Les Cahiers du GERAD ISSN: 0711–2440

A heuristic method to solve an assignment problem using a random walk approximation

L. Monteiro, H. Tremblay, S. Séguin

G–2024–48

August 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A heuristic method to solve an assignment problem using a random walk approximation

Léo Monteiro [a, b]

Hugo Tremblay [a]

Sara Séguin [a, b]

[a] *Université du Québec à Chicoutimi, Saguenay (Qc), Canada, G7H 2B1*

[b] *GERAD, Montréal (Qc), Canada, H3T 1J4*

lmonteiro@etu.uqac.ca
h7trembl@uqac.ca
sara.seguin@uqac.ca

August 2024
Les Cahiers du GERAD
G–2024–48

Abstract : Over the past years, Robotic Process Automation (RPA) has emerged as a significant tool to enhance productivity across various industries by automating repetitive tasks performed on computer user interfaces, thereby reducing error rates. In this paper, the RPA problem is addressed as an unbounded assignment problem. Specifically, the objective of the problem is to assign robots to transactions that must be completed at specific time periods, while minimizing the total number of robots required. The problem is solved using a bipartite graph representation and a random walk approximation, which defines an ordering of the transactions and periods in order to determine a valid assignment. The heuristic is evaluated on a real data set from a financial institution and compared to previous results obtained on generated data. The results obtained with the random walk approximation heuristics on the real data set is optimal in terms of number of robots.

Keywords: Random walks, graph theory, effective resistance distance, robotic process automation, unbounded assignment problem

Acknowledgements: The authors would like to thank Guillaume Routhier from CGI for providing insight and data for this project.

Notation

The following notation is used throughout the paper:

T	set of transactions types
P	set of periods (including added periods)
P_I	set of initial periods of the problem
N_P	number of initial periods of the problem
R_p	set of robots assigned to period $p \in P$
v_t	volume of a transaction of type $t \in T$
d_t	processing time of a transaction of type $t \in T$
l_p	duration of period $p \in P$
w_{pt}	boolean variable denoting whether transaction type $t \in T$ can be processed at period $p \in P$
n_p	number of robots required at period $p \in P$
x_{pt}	number of transactions of type $t \in T$ treated at period $p \in P$
y_{rpt}	number of transactions of type $t \in T$ treated at period $p \in P$ by robot $r \in R_p$
N	upper bound for the number of robots
α_i	parameters for the random walk approximation bounds, for $i \in [1, 2]$

1 Introduction

Robotic Process Automation (RPA) is an advanced technology tool developed to mimic human interactions on user interface computer systems to automate repetitive task with a lower rate of errors [16]. The RPA tools are software licences, where each licence is called "robots" in this context. The RPA is focused on tasks completed on a user interface and completes actions such as identifying applications, copy and paste data, opening e-mails, filling and sending forms [10]. Moreover, not only the RPA process is faster than humans, but the risk of error is also lower [13]. Over half of European companies aimed to implement RPA to automate a minimum of 10 processes by 2020 [5]. In [16] the authors mention that commercial vendors are noticing an increase in the need for RPA tools as it is considered a quick way to have a high return on investment. However, the cost of each robot license is expensive. The RPA problem studied in this paper is an unbounded assignment problem. The goal of the problem is to model a transactions set that can only be achieved at specific periods in time. The goal of the problem is to minimize the number of robots required to complete all the transaction types while respecting their periods constraints. Each transaction type have a volume and a processing time, and each period a length. Minimizing the number of robots on such a problem have been studied before, using Integer Linear Programming (ILP) [11], computing upper bounds using heuristics [12], using network flows [1] and using graph properties with the effective resistance of a graph [15]. More details are presented in Section 2.

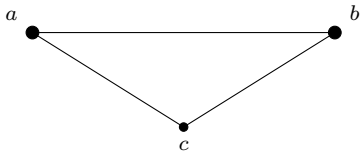
Our contribution with this paper is twofold : First, we use a random walk approximation approach [3] to compute the effective resistance of the graph. Then, the ordering heuristic presented in [15] is applied to solve the RPA problem. Second, we compare the approximation method, obtained with random walk, and the effective resistance method on 50 randomly generated test cases similar to a real-life case problem. Using the random walk approximation to solve the RPA problem returns the optimal number of robots for a specific real case problem.

This paper is organized as follows : Section 2 presents the previous work done on the RPA problem, the effective resistance and random walk approximation concepts. Section 3 presents the model and the algorithm used to solve the RPA problem using graph theory and the random walk approximation. Section 4 is divided in two parts. We first present the real-life case problem and its solution with the ILP formulation, the effective resistance heuristics and our method using the random walk approximation. We then introduce 50 randomly generated problem and compare the results between effective resistance and the random walk approximation. Section 5 opens a discussion on how to improve the complexity of the algorithms presented in the paper and, finally, Section 6 concludes the paper.

