to prove Lemma 2.1.

Proof. (of lemma 2.1): Given a weighted graph $G(V, E, w)$ and a subset of vertices S, a parameter $d \geq 0$, and a subset $Q \subset S \subset V$, we say that Q is a d-maximal-independent subset of S if the following conditions hold:

- there exists a mapping $\text{Dom}_d : S \rightarrow Q$, so that, for all $v \in S$, $\text{dist}_G(v, \text{Dom}_d(v)) \leq d$.
- for all distinct members of $u, v \in Q, u \neq v$, $\text{dist}(v, u) > d$.

In other words, d-maximal-independent subset is simply a maximal independent subset of S in a "power d" graph where edges represent paths of length less or equal than d between vertices in S.

Observe that a d-maximal-independent subset can be constructed greedily, starting with empty set Q and repeatedly adding to it yet (d-) undominated vertices from S, (i.e., vertices $u \in S$ for which no node $v \in Q$ such that $\text{dist}(u, v) \leq d$ exists) until no more such vertices exist.

Let V_C be the set of vertices of C, and let V_d be a d-maximal-independent subset of the set V_C. Now, for a specific $\ell > 0$, We define the set of edges $E_{\ell,d}$. and for each pair $p = \{u, v\} \in P_\ell(C)$, such that u and v are dominated respectively by $u', v' \in V_d$ (i.e., $u' = \text{Dom}_d(u)$, and $v' = \text{Dom}_d(v)$), add an edge from u' to v'. For such a pair $p = \{u, v\}$ we say that it is a *creating pair* for the edge (u', v'). (Notice that the definition of $E_{\ell,d}$ allows having parallel edges.) Consider now the unweighted auxiliary graph $G_{\ell,d} = (V_d, E_{\ell,d})$.

Observe that, by construction,

$$|P_\ell(C)| = |E_{\ell,d}|. \tag{2.1}$$

To complete the proof we now prove the following two lemmas:

LEMMA 2.2. *For any $d \leq 2weight(C)$ the number of vertices in the auxiliary graph $G_{\ell,d}$ can be upper bounded as*

$$|V_d| \leq \frac{weight(C)}{d/2}. \tag{2.2}$$

LEMMA 2.3. *For $\ell = \Omega(d \cdot \log n)$, the number of edges in auxiliary graph can be upper bounded as*

$$|E_{\ell,d}| = O(|V_d|) \tag{2.3}$$

Indeed, we pick $d = \Theta(\ell/\log n)$ and get Lemma 2.1. It remains to prove Lemmas 2.2, 2.3.

Proof. (of Lemma 2.2): If $|V_d| = 1$ the bound is trivial. Hence we assume that $|V_d| > 1$. Consider now the collection of $d/2$-spheres in the original network around nodes in V_d. Each one of these nodes is connected to a node outside the corresponding sphere, since all nodes are in the same connected component C. Since these nodes are d-separated, these spheres are disjoint, and the total cost sums up to $|V_d|d/2$. This cost cannot exceed the total weight of C, $weight(C)$, and thus

$$|V_d| \leq \frac{weight(C)}{d/2} \tag{2.4}$$

Proof. (of Lemma 2.3): The *girth* [Bol78] of a graph is the length of a shortest cycle. It is proved in [Bol78] that the number of edges in $G_{\ell,d}$ can be upper-bounded as

$$|E_{\ell,d}| = O(|V_d|^{1+\frac{O(1)}{g(G_{\ell,d})}}) \tag{2.5}$$

where $g(G_{\ell,d})$ denotes the girth of $G_{\ell,d}$.

We prove the following proposition

Proposition 2.1. *The girth $g(G_{\ell,d})$ of $G_{\ell,d} = (V_d, E_{\ell,d})$ is at least $\ell/(2d)$.*

Proof. Assume that there is a cycle of length $r < \ell/(2d)$. Consider the order of arrival of the edge creating pairs of the edges of the cycle. Let $p = (u, v)$ be the last pair in that order. By the definition of an edge $\text{Cost}(p) \geq \ell$. However, since all previous pairs already connected we can connect u to v thru the "detour" path in the auxiliary graph. The vertices on this path are "equivalence classes" of vertices V_C, that are dominated by the same vertex in V_d in the maximal independent set. The diameter of such equivalence class in the original network is at most $2d$. Thus, the detour path in the auxiliary graph induces a path in the original network of cost of $2dr < \ell$. This contradicts the definition of the algorithm since it uses the minimum cost path.

