Energy-Efficient Data Collection in Robotic Sensor Networks

Christopher Beauchamp, Soham Patil, Bin Tang

Department of Computer Science, California State University Dominguez Hills

Abstract

We study how to collect data efficiently in robotic sensor networks (RSNs) and propose a new algorithmic framework called budget-constrained covering salesman problem *(i.e., BC-CSP). Given an RSN graph wherein sensor nodes have data of different values and a robot (i.e., salesman) with limited battery power (i.e., budget), the goal of the BC-CSP is to find a* data-covering tour *for the robot to collect data with the maximum sum of values before running out of battery and returning to the depot. We propose a suite of algorithmic solutions to solve the BC-CSP, including Integer Linear programming (ILP), Greedy, and Random algorithms. Using commonly adopted mobility models of the robots and realistic battery power measurements from robotic applications, we show that a) Greedy performs very close to ILP, collecting the number of packets within 4.3% to 14.2% of ILP, and b) Greedy significantly outperforms Random by collecting between 41.9% and 80.1% more packets.*

Keywords – robotic sensor networks, budget-constrained covering salesman problem, algorithms.

1. Introduction

Background and Motivation. Robotic sensor networks (RSNs) are wireless sensor networks consisting of static sensors and mobile robots that collectively perform sensing, communication, and actuation in physical environments [1]. RSNs significantly enhance our ability to monitor and interact with the physical world. The wide range of land applications of the RSN includes industry automation [2], security and intrusion detection [3], and environmental monitoring and disaster relief [4].

For robots to operate at the above large-scale sensing applications, they must be untethered and powered by rechargeable batteries. A vital performance measurement of a modern rechargeable battery is its *energy density*, the amount of energy it can store with a given weight or volume. Although various efforts have been proposed to increase the energy density of Li-ion batteries [5], one of the most popular commercial energy sources, its current energy density value of 250 Wh/kg makes it too hefty for a mobile robot to move around even for a couple of hours.

Research Question. Therefore, it could happen that the robot does not have enough battery power to visit all parts of the sensing area in an RSN application. This is especially true for many large-scale sensing applications such as disaster relief and underwater exploration, wherein robots are dispatched into a vast area for a relatively long period (e.g., one day). In this paper, we specifically focus on a robot's limited battery power and a following question: *how to find a path for a battery-constrained robot to maximize its data-collection performance in a large-scale RSN application?*

Our Contributions. In our RSN model, the sensor nodes have already generated various data packets from the environment. We use the number of data packets to indicate the value of information available at a node (thus the importance of visiting this node by the robot). A robot with limited battery power is dispatched from a depot of the RSN into the network to collect those data packets. The depot consists of a base station, wherein the data brought back by the robot can be uploaded for further analysis and actuation, and a charging station, wherein the robot can be fully recharged. The robot collects the data packets wirelessly from the sensor nodes it visits and from all the sensor nodes within its wireless data-collecting range. We say the robot *covers* the sensor nodes from which it collects packets. The goal is for the robot to visit a sequence of sensor nodes (referred to as a *data-covering route*) to collect as many data packets as possible before it runs out of battery power and returns to the depot, where the robot uploads its collected data packets and fully recharge its battery power. We refer to this problem as *data collection in RSNs* (DCR).

Underlying DCR is a new graph-theoretical problem, which we call *budget-constrained covering salesman problem* (i.e., BC-CSP). Given a graph wherein each node has a prize, each edge has a cost, and a salesman with a limited budget, the BC-CSP aims to find a *Hamiltonian covering tour* that collects the maximum amount of prizes within the budget. Here, "covering" means that the salesman collects prizes from the nodes he visits and those within some distance from a visited node. To the extent of our knowledge, BC-CSP has not been identified and solved by the network community.

We design a suite of algorithms, including optimal integer linear programming (ILP) and heuristic greedy and random algorithms to solve the BC-CSP. Using commonly adopted mobility models of the robots and realistic battery power measurements ot robots, we show that a) greedy performs very close to ILP, collecting the number of packets within 4.3% to 14.2% of ILP, and b) greedy significantly outperforms random by collecting between 41.9% and 80.1% more packets.