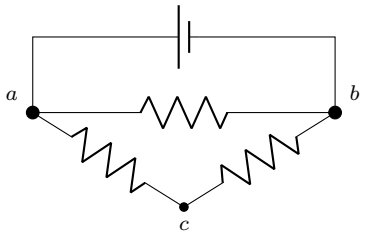
2 Previous work

An optimal solution to the RPA problem presented in Section 4 is obtained with a linear integer programming formulation. Nevertheless, in [12] the problem has been proven to be NP-Hard. The NP-hardness of the problem justifies the need for faster heuristics. In this paper, we use a random walk approximation method [3] in order to obtain an ordering of the edges of a bipartite graph. The random walk approximation of the effective resistance converges to the exact effective resistance value as the number of iterations grows. Remark that the effective resistance was used previously in [15] to solve the RPA problem, but was computed exactly using a different method as the one presented in this paper.

As its name suggests, the effective resistance is derived from the resistance in electrical circuits. Studying electrical circuits under the spectrum of graph theory was first introduced by Kirchhoff's analysis in the 19^{th} century [7]. In 1993, D.J. Klein defined a new measure of distance between two nodes in a graph, the effective resistance [8]. Usually the distance between two nodes is defined as the shortest path between them, meaning counting the edges, or taking the sum of their weight. Now, consider a graph as an electrical network where each edge is a resistor and two nodes a and b are connected by an electrical source (e.g. a battery). Then, the effective resistance R_{ab} between a and b is simply the resistance of the associated circuit. Figure 1 shows an example of a graph and its associated circuit. Recall from Kirchhoff's current laws that two edges (resistors), r_1 and r_2, connected in series can be replaced with one edge by adding the resistance of two resistors such that the new edge is equal to $r_1 + r_2$ [3]. Similarly, two edges (resistors), r_1 and r_2, connected in parallel can be replaced with one edge with the new resistance $\left(r_1^{-1} + r_2^{-1}\right)^{-1}$ [3]. Often in graphs, the measure of distance refers to the shortest path. For example, in Figure 1a the shortest is distance $\delta_{ab} = 1$ taking the existing edge (a, b) as the shortest path. In Figure 1b the effective resistance $R_{ab} = \frac{2}{3}$ using the parallel calculation defined above with $r_1 = 1$ (resistance of edge $\Omega_{ab} = 1$) and $r_2 = 2$ (resistance of edges $\Omega_{ac} + \Omega_{ca} = 1 + 1 = 2$). The effective resistance of a graph is equal to the sum of the resistance of each distinct pair of vertices in that graph [8]. In an undirected graph that is $R_G = \frac{1}{2} \sum_{i \in V} \sum_{j \in V} R_{ij}$. If the graph is not connected, some pairs of vertices do not have an effective resistance defined, just like they do not have a shortest path defined.



(a) Usual view of a graph. Each edge has a weight of 1.



(b) Graph viewed as a circuit. Each resistor has a resistance of $\Omega = 1$

Figure 1: Comparison of distance and effective resistance between two nodes under an usual graph view and a circuit view.

It is possible to compute the effective resistance using the adjacency matrix of the graph. Specifically, given an undirected graph G, let A be its adjacency matrix, Δ its degree matrix and $Q = \Delta - A$ its Laplacian matrix [17]. In section 3 and in Section 4.2 we use Theorem 2.1 of [3] to compute the effective resistance R_{ab} between two nodes a and b. Let a and b represent both their nodes and their indexes in V, such that $1 \leq a \leq |V|$ with $a \in V$ and $1 \leq b \leq |V|$ with $b \in V$, the authors give the following equation :

$$R_{ab} = (e_a - e_b)^T Q^{-1} (e_a - e_b) \tag{1}$$

where Q^{-1} is the pseudo inverse of the Laplacian matrix, e_a and e_b are vectors of size $|V|$ with each entry equal to 0 except at index a and b, both respectively equal to 1 [3]. Remark that Equation (1) computes the effective resistance differently from the methodology used in [15], which uses the reduced

