Optimal Communication Spanning Trees (OCT)

Kun-Mao Chao @2019

nonnegative

$G = (V, E, w)$

Requirements $\lambda(u,v)$ for each pair of vertices

$$\min \sum_{u,v \in V} \lambda(u,v)\, d_T(u,v)$$

- $\lambda(u,v) = 1 \Leftarrow$ MRCT
- $\lambda(u,v) = r(u) \times r(v)$ $\quad \boxed{r: V \to Z_0^+}$

  PROCT

  (Optimal Product-Requirement Communication Spanning Trees)
- $\lambda(u,v) = r(u) + r(v)$

  SROCT

  (Optimal Sum-Requirement Communication Spanning Trees).

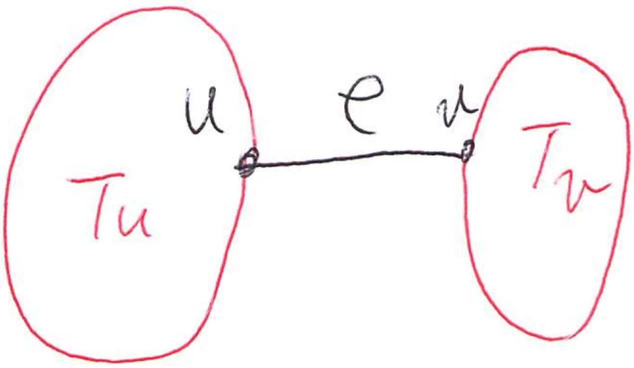
PROCT

Kun-Mao Chao @2019

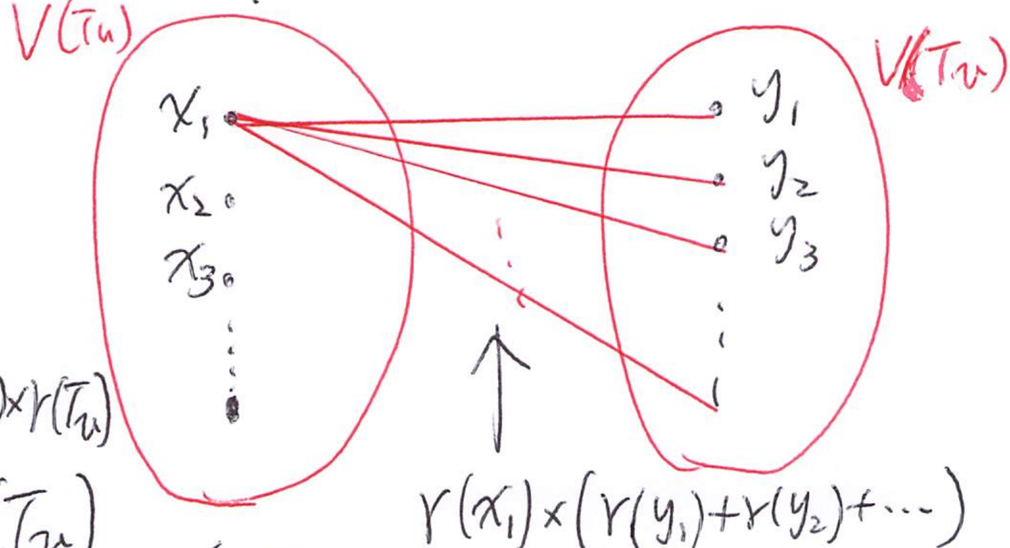
$$C_p(T,r) = \sum_{u,v} r(u)\times r(v)\times d_T(u,v)$$

