Computing a lower bound for the solution of a Robotic Process Automation (RPA) problem using network flows

Imène Benkalaï
Dept. of computer science and mathematics
Université du Québec à Chicoutimi and GERAD
Saguenay, Canada
imene1_benkalai@uqac.ca

Sara Séguin
Dept. of computer science and mathematics
Université du Québec à Chicoutimi and GERAD
Saguenay, Canada
sara.seguin@uqac.ca

Hugo Tremblay
Dept. of computer science and mathematics
Université du Québec à Chicoutimi
Saguenay, Canada
hugo_tremblay2@uqac.ca

Geoffrey Glangine
Dept. of computer science and mathematics
Université du Québec à Chicoutimi
Saguenay, Canada
geoffrey.glangine1@uqac.ca

***Abstract*—Robotic process automation (RPA) helps companies reduce the time required to process tasks by using software or robots to mimic human actions on graphic interfaces. In this paper, the RPA problem is solved for a financial institution. A set of different types of financial transactions are to be processed with different processing times, volumes, market hours and clearance delays. In a previous work, a two-phase linear integer model was used to solve the problem on small instances. In this study, a network flow algorithm is used to compute a lower bound for the problem, thus reducing the computational time required to obtain a solution. The method is tested on a real case provided by a bank in North America and on synthetic test cases containing a greater number of transaction types. Results show that combining the computation of the lower bound with a linear integer model is faster and more practical.**

***Index Terms*—Robotic process automation, integer programming, network flows, optimization.**

I. Introduction

The world we live in is always going faster and presents ever more complicated challenges. Every sector of human activity is subject to a high competition between the different actors that interact within it. In this context, companies seek to achieve better productivity as well as a more efficient management of their resources which are usually scarce and expensive. This paper deals with the particular field of Robotic Process Automation (RPA). It allows companies to use robots instead of humans to perform repetitive tasks on graphic interfaces in a computer system. In the field of information technologies (IT), a robot is a software license. The robots are programmed to learn how to mimic human actions using machine learning and artificial intelligence algorithms [1]. The importance of this field is twofold. First, it helps reduce the time required to perform the tasks [2]. For example, since it is a software, a robot can be granted background-access to all the necessary resources on the computer to perform its task without wasting time waiting for a display. Also, unlike humans, robots are not prone to fatigue or exhaustion. They can therefore keep performing tasks without needing a break while avoiding human-made errors. However, RPA does not aim to replace humans but rather to help them process repetitive tasks, especially when they come in large volumes and with short processing times.

A lot of companies are switching to RPA to modernize their operating processes. For example, instead of outsourcing invoice processing, bookkeeping or data entry, RPA can help reduce costs by completing these tasks in-house [3], [4]. A case study presented in [5] showcases how RPA allows for better efficiency and faster processing times. A company was able to increase the number of treated cases by 20% by assigning robots rather than humans to back-office tasks. In another study that addresses RPA at the company Telefònica O2 [6], authors conclude that obtained RPA solution leads to a better consumer experience as well as a reduction of costs. The use of RPA is also of interest for insurance companies. A study presented in [7] shows that the use of RPA leads to a reduction of 300% in the size of teams assigned to process insurance claims. This is a good example of the fact that, despite its fast processing times and high efficiency, RPA still needs to be implemented in complement of existing teams.

As previously mentioned, RPA is particularly useful for companies that need to manage huge amounts of data, for example financial institutions. Indeed, in that context, it is common to have, on a daily basis, millions of transactions where the majority have very short processing times.

An analysis of the literature dedicated to RPA points out

GERAD, Group for Research in Decision Analysis, HEC-Montréal, Montréal, Canada.

that while it is undeniable that RPA is useful and leads to a better productivity, little information is provided on how to effectively implement RPA.

In [4], the problem is formulated as a two-phase integer mathematical program. The first phase aims to find the minimum number of robots required to complete all the tasks while in the second phase, the tasks are assigned to the robots. In that problem, there are several transaction types that need to be processed for a financial institution. Each type has a processing time, a volume *i.e.* a number of transactions, market operating hours and clearance dates *i.e.* a maximal deadline by which all transactions of that type need to be completed.