Laplacian matrix and Lyapunov equations. In this paper, we develop a modified version of the heuristics presented in [15] based on an approximation of the effective resistance based on random walks [3]. Based on Chandra & al. work [2], the authors of [3] have rewritten the following equation for defining the effective resistance using random walk algorithm properties:

$$R_{ab} = \frac{1}{2|E|}(\mathbf{E}(T_{ab}) + \mathbf{E}(T_{ba})) \tag{2}$$

where T_{ab} represents the number of edges to get to vertex b starting in vertex a using a random walk algorithm on the graph and $\mathbf{E}$ is the expected value. From Equation (2), the authors deduce the following formula to compute R_{ab}, for all $a \neq b$, using Markov chain properties :

$$R_{ab} = \sum_{t=0}^{\infty} {}_a p_{bb}^{(t)} \frac{1}{\delta_b} \tag{3}$$

Let us break down Equation (3) : $p_{ab}^{(t)}$ denotes t-th power of the transition matrix p, where the element at the b-th row and b-th column is extracted. The prefix ${}_a$ preceding the transition matrix element represents a set of indices, indicating that the a-th column of the transaction matrix P is entirely set to zeroes. Note that in Equations (4) and (5) below, one of the prefix set is ${}_{a,b}$ meaning that both the a-th and the b-th column are replaced with zeroes. Finally, δ_b denotes the degree of the node b. For the details on how the Equation (4) works, we refer to Section 2.3 of [3]. However, since infinite sums are not computable in practice, the authors also provide the following bounds for the effective resistance:

$$L_{ab} = \frac{1}{\delta_b} \sum_{t=0}^{\alpha_1} {}_a p_{bb}^{(t)} \leq R_{ab} \leq \frac{1}{\delta_a \sum_{t=0}^{\alpha_2} \sum_{k=1}^{|V|} {}_{a,b} p_{ak}^{(t)} p_{kb}} = U_{ab} \tag{4}$$

where L_{ab} is the lower bound and U_{ab} is the upper bound. From these bounds, the authors obtain the following approximation for the effective resistance :

$$R_{ab} \approx \frac{1}{2} \left(\frac{1}{\delta_b} \sum_{t=0}^{\alpha_1} {}_a p_{bb}^{(t)} + \frac{1}{\delta_a \sum_{t=0}^{\alpha_2} \sum_{k=1}^{|V|} {}_{a,b} p_{ak}^{(t)} p_{kb}} \right) = W_{ab} \tag{5}$$

The exact approximation of the effective resistance is $R_{ab} = W_{ab} \pm \epsilon$, with W_{ab} being the random walk approximation and $\epsilon = \frac{1}{2}(U_{ab} - L_{ab})$, where U_{ab} and L_{ab} are, respectively, the upper bound and lower bound of Equation (4).

3 Mathematical model and algorithm

It is important to note that the model and algorithm defined in this Section is not our direct contribution, and both the model and algorithm are taken from [15]. However, as mentioned in the introduction (Section 1), the contribution of this paper is to compute an approximation of the effective resistance using random walk properties. Instead of effective resistance itself, the random walk approximation is computed for the assignment problem Algorithm (Figure 3). A comparison of the results between the two methods, along with an ILP formulation to have optimal solution as reference, is presented in Section 4.

3.1 Mathematical model

Let G be a weighted bipartite graph, where $G = (T \cup P, E, w)$. The set of nodes T represents the transaction types, the set of nodes P represents the periods, and an edge $(t, p) \in E$ if and only if a

transaction t can be processed at a period p. The weight w of an edge defines how many transactions t are done at a period p and which robot is doing it. To determine how many robots are required to solve the RPA assignment problem or to know which robot is doing a transaction, the set of periods P is used. The operation of adding a robot to G, as defined in [15], is to create a disjoint copy P' from the initial set of periods P_I and adding the edge (t_i, p'_j) if $(t_i, p_j) \in E$ to G, for all $t_i \in T$, $p'_j \in P'$ and $p_j \in P_I$. After creating a copy of the set P_I and adding the edges, the set of periods is updated such that $P = P \cup P'$. Let N_P be the number of initial periods defined by the problem, we can now define R as the set of robots, and the total number of robots $|R| = \frac{|P|}{N_P}$. More specifically, a task $t_i \in T$ done at a period $p_j \in P$ defined by the edge (t_i, p_j) is done by the robot $r_l = \left\lceil \frac{j}{N_P} \right\rceil$, where $r_l \in R$ and $1 \leq j \leq |P|$. This graph formulation contains all the relevant information for the problem and its solution: the assignment structure of the problem, the total number of robots required, the details of the assignment to the robots with the volumes completed for a transaction type and the period during which it was completed. Figure 2 gives an example of a RPA assignment problem, the processing time of a transaction type d_t, the volume needed for a transaction type v_t, the length of a period l_p.

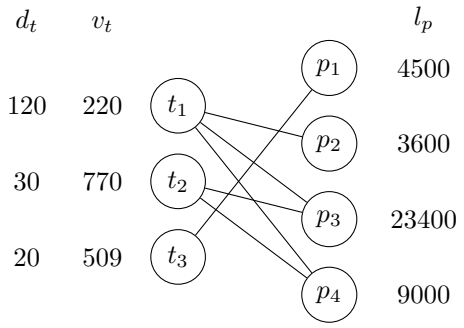


Figure 2: Graph example of the assignment RPA problem.