$$= \sum_{e\in E} l_p(T,r,e)\cdot w(e), \text{ where}$$

$$l_p(T,r,e) = 2\times r(T_u)\times r(T_v)$$



$$r(T_u) = \sum_{x\in V(T_u)} r(x) \qquad r(T_v) = \sum_{y\in V(T_v)} r(y)$$



$$(r(x_1)+r(x_2)+\cdots)\times r(T_v)$$
$$= r(T_u)\times r(T_v)$$

$$\Longleftarrow$$

$$r(x_1)\times(r(y_1)+r(y_2)+\cdots)$$
$$= r(x_1)\times r(T_v)$$

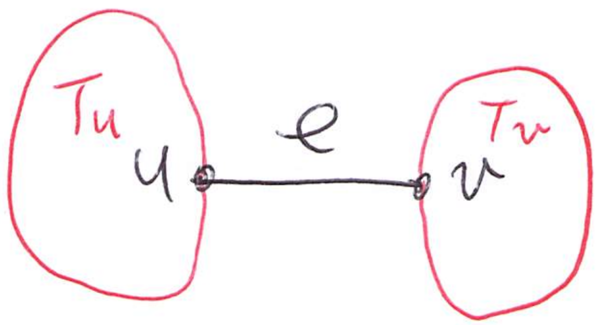
SROCT $\quad 2\sum_{u,v} r(u)\, d_T(u,v)$ Kun-Mao Chao @2019 Seoul

$$\left(\sum r(u)\, d_T(u,v) = \sum r(v)\, d_T(u,v)\right)$$

$$C_S(T,r) = \sum_{u,v} (r(u)+r(v))\, d_T(u,v)$$

$$= \sum_{e\in E} l_S(T,r,e)\cdot w(e), \text{ where}$$

$$l_S(T,r,e) = 2\,(r(T_u)\times|V(T_v)| + r(T_v)\times|V(T_u)|)$$



For each $x\in V(T_u)$, $r(x)\cdot w(e)$ is counted $2\times|V(T_v)|$ times.

For each $y\in V(T_v)$, $r(y)\cdot w(e)$ is counted $2\times|V(T_u)|$ times.

Summing them up, we have

$$l_S(T,r,e) = 2\times(r(T_u)\times|V(T_v)| + r(T_v)\times|V(T_u)|)$$

A note on optimal communication spanning trees

Bang Ye Wu Kun–Mao Chao

The *optimal communication spanning tree* (OCT) problem is defined as follows. Let $G = (V, E, w)$ be an undirected graph with nonnegative edge length function w. We are also given the requirements $\lambda(u, v)$ for each pair of vertices. For any spanning tree T of G, the communication cost between two vertices is defined to be the requirement multiplied by the path length of the two vertices on T, and the communication cost of T is the total communication cost summed over all pairs of vertices. Our goal is to construct a spanning tree with minimum communication cost. That is, we want to find a spanning tree T such that $\sum_{u,v \in V} \lambda(u, v) d_T(u, v)$ is minimized.

The requirements in the OCT problem are arbitrary nonnegative values. By restricting the requirements, several special cases of the problem were defined in the literature. We list the problems in the following, in which $r : V \rightarrow Z_0^+$ is a given vertex weight function and $S \subset V$ is a set of k vertices given as sources.

- $\lambda(u, v) = 1$ for each $u, v \in V$: This version is the MINIMUM ROUTING COST SPANNING TREE (MRCT) problem.
- $\lambda(u, v) = r(u)r(v)$ for each $u, v \in V$: This version is called the OPTIMAL PRODUCT-REQUIREMENT COMMUNICATION SPANNING TREE (PROCT) problem.
- $\lambda(u, v) = r(u) + r(v)$ for each $u, v \in V$: This version is called the OPTIMAL SUM-REQUIREMENT COMMUNICATION SPANNING TREE (SROCT) problem.
- $\lambda(u, v) = 0$ if $u \notin S$: This version is called the p-SOURCE OCT (p-OCT) problem. In other words, the goal is to find a spanning tree minimizing $\sum_{u \in S} \sum_{v \in V} \lambda(u, v) d_T(u, v)$.
- $\lambda(u, v) = 1$ if $u \in S$, and $\lambda(u, v) = 0$ otherwise: This version is called the p-Source MRCT (p-MRCT) problem. In other words, the goal is to find a spanning tree minimizing $\sum_{u \in S} \sum_{v \in V} d_T(u, v)$.

The relationship of the different versions of the OCT problems is illustrated in Figure 1. Table 1 summarizes the results.

Bibliographic Notes and Further Reading

The application of Yair Bartal's algorithm [1] to approximating the OCT problem was pointed out in [10]. A better bound was later given by Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar [5].