In this paper, two different approaches are compared. The first one is the two-phase integer programming approach presented in [4] and adapted to the problem at hand while the second one is based on network flow algorithms developed in graph theory. In the latter, the problem is modelled as a network in which the maximum flow is sought. This provides a lower bound that allows to model the problem in one phase and thus considerably reduce running times. Another advantage of the second approach is that it is adaptable to dynamic contexts.

Integer programming has been extensively studied in the field of operations research. Many different solution methods have been proposed throughout the literature such as branch-and-bound, cutting planes, branch-and-cut, etc. Each one of them attempts to tackle the complex nature of the problems modelled as integer programs by trying to approach the convex hull of the solution space as fast and as intelligently as possible. Many of those methods are now embedded in commercial solvers that allow users to efficiently compute solutions without having to code the methods from scratch. A good review of the modelling and applications of integer programming are presented in [8]. In this paper, the Xpress solver is used to solve the linear programs coded in AMPL.

In the second method, the problem is modelled as a network which is used to find a lower bound on the number of robots that are necessary to process all of the transactions. As for the rest of the field of graph theory, network flows received considerable attention from the scientific community since they were first formulated in 1954 to model a simplified Soviet railway traffic flow [9], [10]. One of the most famous algorithms designed to solve the maximum network flow problem is the Ford-Fulkerson method [11]. It is a greedy algorithm in which the main idea is to find so-called augmenting paths, i.e. a path from the source (start node) to the sink (end node) with available capacity on all edges. Then, a flow is sent along those paths and so on until no such path exists. A recent version of this algorithm was proposed by Boykov and Kolmogorov [12], [13]. Instead of computing, at each iteration, a shortest path from the source to the sink, it builds up two search trees : one for the source and one for the sink. Then, it uses a color-labelling system to find augmenting paths in a more efficient way. More details on network flows are provided in Section III.

The rest of the paper is organized as follows. The problem is presented in Section II along with the adaptation of the first solution method. The method to compute the lower bound on the number of robots is then detailed in Section III. The methodology and conducted experiments are summarized in Section IV. Finally, concluding remarks are presented in Section V.

NOTATION

The following notation is uses throughout the paper.

The sets are defined as follows:

- P: types of transactions.
- K: periods.
- R: robots.

The parameters are:

- t_p= time required to execute a transaction of type $p \in P$.
- v_p= volume of transactions for type $p \in P$.
- $w_{kp} = \begin{cases} 1, & \text{if transaction type } p \in P \text{ can be} \\ & \text{processed at period } k \in K \\ 0, & \text{otherwise} \end{cases}$
- N= minimal number of robots required.
- l_k= length of period $k \in K$.

The variables are the following:

- x_{rkp}= number of transactions of type $p \in P$ processed by robot r at period $k \in K$.
- n_k= number of robots required at period $k \in K$.
- y_k= number of robot start-ups at period $k \in K$.

II. THE TWO-PHASE OPTIMIZATION METHOD

This section presents the problem description and a two-phase optimization method derived from a previous work [14]. The numerical results presented in this paper are compared with this two-phase optimization method, therefore the reader can refer to [14] for more details.

A. Problem description

Data available to formulate the RPA problem for financial institutions are: types of transactions, processing times, volumes, market opening hours and clearance dates. The problem aims at finding the optimal number of robots and the exact assignation or schedule of transactions for each robot.

Each robot is a software that requires a licence that incurs a certain cost. The objective is thus to minimize the costs, i.e. the number of robots. In that context, the different types of transactions need to be scheduled in a way that ensures that all the transactions are processed within their market opening hours while respecting their clearance dates. In other words, the output of a solution method would be the number of required robots along with the schedule of transaction types which specifies the periods during which each transaction type is processed but also the robot(s) to which they are assigned.