3.2 Algorithm

In order to solve the RPA problem, each period is filled with the maximum volume of transaction types it can hold. When no more transactions can be assigned to periods, a copy of the initial period set is made (i.e. adding a new robot) if necessary and this process continues until all the volume of transactions is completed [15]. To choose which transaction type and period to match first, we use the following algorithm, which is a modified version of the algorithm presented in [15] where the effective resistance is approximated using Equation (5):

1. Compute the random walk approximation W_{tp}, as defined in Equation (5), for all edges (t, p) where $t \in T$ and $p \in P$;
2. Sorts edges by decreasing approximation values;
3. Successively load each edge with the maximum possible volume of a transaction type;
4. If the assignment is valid, finish the algorithm. Otherwise, create a copy of P (add a new robot) and loop back to Step 1;

Figure 3: Algorithm to solve the RPA assignment problem through bipartite graph, using the random walk approximation, based on [15]

The program is implemented in Python language [18], using the Rustworkx library [14] for efficient graph implementation and computation, NumPy library [4] for matrix manipulation, Pandas library [9] and Matplotlib library [6] for exporting and showing the data in graphs presented in Section 4. It is important to highlight that both the effective resistance and the random walk approximation are not defined between nodes not connected by a path. It would mean dividing by 0 in the computation of the upper bound in Equation (4). For example, in Figure 2 the approximation $W_{t_1 p_1}$ is not defined. It is

however not an issue since the algorithm only requires computing the random walk approximation on edges of the graph, meaning that the path always exists between the two incident nodes of that edge. Figure 4 shows a solution to the small problem presented above using the random walk approximation, with parameters $\alpha_1 = \alpha_2 = 5$. Table 1 presents the state of the graph at the very first loop of the algorithm presented in Figure 3, where one robot is available. It shows a comparison between the effective resistance values, computed with Equation (1), and the random walk approximation with three different cases of parameters : $\alpha_1 = \alpha_2 = 2$, $\alpha_1 = \alpha_2 = 5$, and $\alpha_1 = \alpha_2 = 10$. On that small example, one can see the values of the approximation converges towards the effective resistance as the parameters α_1 and α_2 grow. The only exception is for the edges (t_2, p_3) and (t_2, p_4), where the approximation $\alpha_1 = \alpha_2 = 2$ gives closer result to the effective resistance result than the other approximation. This is a lucky coincidence, since the error ϵ for those specific edges is 0.25, meaning that the value for the effective resistance could be anything within the range $[0.5, 1]$.

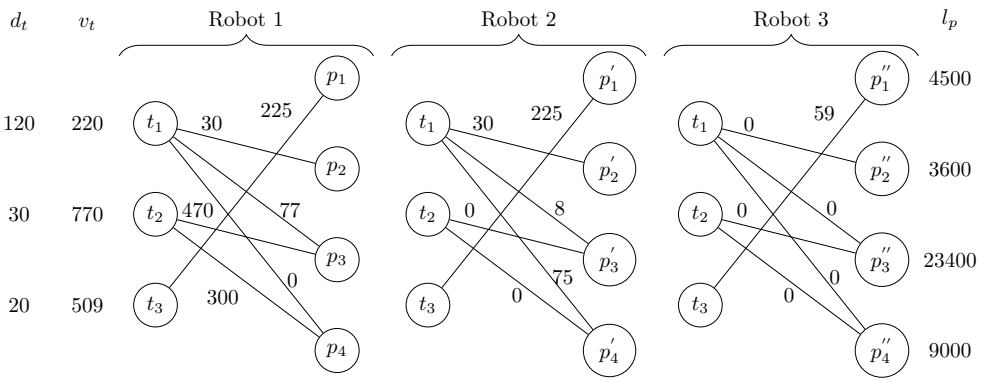


Figure 4: Valid assignment for the RPA problem. Three graphs are represented for readability, however the nodes t_1, t_2, t_3 are the same ones for each robot. To highlight which periods were added, we are using the notation $p_1', p_2', \cdots, p_3'', p_4''$, but in reality for the algorithm $p_1' = p_5, p_2' = p_6, \cdots, p_3'' = p_{11}, p_4'' = p_{12}$. For example, the node t_3 has a degree of 3 and is connected to the nodes $\{p_1, p_5 = p_1', p_9 = p_1''\}$. The solution is using the random walk approximation, with $\alpha_1 = \alpha_2 = 5$ as parameters, to sort the edges.