2. Problem Formulation of DCR

System Model. We model the RSN as an area of len meter $\times$ wid meter, where n sensor nodes $V_s = \{1, 2, ..., n\}$ are randomly placed inside the RSN. Sensor node $i \in V_s$ is located at (x_i, y_i), $0 \leq x_i \leq len$, $0 \leq y_i \leq wid$, and it has generated $d_i > 0$ data packets, each is of b-bit. A depot, denoted as $s = (0, 0)$, is located at one corner of the RSN. The depot has the functions of both a base station for the robot to upload its collected packets and a charging station to charge the robot when it returns from its data-collecting trip. Let $c(i, j)$ denote the Euclidean distance between $i, j \in V$, where $V = V_s \cup \{s\}$; $c(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. We assume the robot has a wireless sensing range of T_r; that is, the robot can sense and collect packets from any sensor nodes within T_r distance. We refer to T_r as the robot's *data-covering range*. □

Mobility Graph and Data Collection Graph. There are two different graphs for the RSN. First, the robot's movement can be characterized by a complete graph $G(V, E)$, where the weight $w(i, j)$ of any edge $(i, j) \in E$ is $c(i, j) \times \mu$, the robot's mobility energy consumption moving from i to j. We refer to this complete graph as the *mobility graph* of the RSN. On the other hand, the robot's data-collecting behavior is modeled by a *data collection graph* $G_1(V, E_1)$, where an edge $(i, j) \in E_1$ if $c(i, j) \leq T_r$, $\forall i, j \in V$. For any node $i \in V$, denote i's *1-covered nodes* as $N_i^1 = \{j | (i, j) \in E_1\}$. When the robot visits i, it collects packets from i and all its 1-covered nodes N_i^1 (if there are any). Then both node i and its 1-covered nodes N_i^1 are *covered* by the robot. Note that G_1 is a subgraph of G. We thus use the mobility graph G as the input graph for the DCR.

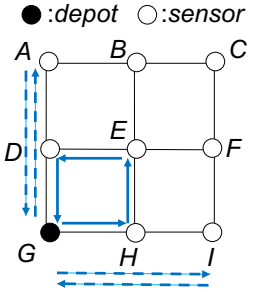


Figure 1. An example.

Energy Model. There are two primary components of a robot's energy consumption during its data-collecting process [6]. One is its *mobility energy*, the energy associated with the robot's movement that overcomes the friction between its wheels and the terrain. The maximum distance a wheeled robot of battery power $\mathcal{E}$ can move is $d = \frac{\mathcal{E}}{w \times C_{crr}}$ [6]. Here, C_{crr} is the rolling friction coefficient of the terrain, and w is the robot's weight. We define a robot's *mobility coefficient* as the battery power consumed per unit of traveled distance and denote it as μ; $\mu = w \times C_{crr}$. For a robot to move l meters, its mobility energy consumption, denoted as E_m, is $E_m = \mu \times l$.

The other component is *robotics energy* E_c, which powers the robot's sensing, computing, and communication capabilities. In data collection, E_c is mainly the robot's wireless energy consumption when sensing and collecting data packets from sensor nodes. For a robot to collect a data packet of b-bit within T_r distance, its robotics energy $E_c = \epsilon_e \cdot b$, where $\epsilon_e = 100$nJ/bit is the energy consumption per bit on the circuit hardware of the robot [7].

Problem Formulation of DCR. Given the mobility graph G, let $R = \{s, t_1, t_2, ..., t_a, s\}$ be a *data-covering tour* of the robot, where the robot starts from depot s, visits a sequence of a distinct sensor nodes $t_j \in V_s$, $1 \leq j \leq a$, $1 \leq a \leq n$, to collect the data packets from t_j and its 1-covered nodes, and returns at depot s before running out of battery. The mobility energy of the robot along R is thus $E_m = \mu \times \big(c(s, t_1) + \sum_{i=1}^{a-1} c(t_i, t_{i+1}) + c(t_a, s)\big)$. Let $\mathcal{N}_R = \bigcup_{j=1}^{a} N_{t_j}^1$ denote all the sensor nodes that are 1-covered by at least one node in R. Denote the total number of data packets the robot collects when moving along R as D_R; $D_R = \sum_{j \in R \cup \mathcal{N}_R} d_j \cdot b$. Thus the robotics energy of the robot is $E_c = \epsilon_e * D_R$. Denote the total battery power spent by the robot along R as E_R; $E_R = E_m + E_c$. Given the initial battery power $\mathcal{E}$ of the robot, the goal of the DCR is to find the robot an *optimal data-covering tour* R to maximize D_R while $E_R \leq \mathcal{E}$.