The PROCT and SROCT problems were introduced in [8]. In that paper, Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang gave a 1.577-approximation algorithm for the PROCT problem and a 2-approximation algorithm for the SROCT problem. The PTAS using the Scaling-and-Rounding technique for a PROCT problem was presented in [9] by the same authors. Scaling the input instances is a technique that has been used to balance the running time and the approximation ratio. For example, Oscar H. Ibarra and Chul E. Kim used the scaling technique to develop a *fully polynomial time approximation scheme* (FPTAS) for the

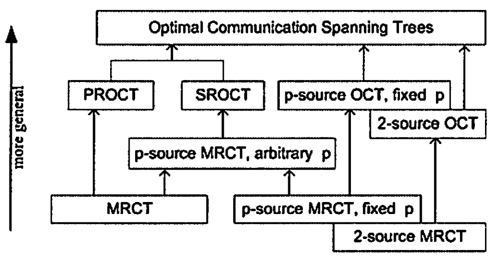


Figure 1: The relationship of the OCT problems

4

Table 1: The objectives and currently best ratios of the OCT problems.

Problem	Objective	Ratio
OCT	$\sum_{u,v} \lambda(u,v) d_T(u,v)$	$O(\log n)$
PROCT	$\sum_{u,v} r(u)r(v)d_T(u,v)$	PTAS
SROCT	$\sum_{u,v} (r(u)+r(v))d_T(u,v)$	2
MRCT	$\sum_{u,v} d_T(u,v)$	PTAS
p-MRCT	$\sum_{u\in S}\sum_{v\in V} d_T(u,v)$	2
2-MRCT	$\sum_{v} (d_T(s_1,v)+d_T(s_2,v))$	PTAS

*Metric SROCT PTAS

knapsack problem [4], and some improvement was made by Eugene L. Lawler [6]. A nice explanation of the technique can also be found in [3](pp. 134–137).

The NP-hardness of the 2-MRCT was shown by Bang Ye Wu [7], in which the reduction is from the Exact Cover By 3-Sets (X3C) problem ([SP2] in [3]). The transformation is simpler and easier to extend to the weighted case, which is designed to show the NP-hardness of the p-MRCT problem for any fixed p. A similar reduction (for 2-MRCT) was also shown by Harold Connamacher and Andrzej Proskurowski [2]. They showed that the 2-MRCT problem is NP-hard. The PTAS for the 2-MRCT problem also appeared in [7]. In addition to the PTAS for the 2-MRCT problem, there is also a PTAS for the weighted 2-MRCT problem. But the PTAS works only for metric inputs and the counterpart on general graphs was left as an open problem.

References

[1] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[2] H.S. Connamacher and A. Proskurowski. The complexity of minimizing certain cost metrics for k-source spanning trees. *Discrete Appl. Math.*, 131:113–127, 2003.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, San Francisco, 1979.

[4] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22:463–468, 1975.

[5] K. Talwar J. Fakcharoenphol, S. Rao. A tight bound on approximating arbitrary metrics by tree metrics. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 448–455, 2003.

[6] E.L. Lawler. Fast approximation algorithms for knapsack problems. *Math. Oper. Res.*, 4(4):339–356, 1979.

[7] B.Y. Wu. A polynomial time approximation scheme for the two-source minimum routing cost spanning trees. *J. Algorithms*, 44:359–378, 2002.

[8] B.Y. Wu, K.-M. Chao, and C.Y. Tang. Approximation algorithms for some optimum communication spanning tree problems. *Discrete Appl. Math.*, 102:245–266, 2000.

[9] B.Y. Wu, K.-M. Chao, and C.Y. Tang. A polynomial time approximation scheme for optimal product-requirement communication spanning trees. *J. Algorithms*, 36:182–204, 2000.

[10] B.Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, and C.Y. Tang. A polynomial time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.*, 29:761–778, 2000.

5

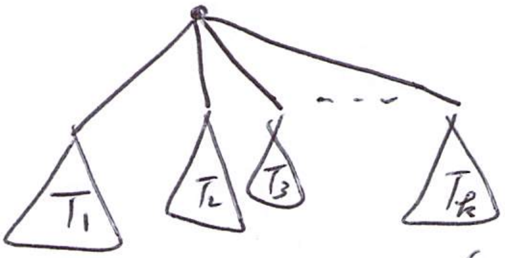
r-centroid

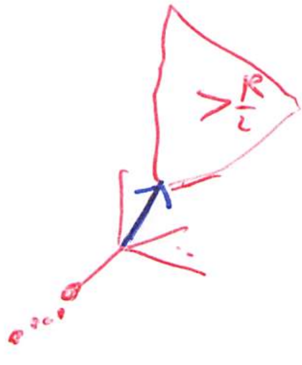
Kun-Mao Chao
@2019 Seoul

$$R = \sum_{v \in V} r(v)$$

T: r-centroid

(optimal SROCT)