Table 1: Comparing effective resistance value and random walk approximation, while increasing α_1 and α_2 parameters, for each edges in Figure 4

Edges	Effective Resistance	Approx. $\alpha_1 = \alpha_2 = 2$	Approx. $\alpha_1 = \alpha_2 = 5$	Approx. $\alpha_1 = \alpha_2 = 10$
(t_1, p_2)	1	0.666667	0.768519	0.839249
(t_1, p_3)	0.75	0, 666667	0.679563	0.721576
(t_1, p_4)	0.75	0, 666667	0.679563	0.721576
(t_2, p_3)	0, 75	0, 75	0.738542	0.742389
(t_2, p_4)	0, 75	0, 75	0.738542	0.742389
(t_3, p_1)	1	1	1	1

4 Results

Section 4.1 presents comparisons between the ILP formulation, the effective resistance heuristics and the random walk approximation heuristics. In subsection 4.2 the effective resistance heuristics and the random walk approximation heuristics are compared on 50 randomly generated problems.

4.1 Real data set

In this Section, the study is based on real data provided by a financial institution. The results are compared between the effective resistance of [15] and the random walk approximation. The optimal result of the problem, computed with an ILP model, is presented in 4.1.1. The data used in this paper are the same one as in [15]. The data, provided by a financial institution, are defined with 12 different

transactions type, each having a different volume and processing time (Table 2). Table 3 presents the 8 periods of the problem, their length and their associated transactions types.

Table 2: Real life data case defining transaction types with their processing time, volume.

Type	1	2	3	4	5	6	7	8	9	10	11	12
Processing time	120	30	20	15	90	360	60	50	1320	150	480	52
Volume	220	770	509	13750	450	210	110	375	95	150	235	76

Table 3: Real life data case defining the period duration and allowed type of transactions. On the last row, the notation $[a, b]$, for $a, b \in [1, 12]$ means that all the transaction types between a and b are included in the period.

Periods	1	2	3	4	5	6	7	8
Period length (in seconds)	900	3600	3600	32400	7200	3600	3600	7200
Associated transaction type	3	3, 5	1, [4, 6], [8, 12]	1, 2, [4, 12]	1, [4, 12]	1, 4, 5, 7, 10, 11	1, 4, 5, 10, 11	4, 5, 10

The results obtained with the effective resistance heuristic proposed in [15] are compared with the random walk approximation algorithm (Figure 3). For the effective resistance, the results are shown in Table 4. The solution requires 12 robots over 8 periods. In Table 5 the solution, using the random walk approximation algorithm, gives a solution using 11 robots over 8 periods. The parameters $\alpha_1 = \alpha_2 = 5$ are set to compute the random walk approximation. The result is optimal for the number of robots to solve the problem but not over the periods, since the ILP model described in the Section 4.1.1 found a solution using only 7 periods. For this specific case, the random walk approximation gives a better result than the exact effective resistance.

Table 4: Effective resistance heuristic results for each robot. Each entry in the table lists the transaction type and the volume in parentheses.

Periods	Robot 1	Robot 2	Robot 3	Robot 4	Robot 5	Robot 6
1	3(45)	3(45)	3(45)	3(14)	×	×
2	5(40)	3(180)	3(180)	5(40)	5(40)	5(40)
3	8(72)	6(10)	6(10)	12(69)	9(2),11(2)	1(8),9(2)
4	2(772),4(2),8(184)	6(90)	6(79),9(3)	1(2),9(24),11(1)	1(2),9(24),11(1)	1(39),9(21)
5	7(50),8(84)	6(20)	4(2),6(1),8(4),9(5)	8(31),9(4),12(7)	1(1),9(5),11(1)	1(5),9(5)
6	7(60)	1(2),11(7)	1(30)	1(2),11(7)	1(2),11(7)	1(30)
7	1(2),11(7)	1(2),11(7)	1(30)	1(2),11(7)	1(2),11(7)	1(30)
8	4(480)	10(48)	10(48)	4(480)	5(80)	5(80)
Periods	Robot 7	Robot 8	Robot 9	Robot 10	Robot 11	Robot 12
1	×	×	×	×	×	×
2	5(40)	5(40)	×	×	×	×
3	1(2),11(7)	4(16),11(7)	4(240)	4(240)	4(240)	×
4	1(2),11(67)	4(656),11(47)	4(2160)	4(2160)	4(2160)	×
5	11(15)	11(15)	4(480)	4(480)	4(480)	×
6	1(2),11(7)	1(19),4(24),11(2)	4(240)	4(240)	4(240)	×
7	1(2),11(7)	1(2),11(7)	4(240)	4(240)	4(240)	×
8	10(48)	4(120),5(50),10(6)	4(480)	4(480)	4(480)	4(480)

4.1.1 Validation with a linear integer program

Since the data used in [11] is slightly different, two linear integer problems are defined in this Section in order to determine the transaction assignment to the robots. The first model minimizes the total number of robots and the second assigns the transactions to the robots.