This completes the proof of Lemma 2.3 and lemma 2.1 and thus completes the proof Theorem 2.1.

It is worthwhile to mention that the bound appears in Lemma 2.3 (which is the heart of the proof) is almost tight. To see that the Lemma is almost tight we construct a graph of girth $g = \log n / \log\log n$ and has $m = n \log n$ edges. Such a graph exists follows from [Bol78]. Then we replace each edge in the graph by three serial edges. We associate a weight of 1 to the first and the last edges in each triplet and a weight of g to middle one. We get a sequence of requests for connecting the two endpoint of all the middle edges. It is easy to see inductively that the current shortest path between each such pair is the corresponding middle edges since every other path contains at least $2g$ side edges and has totals weight of at least $2g$. Hence the cost of the on-line algorithm is $\Omega(gm) = \Omega(n \log^2 n / \log\log n)$. On the other hand if we can build a spanning tree which consists of all the edges of size 1 and $n-1$ edges of size g and thus has a weight of $O(gn + m) = O(n \log n)$. Thus the number of times that we paid a cost of g is $\Theta(m)$ which is smaller only by a factor of $\log\log n$ from the bound that the lemma implies. We note that it is still possible that the competitive ratio of the algorithm is better than what is proved.

3 Randomized Network Connectivity Leasing Algorithms

In this section we present a randomized algorithm which is a generalization of the GSP algorithm to the network connectivity leasing problem.

Define graph $\hat{G}$ as in previous section where edges bought by the algorithm are assigned zero weight.

GSP-based Leasing Algorithm: For request $p = \{q, \bar{q}\}$ connect q to $\bar{q}$ thru the current minimum cost path in the graph $\hat{G}$. With probability $1/2F$ buy all non-bought edges in the path and otherwise lease.

Theorem 3.1. *The GSP-based randomized network connectivity leasing algorithm is $O(\log^2 n)$ competitive against adaptive on-line adversaries.*

The theorem is a consequence of a more general theorem for *task systems* [BLS87].

A metrical *forcing task system* [MMS88] is an on-line problem composed by a configurations metric space and a set of tasks. At every time the algorithm is associated with a configuration, and each task defines a set of allowable tasks, that may be associated with the algorithm after the arrival of that task.

Clearly, the generalized Steiner problem can be viewed as a forcing task system, where configurations are subgraphs of the graph, and a request sequence defines the set of all allowable configurations to be the set of subgraphs where every pair is connected by a path.

Given a forcing task system, we define the "relaxed" version of the problem. In the matching *F-relaxed* task system a request may be served in every configuration at the cost of the distance from that configuration to the nearest allowable configuration in the original problem. However changing configurations is F times more expensive.

Thus, the network connectivity leasing problem is the F-relaxed version of GSP.

Theorem 3.1 is a corollary of the following theorem of [Bar94] which is based on the *natural potential function* [BFR92]:

Theorem 3.2. *Given a c-competitive algorithm for a forcing metrical task system, there exists a $(3 - \frac{1}{F}) \cdot c$-competitive algorithm for the associated F-relaxed task system.*

Formal definitions and proofs appear in appendix A.

4 Open Problems

We prove that the Min-Cost GSP algorithm is $O(\log^2 n)$ competitive, whereas the best known lower bound which follows from the on-line Steiner tree problem is $\Omega(\log n)$. The obvious open problem is to close the gap. We conjecture that the Min-Cost GSP algorithm achieves the best possible competitive for the on-line GSP on arbitrary graphs.

For the network connectivity leasing problem we have shown that there exists a randomized on-line algorithm with competitive ratio within a constant factor from that of the Min-Cost GSP algorithm. We believe that there exists a deterministic on-line connectivity leasing algorithm with the same property. Can a similar deterministic result be obtained in the general framework of relaxed task systems ?

Acknowledgments

We thank Noga Alon for helpful discussions.

References

[AA93] N. Alon and Y. Azar. On-line steiner trees in the euclidean plane. *Discrete and Computational Geometry*, 10:113–121, 1993.

[ABF93] Baruch Awerbuch, Yair Bartal, and Amos Fiat. Competitive distributed file allocation. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 164–173, May 1993.

[AK94] S. Albers and H. Koga. New on-line algorithms for the page replication problem. In *Proc. 4th Scandinavian Workshop on Algorithmic Theory*, July 1994.

[AKR91] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 134–144, 1991.