EXAMPLE 1: Fig. 1 shows a small RSN with nine nodes, wherein node G is the depot and other nodes are sensor nodes (we use the grid network only for illustration purposes). For clarity, we only show the data collection graph, and each sensor node has one data packet available; the weight of each edge is one unit (i.e., the mobility energy of the robot on each edge is one unit). The robot's initial battery power $\mathcal{E} = 4$.

Given this battery constraint, there are several feasible data-covering tours for the robot: a) G-H-E-D-G (solid blue lines), b) G-H-I-H-G (dashed blue lines), and c) G-D-A-D-G (dashed blue lines). Tour a) collects a maximum of 7 data packets while tours b) and c) each collect 5 data packets, all giving the robot a remaining battery power of zero when it returns to G. Other feasible tours involve the robot moving directly between G and E; they are not optimal. E.g., G-E-H-G collects 6 data packets, with a remaining energy of $2 - \sqrt{2}$ for the robot.

BC-CSP. Given a complete graph $G'(V', E')$ where a node $i \in V'$ has a prize $p_i \geq 0$ and an edge $(u, v) \in E'$

has a weight $w(u,v) \geq 0$. Each node $i \in V'$ can cover a subset set of nodes $S_i \subset V'$, referring to i's *covering set*. A traveling salesman is located at $r \in V'$ and has a budget of $\mathcal{B}$, the maximum distance he can travel before returning to r. When the salesman visits a node i, he can collect prizes from i and all nodes in S_i (if they are still available). We assume each prize can be collected at most once. Given a *prize-covering cycle* $R = \{r = v_1, v_2, v_3, ..., v_x = r\}$ of the salesman, the total prize it collects is $P_R = \sum_{i \in R \cup \{j | j \in S_i \wedge i \in R\}} p_i$ and the cost along R as $C_R = \sum_{i=1}^{x-1} w(v_i, v_{i+1})$. The BC-CSP aims to find a R to maximize P_R under the budget constraint that $C_R \leq \mathcal{B}$. BC-CSP is NP-hard as its special case of CSP, where $\mathcal{B} = +\infty$, is NP-hard [8]. We give the theorem below without proof due to space constraints.

Theorem 1: DCR on the mobility graph $G(V,E)$ is a BC-CSP on $G'(V',E')$.

3. An Optimal ILP Algorithm

We formulate DCR as an ILP, as shown in ILP (A). We introduce four decision variables: $x_{i,j}$ indicating if edge (i,j) is on the data-covering tour of the robot; y_i indicating if node i is visited by the robot (i.e., on the data-covering tour); z_i indicating if node i's data packets are collected by the robot (i.e., i is either visited by the robot or 1-covered by a node visited by the robot). u_i is a position variable showing the order in which the node i is visited. $u_s = 1$ as s, being the depot, are both starting and ending nodes.

$$(A) \quad \max \sum_{j \in V_s} d_i \cdot z_i \tag{1}$$

s.t.

$$x_{i,j}, y_i, z_i \in \{0,1\}, \quad \forall i,j \in V \tag{2}$$

$$\sum_{j \in V_s} x_{s,j} = \sum_{i \in V_s} x_{i,s} = 1 \tag{3}$$

$$\sum_{j \in V} x_{i,j} = \sum_{j \in V} x_{j,i} = y_i \leq 1, \quad \forall i \in V_s \tag{4}$$

$$\sum_{i \in V} \sum_{j \in V} (d_{i,j} \cdot x_{i,j} \cdot \mu) \leq \mathcal{E}, \tag{5}$$

$$z_i \geq y_i + \sum_{j : c(i,j) \leq T_r} y_j, \quad \forall i \in V_s \tag{6}$$

$$2 \leq u_i \leq n, \quad \forall i \in V_s \tag{7}$$

$$u_i - u_j + 1 \leq n \cdot (1 - x_{i,j}). \quad \forall i,j \in V_s \tag{8}$$

Objective function 1 is to maximize the total number of data packets collected from all the sensor nodes covered by the robot. Constraint 2 shows the integer constraints of $x_{i,j}$, y_i, and z_i. Constraint 3 ensures that the data-covering tour starts from and ends at depot s. Constraint 4 ensures that each sensor node in V_s is visited by the robot at most once. Constraint 5 enforces the battery power constraint of the robot. Constraint 6 is the *covering constraint*, indicating that if a node is visited or within T_r of any visited node, it is covered by the robot, and the robot collects its packets. Constraints 7 and 8 are Miller–Tucker–Zemlin (MTZ) formulation for subtour elimination [9] that guarantee the final data-covering tour is one tour instead of multiple tours.