Table 5: Random walk approximation results for each robot. Each entry in the table lists the transaction type and the volume in parentheses. Parameters $\alpha_1 = \alpha_2 = 5$ are set to compute the random walk approximation.

Periods	Robot 1	Robot 2	Robot 3	Robot 4	Robot 5	Robot 6
1	3(45)	3(45)	3(45)	×	×	×
2	3(180)	3(180)	3(14),5(36),9(2)	5(40)	5(40)	5(40)
3	12(69)	8(19),9(2)	8(19),9(24)	8(19),9(2)	1(2),6(5),8(31)	6(10)
4	2(770),8(1),9(7)	4(1),8(14),9(24)	4(1),8(14)	8(199),9(17)	6(90)	1(3),6(65),11(18)
5	4(3),7(50),8(23), 9(2),12(7)	8(12),9(5)	8(12),9(5)	8(12),9(5)	6(20)	6(20)
6	7(60)	1(2),11(7)	1(2),11(7)	1(2),11(7)	1(2),11(7)	1(2),11(7)
7	1(2),11(7)	1(2),11(7)	1(2),11(7)	1(2),11(7)	1(2),11(7)	1(2),11(7)
8	10(48)	10(48)	10(48)	4(420),10(6)	4(480)	4(480)
Periods	Robot 7	Robot 8	Robot 9	Robot 10	Robot 11	Robot 12
1	×	×	×	×	×	×
2	5(40)	5(40)	5(40)	5(40)	5(40)	×
3	1(2),11(7)	1(2), 11(7)	4(240)	4(240)	4(240)	×
4	1(2),11(67)	1(179), 4(496),11(1)	4(2160)	4(2160)	4(1349),5(94)	×
5	11(15)	11(15)	4(480)	4(480)	4(480)	×
6	1(2),11(7)	1(2), 11(7)	4(240)	4(240)	4(240)	×
7	1(2),11(7)	1(2), 11(7)	4(240)	4(240)	4(240)	×
8	4(480)	4(480)	4(480)	4(480)	4(480)	×

$$\min_{n_p, x_{pt}} \sum_{p \in P} n_p \tag{6}$$

s.t.

$$\sum_{p \in P} x_{pt} w_{pt} = v_t, \qquad \forall t \in T, \tag{7}$$

$$\sum_{t \in T} x_{pt} t_p \leq l_p n_p, \qquad \forall p \in P, \tag{8}$$

$$n_p \leq N, \qquad \forall p \in P, \tag{9}$$

$$x_{pt} \in \mathbb{N}_0^+, \qquad \forall p \in P, t \in T \tag{10}$$

$$\max_{y_{rpt}} \sum_{p \in P} \sum_{r \in R_p} \sum_{t \in T} y_{rpt} \tag{11}$$

s.t.

$$\sum_{r \in R_p} y_{rpt} = x_{pt}, \qquad \forall p \in P, t \in T, \tag{12}$$

$$\sum_{t \in T} y_{rpt} t_p \leq l_p, \qquad \forall p \in P, r \in R_p, \tag{13}$$

$$y_{rpt} \in \mathbb{N}_0^+, \qquad \forall p \in P, t \in T, r \in R_p \tag{14}$$

The objective function shown in Equation (6) minimizes the total number of robots required. Constraints (7) ensure that the volume of transactions is treated. Constraints (8) guarantee that the length of the periods are respected, multiplying the duration of the period by the number of robots used. The upper bound on the number of robots is given by (9). Finally, variables' domain is given by Equation (10). The result from this model is given as an input to the second model. More precisely, the number of transactions of each type treated at each period, as well as the number of robots required at each period are given as parameters. The objective function shown in Equation (11) maximizes the assignment, guaranteeing that all the transactions are treated. Constraints (12) assign the transactions types to the robots. Constraints (13) ensures that each robot completes the transactions given the length of the period. Finally, variables' domain is given in (14). Without adding a bound on the number of robots, the optimal solution requires 21 robots, which is not realistic unless we have an unlimited budget. By specifying bounds on the number of robots, we can find the actual optimal solution. Bounding by using a maximum of 11 robots gives the solution presented in Table 6, where the problem is infeasible using only 10 robots. For the problem defined in this Section, the optimal solution minimizing the number of robot required is 11 for 7 periods of work.

4.2 Experimental results on generated problems

In this Section, we compare the effective resistance algorithm [15] with the random walk approximation algorithm (Figure 3). We do not have the optimal result in consideration for those problems. The problems are randomly generated. Some constraints are defined to generate the parameters of the problem : the number of periods and transaction types are between 3 and 5, each transaction has 30% chances to be associated to a period, the period length is between 100 and 25000, the type processing

Table 6: Solution of the RPA problem using ILP formulation. Each entry in the table lists the transaction type and the volume in parentheses.