B. Previous work

In a previous work [14], a two-phase integer linear program is developed to solve the problem. The first phase computes the total number of robots required at each period to process all of the transactions and the second phase is an assignment problem that determines the schedule of each robot. In [14], the startup and reconfiguration times of the robots are taken into consideration, but in this paper, they are ignored to assess the new methodology developed. In future works, these costs will be reconsidered. The solution methodology is explained below.

1) Phase I: In this phase, the minimal number of robots required to complete the transactions is computed using a linear integer program. The only difference with [14] is that the startup and reconfiguration costs are dropped. The objective function consists at minimizing the total number of robots. Constraints ensure that the required volume of transactions is treated and that the transactions are processed only during market opening hours. Consecutive time periods are obtained by ordering start and end dates for all tasks. The decision variables are the number of transactions of each type processed at each period, the number of required robots at each periods and the number of robot startups at each period.

2) Phase II: This phase is simply an assignment problem and uses as input the minimum number of required robots at each period obtained from Phase I to determine exactly which transactions are processed by which robot and at which period. Constraints ensure that the number of transactions processed at each period is equal to the Phase I solution. The decision variables are the number of transactions of each type executed by each robot at each period.

Although the running times of this two-phase method are rather short, it is worth mentioning that before starting Phase II, the data files require formatting, which is time consuming. Therefore, this study presents a more practical way of solving the problem which would come in handy, especially as the number of transaction types grows.

III. Computing the lower bound

By recasting the problem as a network flow and using a modified version of the Ford-Fulkerson algorithm, a lower bound to the solution of the RPA problem is obtained (see discussion in Section IV).

A. Network flow model

Using the notation of [15], let $G = (V, E)$ be a digraph with $V = \{s, t\} \cup P \cup K$ and

$$E = \{(s, p) \mid p \in P\} \cup \{(p, k) \mid w_{kp} = 1\} \cup \{(k, t) \mid k \in K\}.$$

Next, the capacity function is defined $\mathrm{cap} : E \longrightarrow \mathbb{R}^+$ where

$$\mathrm{cap}(e) = \begin{cases} v_p t_p & \text{if e is an arc of type (s,p) or (p,k)} \\ l_k & \text{otherwise} \end{cases}.$$

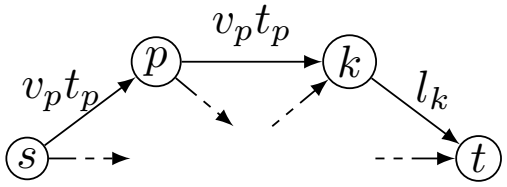

Fig. 1: The flow s-t-network G

Figure 1 illustrates the s-t-network G.

Now, let $f : E \longrightarrow \mathbb{R}^+$ be a feasible flow in G and (G, f) the associated network. Let us note that (G, f) is a linear relaxation of the two-phase integer program of Section II. First, the set $\mathrm{In}(v)$ of incoming arcs for vertices v corresponds to the boolean variables w_{kp}. Indeed, it is possible to process transaction p at period k if and only if the corresponding arc (p, k) exists in G. Also, if $f(p, k) \neq 0$, then transaction is at least partially processed at period k. Remark that since the outdegree of k is 1 and the periods are consecutive and non-overlapping, all transactions at period k are assumed to be treated sequentially by one robot (this assumption will be lifted later on).

Obviously, a max flow assignation doesn't necessarily corresponds to a valid integer solution. However, it follows from the RPA problem definition that if the maximum flow of G is less than $\sum_{e \in \mathrm{Out}(s)} \mathrm{cap}(e)$, then more than one robot is necessary to complete all transactions. The lower bound problem can thus be restated as follows:

Problem III.1. *Given a network (G, cap), find a maximum flow f^* such that all arcs $e \in \mathrm{Out}(s)$ are saturated.*

Of course, it is an easy task to find small examples in which problem III.1 is unsolvable, as Figure 2 illustrates.

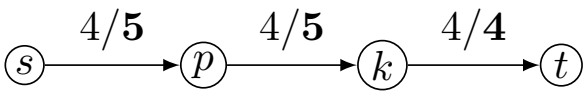

Fig. 2: A network (G, f) with no solution for problem III.1. Here, one transaction requiring 1 second to be completed must be executed 5 times in a single 4 seconds period.