[AP90] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *Proc. 31st IEEE Symp. on Found. of Comp. Science*, pages 514–522, 1990.

[AP91] Baruch Awerbuch and David Peleg. Concurrent online tracing of mobile users. In *Proc. of the Annual ACM SIGCOMM Symposium on Communication Architectures and Protocols, Zurich, Switzerland*, September 1991.

[Bar94] Yair Bartal. Competitive Analysis of Distributed On-line Problems — Distributed Paging, *Ph.D. Thesis*. Tel-Aviv University. Dept. of Computer Science. September, 1994.

[BFR92] Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 39–50, 1992.

[BLS87] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. In *Proc. of the 19th Ann. ACM Symp on Theory of Computing*, pages 373–382, may 1987.

[Bol78] B. Bollobás. *Extremal Graph Theory*. Academic Press, 1978.

[BS89] D.L. Black and D.D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Carnegie-Mellon, 1989.

[CV93] Rohit Chandra and Sundar Vishwanathan. Constructing reliable communication networks of small wight online. unpublished manuscript, Nov 1993.

[GW92] M. Goemans and D. Williamson. General approximation technique for constrained forest problems. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 307–315, 1992.

[IW91] M. Imaze and B.M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, august 1991.

[Kar72] R.M. Karp. *Reducibility among Combinatorial Problems, R.E. Miller and J.W. Thatcher (eds.), Complexity of Computer Computations*. Plenum Press, 1972.

[Kar92] R.M. Karp. "on-line algorithms bersus off-line algorithms: how much is it worth to know the future?,". In *Proc. World Computer Congress*, 1992.

[Kog93] H. Koga. Randomized on-line algorithms for the page replication problem. In *Proc. of the 4th International Symp. on Algorithms and Computation*, 1993.

[LRWY94] C. Lund, N. Reingold, J. Westbrook, and D. Yan. On-line distributed data management. In *Proc. of European Symp. on Algorithms*, 1994.

[MMS88] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms or on-line problems. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 322–333. ACM SIGACT, ACM, May 1988.

[Win92] P. Winter. Steiner problem in networks: A

survey. *Networks*, 17(6):129–167, June 1992.

[WY93] J. Westbrook and D.K. Yan. Greedy algorithms for the on-line steiner tree and generalized steiner problems. In *Workshop on Algorithms and Data Structures*, 1993.

A A General Theorem for Relaxed Task Systems

In this section we give a general theorem in the context of task systems ([BLS87]). This section appears in [Bar94].

Definition A.1. *A* task system, $\mathcal{P}$, *is an on-line configuration problem where the cost function has the following structure. Define the cost of a move between configurations in* Con, *denoted* $\mathrm{dist}(C_1, C_2)$ *(where* $C_1, C_2 \in Con$*) (this is the* move cost*). Associate with every request* r *and every configuration* C *the cost of serving* r *in configuration* C, *denoted* $\mathrm{task}(C, r)$ *(this is the* task cost*). The cost function of a task system is defined by:* $\mathrm{cost}(C_1, C_2, r) = \mathrm{dist}(C_1, C_2) + \mathrm{task}(C_2, r)$. *For a task system, input requests are usually called* tasks. *If the move cost function* dist *forms a metric space over* Con, *then the task system is called* metrical.

The following definition was also used in [MMS88]:

Definition A.2. *A* forcing *task system,* $\mathcal{P}$,*is a task system such that for every request* r *and every configuration* C $\mathrm{task}(C, r)$ *is either* 0 *or* ∞. *That is, for every request* r *we may associate a set of allowable configurations,* $\mathcal{C}(r)$, *in which it can be served.*

Given a forcing task system we may define the "relaxed" version of the problem, in which the request may be served in every configuration at the cost of the distance from that configuration to the nearest allowable configuration in the original problem. However changing configurations is D times more expensive.

Definition A.3. *A* D-relaxed *task system,* D-$\mathcal{P}$, *with respect to a forcing task system* $\mathcal{P}$ *and some parameter* $D \geq 1/2$, *is the task system with cost, distance, and task functions denoted* cost′, dist′ *and* task′ *respectively.* dist′ *and* task′ *are defined as follows: Given* $C_1, C_2 \in Con$, $\mathrm{dist}'(C_1, C_2) = D \cdot \mathrm{dist}(C_1, C_2)$. *Given* $C \in Con$ *and a request* r, $\mathrm{task}'(C, r) = \min_{C' \in \mathcal{C}(r)} \mathrm{dist}(C, C')$.