Periods	Robot 1	Robot 2	Robot 3	Robot 4	Robot 5	Robot 6
1	×	×	×	×	×	×
2	5(40)	5(40)	5(40)	3(180)	5(40)	5(40)
3	6(10)	6(10)	6(10)	6(10)	6(10)	6(10)
4	2(6),4(2148)	2(8),11(67)	4(16),6(5),9(23)	4(8),9(24),10(4)	4(4),10(142),11(23)	2(4),9(24),10(4)
5	1(60)	8(144)	4(480)	5(80)	8(144)	1(60)
6	4(240)	4(240)	4(240)	4(240)	4(240)	4(240)
7	4(240)	4(240)	4(240)	4(240)	4(240)	4(240)
8	4(480)	4(480)	4(480)	4(480)	4(480)	4(480)
Periods	Robot 7	Robot 8	Robot 9	Robot 10	Robot 11	Robot 12
1	×	×	×	×	×	×
2	5(40)	3(149), 5(6)	3(180)	5(28)	5(40)	×
3	6(10)	6(10)	6(10)	6(10)	6(10)	×
4	2(8),11(67)	6(2),9(24)	6(3),11(57),12(76)	6(90)	2(744),11(21)	×
5	1(60)	1(35),8(60)	1(5),7(110)	4(390),8(27)	4(144),5(56)	×
6	4(240)	4(240)	4(240)	4(240)	4(240)	×
7	4(240)	4(240)	4(240)	4(240)	4(240)	×
8	4(480)	4(480)	4(480)	4(480)	4(480)	×

time is between 10 and 150, and finally the volume of transaction is between 10 and 500. The method to compute the effective resistance for the problem is Equation (1). In Figure 5 and in Figure 6 both algorithms are compared on 50 generated problems. The random walk approximation parameters for the bounds are set to $\alpha_1 = \alpha_2 = 2$. Figure 5 compare the number of robots needed for a problem. We can see that in 8 problems out of 50 the random walk approximation is better, and in 2 cases out of the 50 the effective resistance is better. Figure 6a and Figure 6b are the time needed for a problem to be solver for the effective resistance and the random walk approximation, respectively. We observe that the random walk approximation takes much more times when the graph gets bigger because of the number of robots. In Section 5, we make a hypothesis on the possibility to reduce the time needed, closer to the effective resistance, for the random walk approximation to find its solution.

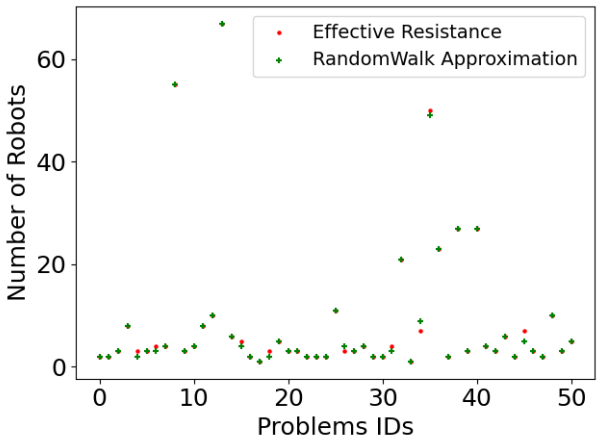
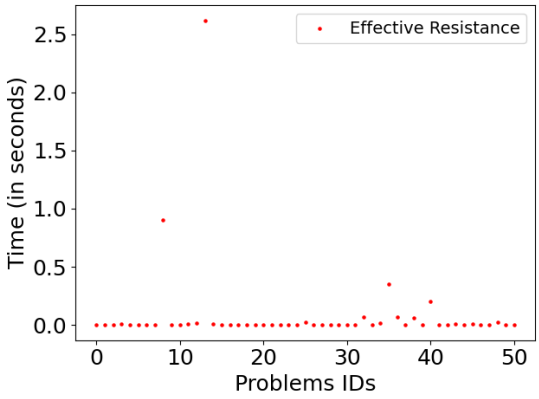
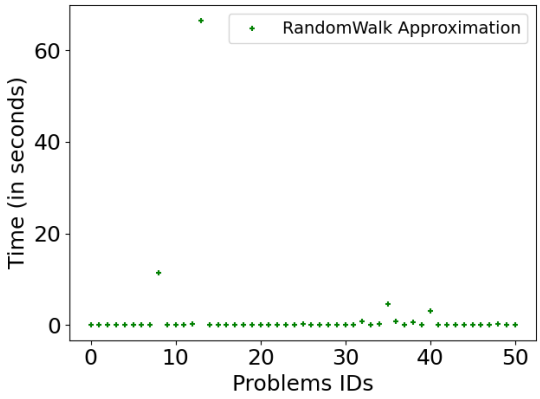


Figure 5: Comparing the random walk approximation and the effective resistance on 50 randomly generated problems. If only one element is observable given an x-axis position, it means that both methods give the same number of robots.