$r(T_1) \le \frac{R}{2}$ ... $r(T_k) \le \frac{R}{2}$

x_1 = centroid

x_2 = r-centroid

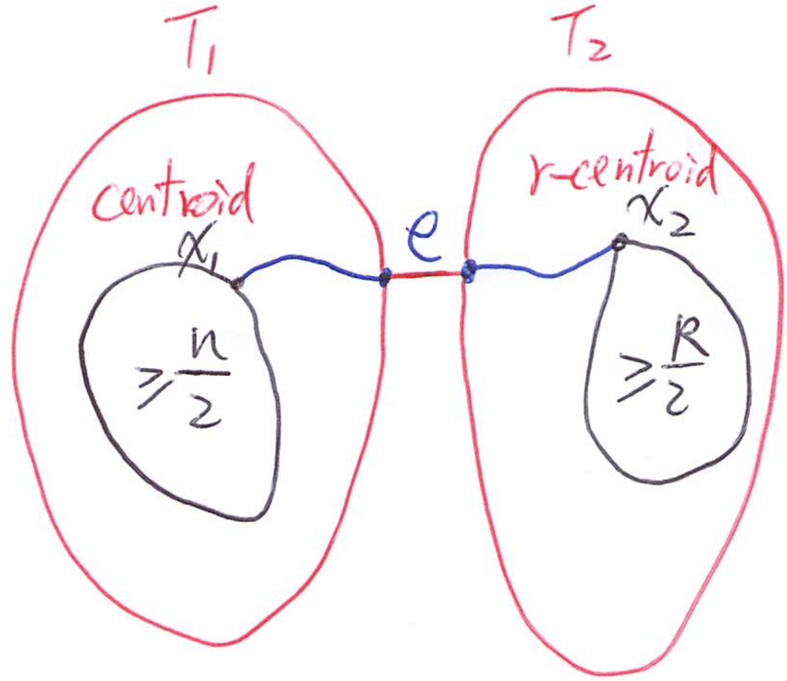
$P = SP_T(x_1, x_2)$

Lemma: For any edge $e \in E(P)$, $l_s(T, r, e) \ge nR$.

Lemma: For any edge $e \in E(P)$, $l_s(T, r, e) \geq nR$.

Kun-Mao Chao @2019

pf.



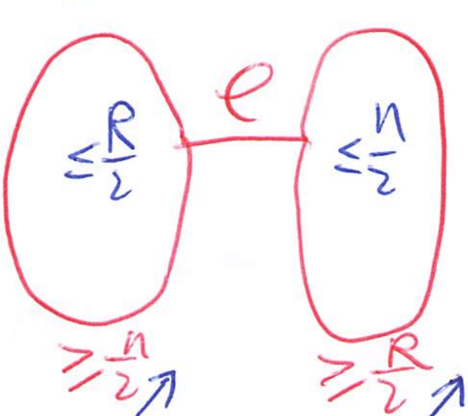
$$l_s(T, r, e) = 2\left(|V(T_1)| \cdot r(T_2) + \underline{|V(T_2)| \cdot r(T_1)}\right)$$

$$= 2\left(|V(T_1)| \cdot r(T_2) + \underline{(n - |V(T_1)|)(R - r(T_2))}\right)$$

$$= 2\left(2|V(T_1)| r(T_2) - n\, r(T_2) - R|V(T_1)| + nR\right)$$

$$= 2\left(2\underbrace{\left(|V(T_1)| - \frac{n}{2}\right)}_{\geq 0}\underbrace{\left(r(T_2) - \frac{R}{2}\right)}_{\geq 0} + \frac{1}{2} nR\right)$$