The proposed solution is to augment graph G by iteratively adding arcs until the condition of problem III.1 is satisfied. First, the following proposition is proven to establish a base case for the proposed algorithm.

Proposition III.2. *Let (G, cap) be the network described in this section and f^* a maximum flow for this network. Then, either one robot is sufficient to treat all transactions or at least one period k is saturated.*

Proof. By construction, for any s-t-cut $\langle X, \overline{X} \rangle$ in G, we have $\langle \overline{X}, X \rangle = \emptyset$. Now, let $\langle X, \overline{X} \rangle$ be a minimal s-t-cut in G. Since $\mathrm{cap}(s, p) = \mathrm{cap}(p, k)$ and the number of outgoing arcs $\mathrm{Out}(p, k)$ is at least 1, $\forall\ p \in P$, then either $X = \{s\}$ or X contains a least one arc (k, t) for some $k \in K$.

If $X = \{s\}$, then by the Max-Flow Min-Cut Theorem [16], $\mathrm{val}(f^*) = \sum_{e\in \mathrm{Out}(s)} \mathrm{cap}(e) = \sum_{p\in P} v_p t_p$, i.e. all outgoing arcs of vertex s are saturated and one robot is sufficient to treat all transactions. Otherwise, X contains at least a saturated arc (k, t). This implies that one robot is not sufficient to complete all transactions at period k. □

Now, given the network (G, cap), the *augmented network* (G', cap) is defined where G' is a directed multigraph constructed from G by allowing the outdegree of vertices $k \in K$ to be greater than 1. The capacity function is extended accordingly by letting $\mathrm{cap}(e_i) = \mathrm{cap}(e_j)\ \forall\ e_i, e_j \in \mathrm{Out}(k)$. This effectively adds robots at a given period k. Furthermore, the *multiplicity of* (G', cap) is defined by

$$\mathrm{mult}(G', \mathrm{cap}) = \max\{\deg^+(k) \mid k \in K\}.$$

For example, (G, cap) is the only network with multiplicity 1.

From Proposition III.2, this obvious result is derived.

Corollary III.3. *Let (G', cap) be an augmented network with multiplicity $n \in \mathbb{N}$. If there exists a maximum flow f^* such that $f^* = \sum_{e\in \mathrm{Out}(s)} \mathrm{cap}(e)$, then at least n robots are required to complete all transactions.*

The following proposition ensures that such an augmented network always exists.

Proposition III.4. *Let (G, cap) be the network described in this section. Then, there exists an augmented network (G', cap) such that a maximum flow f^* saturates all arcs $e \in \mathrm{Out}(s)$.*

Proof. Let (G', cap) be the augmented network obtained from (G, cap) by adding arcs (k, t) until $\sum_{k\in K} \sum_{e\in \mathrm{Out}(k)} \mathrm{cap}(e) > \sum_{e\in \mathrm{Out}(s)} \mathrm{cap}(e)$. Then, the set X of a minimal cut $\langle X, \overline{X}\rangle$ cannot contain any vertex $k \in K$, otherwise $\langle \{s\}, V \setminus \{s\}\rangle$ would be a cut with a smaller capacity. By the Max-Flow Min-Cut theorem, all arcs $e \in \mathrm{Out}(s)$ are saturated. □

Problem III.1 is generalized to augmented networks.

Problem III.5 (Linear relaxation of the RPA problem in graph theoretic form)**.** *Given a network (G, cap), find an augmented network (G', cap) of minimal multiplicity such that a maximum flow f^* saturates all arcs $e \in \mathrm{Out}(s)$, i.e. $\langle \{s\}, V \setminus \{s\}\rangle$ is a minimal cut in G'.*

B. Computing the optimal assignation

The idea of the algorithm designed in this paper to solve Problem III.5 is to iteratively construct a sequence of augmented graphs of non-decreasing multiplicity. At each step, the algorithm checks if all arcs $e \in \mathrm{Out}(s)$ are saturated, in which case it stops. The pseudo-code is presented in Algorithm 1.