(a) Time required for each problem using the effective resistance



(b) Time required for each problem using the random walk approximation

Figure 6: Comparing the time required to compute random walk approximation and the effective resistance on 50 randomly generated problems.

5 Discussion

The results presented are very preliminary. On the real data case, the random walk approximation is optimal on the number of robots, so it would be interesting to compare the algorithm on bigger problems with more transaction types and more periods. Unfortunately, the approximation method is not faster to compute than the effective resistance due to the large number of matrix multiplications required. However, it might be possible to accelerate the computation by not recomputing the random walk approximation each loop. Indeed, after adding a robot, the initial periods are copied and added to the graph before computing a new approximation. By adding a copy of the set of periods, the structure of the graph is only slightly changed, since it has the same number of added nodes and a similar structure regarding the added edges. In the algorithm, it could be possible to keep the order of the sorted edges computed at Step 2 when the graph had no robot added, and keeping that order for the new edges coming with the new periods. On very small empirical sets this seems to be working, though it would require further exploration. Another interesting exploration would be to ignore the edges of previous iterations. By doing so, adding a new robot and making the graph bigger would only change the complexity of the algorithm by a constant, since we would not be redoing any calculation over the graph after the first sorting of the edges. The method would work on effective resistance and potentially speed up our implementation as well.

6 Conclusion

This paper offers new results on solving the RPA problem using heuristics. We are using the same methodology used in [15], but changing the method to sort the edges on their algorithm, using the random walk approximation instead of the effective resistances. The solution computed in this paper is optimal for the number of robot required using the random walk approximation. A comparison between the random walk approximation and the effective resistance on 50 randomly generated problems gives better results for the random walk approximation. However, it is important to note that, on a single real data case, the random walk approximation takes longer to compute. It would be interesting to further test both methods on larger problems. Other improvements could be made to the algorithm by taking into considerations weights on edges before computing either the effective resistance or the random walk approximation.

References

[1] I. Benkalaï, S. Séguin, H. Tremblay, and G. Glangine. Computing a lower bound for the solution of a robotic process automation (RPA) problem using network flows. 1:118–123, 2020.

[2] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In Proceedings of the twenty-first annual ACM symposium on Theory of computing, pages 574–586, 1989.

[3] W. Ellens, F. M. Spieksma, P. Van Mieghem, A. Jamakovic, and R. E. Kooij. Effective graph resistance. Linear algebra and its applications, 435(10):2491–2506, 2011.

[4] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. Nature, 585(7825):357–362, September 2020.

[5] P. Hofmann, C. Samp, and N. Urbach. Robotic process automation. Electronic markets, 30(1):99–106, 2020.

[6] J. D. Hunter. Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3):90–95, 2007.

[7] G. Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. Annalen der Physik, 148(12):497–508, 1847.

[8] D. J. Klein and M. Randic. Resistance distance. Journal of Mathematical Chemistry, 12:81–95, 1993.

[9] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[10] J. Ribeiro, R. Lima, T. Eckhardt, and S. Paiva. Robotic process automation and artificial intelligence in industry 4.0–a literature review. Procedia Computer Science, 181:51–58, 2021.

[11] S. Séguin and I. Benkalai. Robotic process automation (RPA) using an integer linear programming formulation. Cybernetics and systems, 51(4):357–369, 2020.

[12] S. Séguin, H. Tremblay, I. Benkalaï, D.-E. Perron-Chouinard, and X. Lebeuf. Minimizing the number of robots required for a robotic process automation (RPA) problem. Procedia Computer Science, 192:2689–2698, 2021.

[13] S.Parcells. The power of finance automation. Strategic Finance, 98(6):40, 2016.

[14] Matthew Treinish, Ivan Carvalho, Georgios Tsilimigkounakis, and Nahum Sá. rustworkx: A high-performance graph library for python. Journal of Open Source Software, 7(79):3968, 2022.

[15] H. Tremblay, S. Séguin, L.-A. Boily, V. Du Paul, and S. Lalancette. Robotic process automation (RPA) using a heuristic method and the effective resistance of a graph. Procedia Computer Science, 225:3388–3394, 2023.

[16] W.M.P. Van der Aalst, M. Bichler, and A. Heinzl. Robotic process automation, 2018.

[17] P. Van Mieghem. Graph spectra for complex networks. Cambridge University Press, 2023.

[18] Guido Van Rossum and Fred L. Drake. Python 3 Reference Manual. CreateSpace, Scotts Valley, CA, 2009.