4. Greedy Data-Covering Algorithms

As the above ILP(A) is time-consuming to compute, we present two efficient and easy-to-implement algorithms: the prize-based covering greedy algorithm (Algo. 1) and a random covering algorithm. We give definitions that are conducive to the design of Algo. 1 below.

Definition 1: (**Battery-Feasible Sensor Nodes.**) Given that the robot is currently located at node $i \in V$ and with battery B, the set of sensor nodes U that have not been visited by the robot, its *battery-feasible sensor nodes*, denoted as $\mathcal{F}(i,B,U)$, is the set of sensor nodes that the robot has not visited and that it has sufficient battery to travel to and then return to s. $\mathcal{F}(i,B) = \{j | \big(c(i,j) + c(j,s)\big) \times \mu \leq B \wedge j \in V_s \wedge U\}$. □

Definition 2: (**Prize at a Sensor Node.**) Given a sensor node $i \in V_s$, its available *prize*, denoted as p_i, is all the data packets that can be collected by the robot when it visits i. I.e., $p_i = \sum_{j \in N_i \cup \{i\}} d_j^c$, where N_i is i's 1-covered nodes, and d_j^c is the current number of data packets available at j. Initially, $d_j^c = d_j$, and it becomes 0 when the robot collects j's packets. We denote a node i's *initial prize* as $p_i^o = \sum_{j \in N_i \cup \{i\}} d_j$. □

Definition 3: (**Covered Prize-Cost Ratio of a Battery-Feasible Sensor Node.**) Given the robot's current location i, for a battery-feasible node $k \in \mathcal{F}(i,B,U)$, its *covered prize-cost ratio*, denoted as $pcr(i,k)$, is the ratio between the prize available at k and the battery consumption of the robot moving from i to k. That is, $pcr(i,k) = \frac{p_k}{c(i,k) \times \mu}$. □

Definition 4: (**2-Covered Nodes.**) Given a node j, its 2-covered nodes are the 1-covered nodes of any 1-covered node of j that are not 1-covered nodes of j, denoted as N_j^2. That is, $N_j^2 = \{i | i \in N_k^1 \wedge k \in N_j^1 \wedge i \notin N_j^1\}$. □

Greedy Algorithm. Algo. 1 works as follows. First, all variables related to the robot's data-covering tour are initialized (line 1). It computes the initial prizes of all sensor nodes (lines 2-5) and then takes place in rounds. In each round, located at current node i, the robot visits a battery-feasible node j with the largest covered prize-cost ratio and updates all the route-related information (lines 7-10). It then collects j's prize p_j by collecting packets from j and its 1-covered nodes and prizes of all the sensor nodes involved (lines 11-31), illustrated next. This continues until it can no longer find a battery-feasible node, at which point it returns to s and outputs the data-covering tour, the

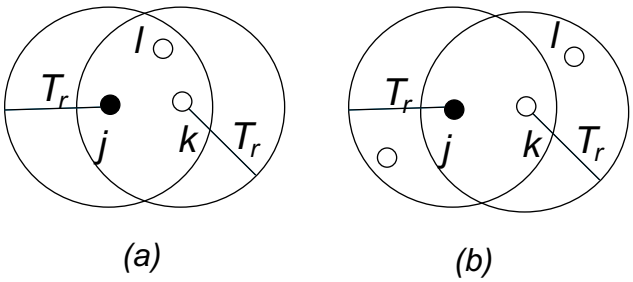
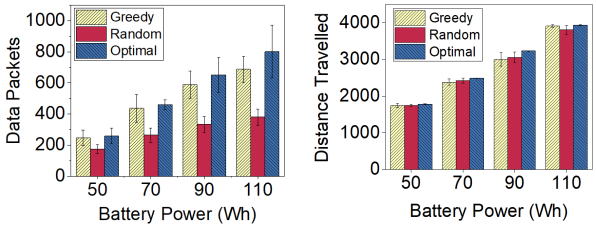


Figure 2. Prize-updating when node j is visited, where k is j's 1-covered node. (a) l is a 1-covered node of both j and k. (b) l is k's 1-covered node and j's 2-covered node.