By Proposition III.4, Algorithm 1 always terminates in polynomial time. Indeed, let $a = (s, p)$ be an arc of maximum capacity. Then, in the worst case, it is required to add $\min_{k\in K} \left\lceil \frac{\mathrm{cap}\, a}{\mathrm{cap}\,(k,t)} \right\rceil$ arcs (this bound is not tight). Thus, the loop is executed $\mathcal{O}(|E|^2)$ times. Computing a maximum flow can also be done in polynomial time [17]. Finally, all other instructions are executed in linear time.

Algorithm 1 Solving Problem III.5

Input: A network (G, cap) with a maximum flow f^*.
Output: An augmented network (G', cap) with maximum flow f^* such that all arcs $e \in \mathrm{Out}(s)$ are saturated

```
function AUGMENT((G, cap) : network, f* : max flow) : (G', f*)
    while there exists a non-saturated arc e ∈ Out(s) do
        (k_i, t) ← the first saturated (k, t)-arc in lexicographical order
                                              ▷ e.g. (k_2, t) ≼ (k_5, t)
        Add a new arc (k_i, t) to G with the same capacity
        ▷ Since this effectively adds a new robot, one might as well add an arc to each period. However, doing so does not change the algorithm's complexity
        Compute the maximum flow of (G', cap)
    end while
    return (G', f*)
end function
```

IV. Methodology & Numerical results

This section presents the methodology followed to solve the problem and the numerical results obtained.

A. Methodology

As mentioned in Section III, a maximal flow algorithm, more specifically the Boykov-Kolmogorov algorithm [12], [13], is used to provide a lower bound on the number of robots. Since this bound is the minimal number of robots required to treat all of the transactions, an integer program is still required to compute the exact assignation of the transactions to the robots.

The lower bound N obtained from the network flow is used as a parameter to solve the RPA problem in a single phase as follows. Please refer to the Section Notation for more details on the formulation of the problem.

The objective is to minimize the total number of robots while penalizing the startups:

$$\min \sum_{k\in K} n_k + y_k \tag{1}$$

s.t.

$$\sum_{r\in R}\sum_{k\in K} x_{rkp}\, w_{kp} = v_p \quad , \quad \forall p \in P, \tag{2}$$

$$\sum_{r\in R}\sum_{p\in P} x_{rkp} t_p \leq l_k n_k \quad , \quad \forall k \in K, \tag{3}$$

$$n_k - n_{k-1} = y_k \quad , \quad \forall k \in K\backslash\{1\}, \tag{4}$$

$$n_1 = y_1 \quad , \tag{5}$$

$$n_k \leq N \quad , \quad k \in K, \tag{6}$$

$$\sum_{r\in R}\sum_{p\in P} x_{rkp} \geq n_k \quad , \quad k \in K, \tag{7}$$

$$n_k \in \mathbb{N},\ y_k \in \mathbb{N} \quad , \quad \forall k \in K, \tag{8}$$

$$x_{rkp} \in \mathbb{N} \quad , \quad \forall k \in K,\ \forall p \in P\ , \tag{9}$$

$$\quad , \quad \forall r \in R.$$

Constraints (2)-(3) ensure respectively that all the transactions are processed and that the period lengths are respected. Constraints (4)-(5) allow to flag the robot startups. Constraints (6) use the computed lower bound from the network flow and state that the number of robots at each period cannot exceed N. If a solution is found with this number of robots, then it is optimal, otherwise, N is increased by 1 at each step until a solution is found. The lower bound designed is often tight on the test cases of this paper. Since these test cases are rather generic, infeasability is expected to occur rarely and if so, to be resolved within a few steps. Constraints (7) ensure that if no transactions are processed during a period, then no robot is used and finally, constraints (8)-(9) describe variables' domains.

B. Data

As only one real test case of 12 transactions types is available, synthetic test cases were created. The following data is available for the real test case: number of transaction types, and for each type: the number of transactions to be processed, the time required to process a transaction, the market opening hours and the clearance dates.