$$\geq 2 \times \frac{1}{2} nR = nR$$



$$2\left(\frac{n}{2} \cdot \frac{R}{2} + \frac{n}{2} \cdot \frac{R}{2}\right)$$

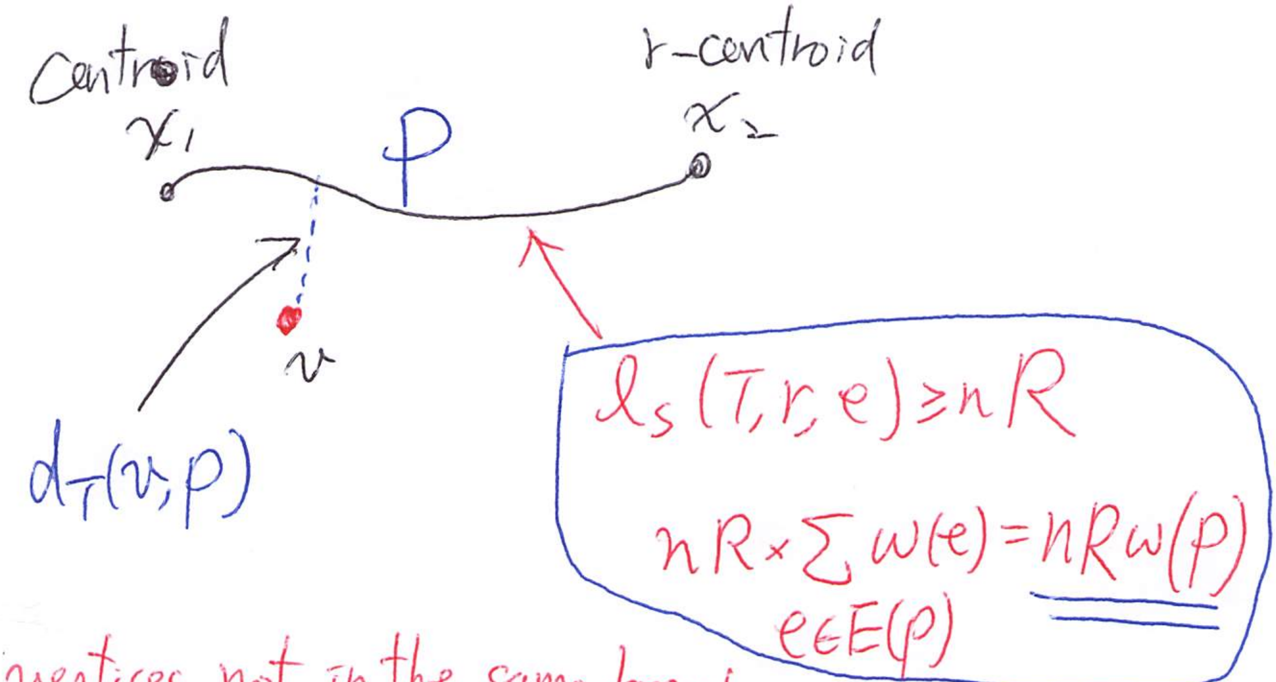
$$= nR$$

min. $(= \frac{n}{2})$ min. $(= \frac{R}{2})$

Kun-Mao Chao @2019

$$C_S(T) \geq \sum_{v \in V} (n r(v) + R) d_T(v, P) + n R w(P)$$

pf.



$\geq \frac{n}{2}$ vertices not in the same branch

$\Rightarrow$ $\underline{r(v) \cdot d_T(v,P)}$ will be counted at least

$$\frac{n}{2} \times \underline{2} \text{ times}$$

"from v" & "to v"

$\geq \frac{R}{2}$ vertex weight in total not in the same branch

$\Rightarrow$ $\underline{\frac{R}{2} \cdot d_T(v,P) \times 2}$ "from v" & "to v"

Thus, $C_S(T) \geq \sum_{v \in V} \left(n r(v) d_T(v,P) + R \cdot d_T(v,P) \right)$

$+ nRw(P)$

Kun-Mao Chao @2019

Let Y be a spanning tree of G. For any vertex $x \in V$,

$$C_s(Y) \le 2 \sum_{v \in V} (n r(v) + R)\, d_Y(v, x).$$

pf.