According to the above definition file-replication [BS89] can be viewed as the relaxed version of the on-line Steiner tree problem, connectivity leasing in networks is the relaxed version of the generalized Steiner problem, file migration is the relaxed version of the trivial 1-server problem, and similarly k-copy migration (which is a special case of the k-server with *excursions* problem [MMS88]) is the relaxed version of the k-server problem.

In this section we show that the competitive ratio for a metrical forcing task system, $\mathcal{P}$, and the D-relaxed task system, D-$\mathcal{P}$, against adaptive on-line adversaries, are within a constant factor.

Let Alg be a c-competitive algorithm for $\mathcal{P}$, and let $D \geq 1/2$. We show that Alg can be used to give a competitive randomized algorithm for the relaxed task system D-$\mathcal{P}$.

Algorithm **D-Alg**.
Algorithm D-Alg simulates a version of algorithm Alg. At all times, the configuration of D-Alg is equal to that of the simulated version of Alg.

Let the current configuration of the algorithm be B.

Upon receiving a request r, with probability $1/2D$, feed Alg with new request r, and change the configuration to the new (allowable) configuration B' of Alg.

With probability $1 - 1/2D$, the algorithm stays in configuration B.

Theorem A.1. *Let* $\mathcal{P}$ *be a forcing metrical task system, and let Alg be* c*-competitive algorithm for* $\mathcal{P}$ *against adaptive on-line adversaries. Algorithm D-Alg is* $(3 - \frac{1}{D}) \cdot c$*-competitive for the* D*-relaxed task system,* D-$\mathcal{P}$, *against adaptive on-line adversaries, for* $D \geq 1/2$.

The proof makes use of the *natural potential function* [BFR92], $\mathrm{Up}(h, A)$, a nonnegative function of the algorithm history h and adversary configuration A. For any forcing task system algorithm that is c-competitive against adaptive on-line adversaries, the natural potential function is a *one-step* potential function, that is it has the following properties:

- When the adversary changes configuration, Up increases by at most c times its cost.

- When the on-line algorithm serves the request, Up decreases by at least the expected on-line cost for the request.

Proof. Let Up be the natural potential function for Alg. We have that Up is a one-step potential function. We use it to define a new one-step potential function Φ for algorithm D-Alg. Let h_n be the history of D-Alg. This history explicitly defines the history of the current version of Alg that D-Alg simulates, denoted $\widehat{h}_n$.

Let σ_n be the sequence of requests already fed to Alg since. Let A_n denote the adversary's current configuration, let B_n denote the on-line algorithm's current configuration. The potential function for D-Alg is: $\Phi(h_n, A_n) = (3D-1) \cdot \overline{\mathrm{Up}}(\widehat{h}_n, A_n)$.

Let r_n be last request in σ_n. $\overline{\mathrm{Up}}$ is defined by

$$\overline{\mathrm{Up}}(\widehat{h}_n, A_n) = \min_{\overline{A} \in \mathcal{C}(r_n)} \{\mathrm{Up}(\widehat{h}_n, \overline{A}) + c \cdot \mathrm{dist}(\overline{A}, A_n)\}.$$

Clearly Φ is nonnegative as Up is a potential function.

Let $\overline{A}$ denote the configuration that minimizes $\overline{\mathrm{Up}}$. Along the proof we will bound the change in $\overline{\mathrm{Up}}$ by extracting a new configuration $\overline{A}_{n+1} \in \mathcal{C}(r_{n+1})$. The new value of $\overline{\mathrm{Up}}$ may only increase if we use the configuration $\overline{A}_{n+1}$ instead of minimizing.

When analyzing the adversary cost we separate between its configuration changes cost and its task costs.

We view the process as if the adversary has made its move from A_n to A_{n+1} before the next request, r_{n+1}, has arrived, and only then we analyze the change in the potential function due to the request.

Adversary Move.

When the adversary moves from configuration A_n to configuration A_{n+1}, we can bound the change in $\overline{\mathrm{Up}}$ by not changing $\overline{A}$. Thus we obtain

$$\begin{aligned} \Delta\Phi &= (3D-1) \cdot \Delta\overline{\mathrm{Up}} \\ &\leq (3D-1) \cdot c \cdot (\mathrm{dist}(\overline{A}, A_{n+1}) - \mathrm{dist}(\overline{A}, A_n)) \\ &\leq (3 - \tfrac{1}{D}) \cdot c \cdot D \cdot \mathrm{dist}(A_n, A_{n+1}). \end{aligned}$$

Request Analysis.