Synthetic test cases are designed with respectively 20, 30, 40 and 50 different transactions types. In order to keep the generated instances realistic, the data distribution is based on the real test case. Therefore, the processing times of transactions are generated between the maximum and the minimum values of the real test case. The same protocol is followed for the volume of transactions. Market hours and clearances are randomly generated among those of the real test case as well. The data is generated following the distribution (proportions) of the real test case shown in Table I.

Table II shows the details of the tested instances.

C. Numerical results

To obtain solutions to the optimization models, the AMPL [18] modelling language was used with the Xpress [19] solver on an Intel Core^{TM} i5-8250U CPU running at 1.6 GHz with a RAM of 16 GB.

TABLE I: Distribution of the time vs. volume per day in the real test case.

Volume / Time	0-200	200-400	400-600	600-800	800-14000	Total
0-200	3	2	2	1	1	9
200-400	0	1	0	0	0	1
400-600	0	1	0	0	0	1
600-1400	1	0	0	0	0	1
Total	4	4	2	1	1	12

TABLE II: Details of the tested instances

$\|P\|$	$\|K\|$	t_p		v_p	
		Min	Max	Min	Max
12	36	15	1320	76	13750
20	72	22	1036	12	13430
30	72	6	805	22	7569
40	80	10	867	12	4952
50	80	15	1180	18	6158

A comparison between the solutions obtained with the two-phase method derived from a previous work (see Section II) and the solutions obtained with the new methodology is conducted.

In this paper, a two-phase method is also required to compute the lower bound on the total number of robots, but a single linear integer optimization model is required to obtain the schedule of the robots.

Table III presents the minimum number of required robots found by each method. *Phase I*, *Max Flow* and *New ILP* refers respectively to the previous methodology developed, the lower bound on the number of robots, and the optimal number of robots respectively. On all the tested instances, the computed lower bound is the optimal solution. So even if there is no proof of tightness for this lower bound, experiments suggest that it is of excellent quality. If the computed lower bound is not the optimal solution, then the value of N in Eq.(6) is increased by 1. The model is then solved another time and the process is repeated until a feasible solution is found.

TABLE III: Number of robots computed by the different methods

Nb. of transaction types	Phase I	Max Flow	New ILP
12	11	11	11
20	15	15	15
30	7	7	7
40	20	20	20
50	51	51	51

Table IV presents the running times in seconds for the different methods. First, the two-phase method derived from a previous work, then the maximum flow algorithm that provides a lower bound on the number of robots and the new integer linear problem that solves the problem in one optimization phase.

The running times are always under 1 second. One can observe that the total running times of the previous work are smaller, but, as mentioned earlier, computing time is required to format the data files for the Phase II input. Also, a comparison between the Phase I column and the Max Flow

TABLE IV: Running times in seconds for the different methods

Nb. of transactions	Previous work		Total	Max Flow	New ILP	Total
	Phase I	Phase II				
12	0.093	0.078	0.171	0.005	0.265	0.270
20	0.110	0.079	0.189	0.014	0.157	0.171
30	0.094	0.079	0.173	0.011	0.156	0.167
40	0.094	0.078	0.172	0.036	0.313	0.349
50	0.109	0.094	0.203	0.043	0.813	0.856

column of Table IV shows that the latter are at least 100 times smaller. Therefore, computing the number of robots is a lot faster with the new methodology, and there is no need to format data files as input to the optimization model. In addition, the network flow modelling of the problem is well suited for the dynamic environment of transactions, as they appear daily in an operational context.

The proposed methodology was tested on instances with up to 50 transaction types following the needs of the bank that provided the real test case. The increase in computational time is correlated with the increase in the number of transaction types but also with the maximum volumes and processing times of those transactions. In other words, one may expect an increase in the running time of the algorithms if they deal with more complex instances. However, as it is shown in Table IV, these times remain very reasonable.