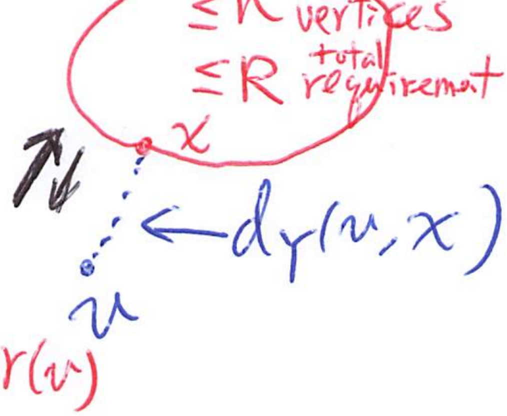
$$C_s(Y) = \sum_{u,v \in V} (r(u) + r(v))\, d_Y(u, v)$$

$$\le \sum_{u,v \in V} (r(u)+r(v))\big(\underline{d_Y(u,x)} + \underline{d_Y(x,v)}\big)$$

$$= 2 \times \sum_{u,v \in V} (r(u)+r(v))\, d_Y(v,x)$$

$$= 2 \times \sum_v \overbrace{\underbrace{\sum_u (r(u)}_{R} + r(v))}^{n r(v)}\, d_Y(v,x)$$

$$= 2 \times \sum_v (n r(v) + R)\, d_Y(v,x)$$



2-approximation for SROCT

Kun-Mao Chao
@2019

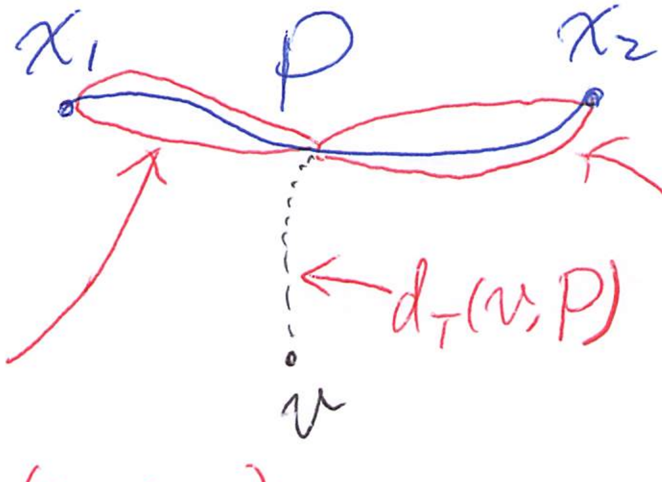
Y^*: shortest-path tree rooted at x_1 (centroid)

Y^{**}: " at x_2 (r-centroid)

$$C_s(Y^*) \le 2 \sum_{v \in V} (n r(v) + R) \cdot d_{Y^*}(v, x_1)$$

$$C_s(Y^{**}) \le 2 \sum_{v \in V} (n r(v) + R) \cdot d_{Y^{**}}(v, x_2)$$

T:



$h_1(v) + h_2(v) = \omega(P)$

$h_1(v) = \omega(SP_T(v, x_1) \cap P)$

$h_2(v) = \omega(SP_T(v, x_2) \cap P)$

$$d_{Y^*}(v, x_1) \le d_T(v, P) + h_1(v)$$

$$d_{Y^{**}}(v, x_2) \le d_T(v, P) + h_2(v)$$

$$\min\{C_s(Y^*), C_s(Y^{**})\}$$

Kun-Mao Chao@2019

$$\le \frac{C_s(Y^*) + C_s(Y^{**})}{2}$$

$$\le \frac{2\sum_{v \in V} (n\,r(v) + R)\cdot\left(d_{Y^*}(v, x_1) + d_{Y^{**}}(v, x_2)\right)}{2}$$

$$\le \sum_{v \in V} (n\,r(v) + R)\cdot\left(d_T(v,P) + h_1(v) + d_T(v,P) + h_2(v)\right)$$

$$= \sum_{v \in V} (n\,r(v) + R)\cdot\left(2\,d_T(v,P) + w(P)\right)$$

$$= 2\sum_{v \in V} (n\,r(v) + R)\cdot d_T(v,P) + \sum_{v \in V} (\overset{\sum r(v)=R}{n\,r(v)} + \overset{n \times}{R})\cdot w(P)$$

$$= 2\sum_{v \in V} (n\,r(v) + R)\cdot d_T(v,P) + 2nR\,w(P)$$

$$= 2\Big(\underbrace{\sum_{v \in V} (n\,r(v) + R)\cdot d_T(v,P) + nR\,w(P)}_{\le C_s(T)}\Big)$$

$$\le 2\,C_s(T)$$