(a)Collected Packets. (b)Traveled Distance.

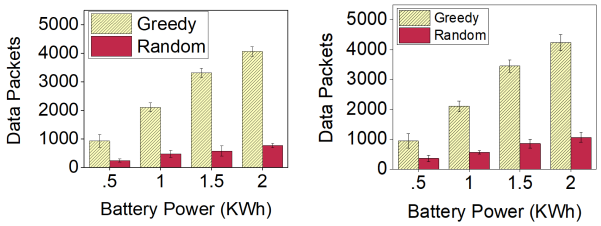
Figure 3. Comparing Greedy, Random, and ILP small-scale RSNs.

collected packets, and the energy cost of the robot on this route (lines 34-35). In Algo. 1, the robot visits at most $n = |V_s|$ nodes. At each node j, it updates the prizes by checking all of its 1-covered and 2-covered nodes, which is $O(n^2)$. Its time complexity is $O(n^3)$. Fig. 2 shows the details of the prize-updating process.

Random Covering Algorithm. We propose a random covering algorithm. The only difference between Greedy and Random is line 7 in Algo. 1, wherein Greedy chooses a feasible node with the largest prize-cost ratio to move to, while in Random, it instead randomly chooses one.

5. Performance Evaluation

Experiment Setup. We compare ILP(A) (**ILP**), greedy algorithm viz. Algo. 1 (**Greedy**), and random algorithm (**Random**). We use CPLEX [10] for ILP computation. We use small RSNs of $1000\text{m} \times 1000\text{m}$, where 20 sensor nodes are randomly placed when the ILP optimal solutions are computed, and large RSNs of $10,000\text{m} \times 10,000\text{m}$, where 100 sensor nodes are randomly placed. In either case, the depot s is at (0, 0) of the RSN. Each sensor generates a random number of data packets in [0, 100], each of 1024B. We set the mobility energy coefficient μ as 100 Joules/m following [6]. Each data point in our plots is an average of ten runs, for which a different RSN instance is constructed and applied to all the algorithms for fair comparison. The error bars indicate 95% confidence intervals. We write our simulator in Java on Windows 11 with AMD Ryzen 5 4000 Series 6-Core and 24GB of DDR4 Memory.



(a)$T_r = 150$m. (b)$T_r = 300$m.

Figure 4. Comparing Greedy and Random in large-scale RSNs.

Comparison in Small RSNs. Fig. 3 compares the three algorithms in small RSNs with the initial battery $\mathcal{E}$ of the robot varied from 50Wh to 110Wh. With a mobility coefficient of $\mu = 100$ Joules/m, this battery range means the robot can travel a distance between 1800m and 3960m. Fig. 3(a) shows that with the increase of battery power, more packets are collected for all the algorithms. Being an optimal solution, ILP always collects the most packets. Greedy performs very close to ILP, though, collecting the number of packets within 4.3% to 14.2% of ILP. Besides, Greedy significantly outperforms Random by collecting between 41.9% and to 80.1% more packets. Fig. 3(b) shows the distances traveled by the robot, showing that in all the test cases, battery power is a constraint limiting the traveling distance of the robot in all the algorithms. Table 1 shows the execution time of different algorithms w.r.t. $\mathcal{E}$. The execution time of both Greedy and Random is one order of magnitude smaller than that of ILP, while Random takes a bit less time.

Comparison in Large-scale RSNs. Fig. 4 compares the Greedy and Random in large-scale RSNs of $10,000\text{m} \times 10,000\text{m}$. Fig. 4(a) shows the data packets collected by both algorithms when T_r is 150m. Greedy collects up to five times more packets than Random, showing that Greedy is more effective. Fig. 4(b) increases the T_r to 300m and shows that Greedy outperforms Random by up to four times. Both algorithms collect almost the same amounts of packets with different T_r. This is because the networks are very sparse, meaning not a lot of extra nodes are covered by the increase in transmission range.

6. Conclusions

We propose a new algorithmic framework called budget-constrained covering salesman problem (i.e., BC-CSP).

Table 1. Execution Time (ms) of Different Algorithms.