As the financial systems are becoming bigger and ever more complex, the proposed solutions method, which consists in computing the lower bound on the total number of robots before conducting the optimization, may provide good quality results within a reasonable time.

V. Conclusion

RPA is used to mimic repetitive humans' tasks and achieve a gain in the time required to perform those tasks but also reduce the risks of errors. In the current highly competitive economical environment, it becomes ever more important to develop efficient solution methods in order to achieve a higher productivity and a better management of resources. The field of robotic process automation is no exception. In this paper, two solution methods for the RPA problem are compared in the context of financial institutions. The first one uses a two-phase integer program while the second one uses a maximum network flow algorithm to compute a lower bound on the number of robots before solving an integer program. In both cases, the optimal number of robots is computed and the schedule of each robot is determined. The two methods are tested on a real test-case and on synthetic ones. Results show that using the lower bound substantially reduces the running times and also induces a simpler optimization model. Further work based on this research will involve adding the reconfiguration costs of the robots to the formulation and exploring other graph theory algorithms to obtain a solution in a single phase.

VI. Acknowledgments

The authors would like to thank Guillaume Routhier for providing the data for this study.

References

[1] W. van der Aalst, M. Bichler, and A. Heinzl, "Robotic process automation," *Business Information Systems Engineering*, vol. 60(4), pp. 269–272, 2018.

[2] L. Willcocks, M. Lacity, and A. Craig, "The IT function and robotic process automation," London, US: University of Missouri-St-Louis - The London School of Economics and Political Science, Tech. Rep., 2015.

[3] A. Asatiani and E. Penttinen, "Turning robotic process automation into commercial success – case opuscapita," *Journal of Information Technology Teaching Cases*, vol. 6(2), pp. 67–74, 2016.

[4] S. Séguin and I. Benkalai, "Robotic process automation (RPA) using an integer linear programming formulation," *Cybernetics and systems (Accepted)*, 2020.

[5] J. Figueroa-García, E. López-Santana, J. Villa-Ramírez, and R. Ferro-Escobar, "Automation of a business process using robotic process automation (RPA): A case study," in *Applied Computer Sciences in Engineering*. Cham. Springer International Publishing., 2017.

[6] M. Lacity, L. Willcocks, and A. Craig, "Robotic process automation at telefonica o2," London, US: University of Missouri-St-Louis - The London School of Economics and Political Science, Tech. Rep., 2015.

[7] C. Lamberton, D. Brigo, and D. Hoy, "Impact of robotics, RPA and AI on the insurance industry: Challenges and opportunities," *Journal of Financial Perspectives*, vol. 4(1), pp. 8–20, 2017.

[8] D.-S. Chen, R. Batson, and Y. Dang, *Applied integer programming : Modeling and solution.* Hoboken, N.J.: John Wiley Sons, 2010.

[9] A. Schrijver, "On the history of the transportation and maximum flow problems," *Mathematical programming*, vol. 91(3), pp. 437–445, 2002.

[10] T. Harris and F. Ross, "Fundamentals of a method for evaluating rail net capacities," Research Memorandum, Tech. Rep., 1955.

[11] L. Ford and D. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[12] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 26(9), pp. 1124–1137, 2004.

[13] V. Kolmogorov, "Graph-based algorithms for multi-camera reconstruction problem," 2003.

[14] I. Benkalaï and S. Séguin, "Robotic process automation (RPA) using an integer linear programming formulation," *Cybernetics and Systems*, 2020, accepted.

[15] S. Séguin and G. Routhier, "Robotic process automation (RPA) using an integer linear programming formulation," in *Proceedings of the sixth International Conference on Control, Decision and Information Technologies*, 2019.

[16] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399–404, 1956.

[17] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *J. ACM*, vol. 35, no. 4, pp. 921–940, Oct. 1988. [Online]. Available: http://doi.acm.org/10.1145/48014.61051

[18] "AMPL: A modeling language for mathematical programming," *Albuquerque, NM: AMPL Optimization inc.*, 2014.

[19] F. I. Corporation., *FICO Xpress optimization suite.* San Jose, CA: FICO., 2014.