Let the next request be r_{n+1}. We show that the change in the potential is bounded above by a constant times the task cost of the adversary to serve the request (not including its move cost) minus the expected work done by D-Alg for serving the request and for changing configuration.

The expected cost of algorithm D-Alg is

$$\begin{aligned} & \mathrm{E}(\mathrm{Cost}_{D-\mathrm{Alg}}(h_n, r_{n+1})) \\ = \; & \tfrac{1}{2D} \cdot D \cdot \mathrm{E}(\mathrm{Cost}_{\mathrm{Alg}}(\widehat{h}_n, r_{n+1})) \\ + \; & (1 - \tfrac{1}{2D}) \cdot \min_{B \in \mathcal{C}(r_{n+1})} \mathrm{dist}(B_n, B) \\ \leq \; & \tfrac{1}{2D} \cdot D \cdot \mathrm{E}(\mathrm{Cost}_{\mathrm{Alg}}(\widehat{h}_n, r_{n+1})) \\ + \; & (1 - \tfrac{1}{2D}) \cdot \mathrm{E}(\mathrm{Cost}_{\mathrm{Alg}}(\widehat{h}_n, r_{n+1})) \\ = \; & \tfrac{3D-1}{2D} \cdot \mathrm{E}(\mathrm{Cost}_{\mathrm{Alg}}(\widehat{h}_n, r_{n+1})). \end{aligned}$$

Now we turn to analyzing the expected change in Φ. To do that we bound the change in $\overline{\mathrm{Up}}$ in the case that r_{n+1} is fed to Alg, by choosing $\overline{A}_{n+1} \in \mathcal{C}(r_{n+1})$ to be the configuration $\overline{A}$ minimizing $\mathrm{dist}(\overline{A}, A_{n+1})$.

$$\begin{aligned} \mathrm{E}(\Delta\Phi) &= (3D-1) \cdot \mathrm{E}(\Delta\overline{\mathrm{Up}}) \\ &\leq (3D-1) \cdot \tfrac{1}{2D} \cdot [\mathrm{E}(\mathrm{Up}(\widehat{h}_{n+1}, \overline{A}_{n+1}) \\ &- \mathrm{Up}(\widehat{h}_n, \overline{A}_n)) \\ &+ c \cdot (\mathrm{dist}(\overline{A}_{n+1}, A_{n+1}) - \mathrm{dist}(\overline{A}_n, A_{n+1}))] \\ &\leq \tfrac{3D-1}{2D} \cdot [\mathrm{E}(\mathrm{Up}(\widehat{h}_{n+1}, \overline{A}_{n+1}) \\ &- \mathrm{Up}(\widehat{h}_n, \overline{A}_{n+1})) \\ &+ (\mathrm{Up}(\widehat{h}_n, \overline{A}_{n+1}) - \mathrm{Up}(\widehat{h}_n, \overline{A}_n)) \\ &+ c \cdot (\mathrm{dist}(\overline{A}_{n+1}, A_{n+1}) - \mathrm{dist}(\overline{A}_n, A_{n+1}))]. \end{aligned}$$

Now using the properties of the natural potential function for the task system $\mathcal{P}$, implies

$$\begin{aligned} \mathrm{E}(\Delta\Phi) &\leq \tfrac{3D-1}{2D} \cdot [c \cdot (\mathrm{dist}(\overline{A}_n, \overline{A}_{n+1}) \\ &+ \mathrm{dist}(\overline{A}_{n+1}, A_{n+1}) - \mathrm{dist}(\overline{A}, A_{n+1})) \\ &- \mathrm{E}(\mathrm{Cost}_{\mathrm{Alg}}(\widehat{h}_n, r_{n+1}))] \\ &\leq \tfrac{3D-1}{2D} \cdot [2c \cdot \mathrm{dist}(\overline{A}_{n+1}, A_{n+1}) \\ &- \mathrm{E}(\mathrm{Cost}_{\mathrm{Alg}}(\widehat{h}_n, r_{n+1}))] \\ &= (3 - \tfrac{1}{D}) \cdot c \cdot \mathrm{task}(A_{n+1}, r_{n+1}) \\ &- \mathrm{E}(\mathrm{Cost}_{D-\mathrm{Alg}}(h_n, r_{n+1})). \end{aligned}$$