Battery power $\mathcal{E}$ (Wh)	Greedy	Random	ILP
50	36	23	256
70	61	18	461
90	86	28	769
110	111	37	906

BC-CSP is inspired by robotic sensor networks (RSNs), wherein battery-constrained mobile robots, including drones and UAVs, are dispatched to collect sensing data or maintain the network in many sensing applications. We design algorithmic solutions to solve the BC-CSP, including ILP, greedy, and random heuristic algorithms. We show that greedy outperforms the random while performing close with the ILP in terms of collecting packets while both greedy and random are more time-efficient than ILP.

Acknowledgment

This work was supported by the NSF Grants CNS-2137791, HRD-1834620, and CNS-2240517.

References

[1] Ghosh P, Gasparri A, Jin J, Krishnamachari B. Robotic Wireless Sensor Networks. Springer International Publishing, 2019; 545–595.

[2] Li H, Savkin AV. An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments. Robotics and Computer Integrated Manufacturing 2018;54:65–82.

[3] Mahjoub W, Nakkach C, Ezzedine T. Design of autonomous wireless sensor network using mobile robots for intrusion detection and border surveillance. In Proc. of International Wireless Communications and Mobile Computing (IWCMC). 2023; .

[4] Wichmann A, Korkmaz T, Tosun AS. Robot control strategies for task allocation with connectivity constraints in wireless sensor and robot networks. IEEE Transactions on Mobile Computing 2018;17(6):1429–1441.

[5] Khan FMNU, Rasul MG, Sayem ASM, Mandal N. Maximizing energy density of lithium-ion batteries for electric vehicles: A critical review. Energy Reports 2023;9:11–21.

[6] Xiao X, Whittaker WL. Energy considerations for wheeled mobile robots operating on a single battery discharge. Technical Report CMU-RI-TR-14-16, Pittsburgh, PA, August 2014.

[7] Heinzelman W, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks. In Proc. of HICSS 2000; .

[8] Current JR, Schilling DA. The covering salesman problem. Transportation science 1989;23:208–213.

[9] Miller–tucker–zemlin (mtz) subtour elimination constraint. Https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html.

[10] Ibm cplex optimizer. Https://www.ibm.com/analytics/cplex-optimizer.

Address for correspondence:

Christopher Beauchamp
Department of Computer Science, California State University Dominguez Hills
cbeauchamp2@toromail.csudh.edu

Algorithm 1 Greedy Covering Algorithm.

Input: A mobility graph $G(V, E)$, d_i, T_r, μ, $\mathcal{E}$, and depot s,

Output: A data-covering tour R, P_R, and C_R.

Notations: R: the data-covering tour, starting from s;
C_R: the energy cost of the robot on R, initially zero;
P_R: the prizes collected on R, initially zero;
U: the set of unvisited sensor nodes, initially V_s;
i: the node where the robot is located currently;
A_j: nodes whose packets are collected when j is visited;
B: current battery power of the robot, initially $\mathcal{E}$;

$i = s$, $R = \{s\}$, $C_R = P_R = 0$, $U = V_s$, $B = \mathcal{E}$;
for (each $i \in V_s$) **do**
$N_i^1 = \{j | j \in V_s \wedge c(i,j) \leq T_r \wedge j \neq i\}$;
$p_i = p_i^0 = \sum_{j \in N_i^1 \cup \{i\}} d_j$;
end for
// if there are still battery-feasible nodes for the robot
while $(\mathcal{F}(i, B, U) \neq \phi)$ **do**
$j = \text{argmax}_{k \in \mathcal{F}(i,B,U)} pcr(i,k)$;
$R = R \cup \{j\}$, $P_R = P_R + p_j$;
$C_R = C_R + c(i,j) \times \mu$;
$B = B - c(i,j) \times \mu$, $U = U - \{j\}$;
$A_j = \phi$ (empty set), $p_j = 0$;
if $(d_j \neq 0)$ **then**
$d_j = 0$;
$A_j = \{j\}$;
end if
for (each $k \in N_j^1$) **do**
if $(d_k \neq 0)$ **then**
$p_k = p_k - d_k, d_k = 0$; $A_j = A_j \cup \{k\}$;
end if
end for
for (each $k \in N_j^1$) **do**
for (each $l \in N_k^1$) **do**
if $(l \in A_j)$ **then**
$p_k = p_k - d_l$;
else
if $(l \in N_j^2 \wedge k \in A_j)$ **then**
$p_l = p_l - d_k$;
end if
end if
end for
end for
$i = j$;
end while
$R = R \cup \{s\}$, $C_R = C_R + c(i,s) \times \mu$;
return R, P_R, C_R.