Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 2689–2698

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

# Minimizing the number of robots required for a Robotic Process Automation (RPA) problem

Sara Séguin[a,b,*], Hugo Tremblay[a], Imène Benkalaï[a,b], David-Emmanuel Perron-Chouinard[a], Xavier Lebeuf[a]

[a] *Université du Québec à Chicoutimi, 555 boul. de l'Université, Saguenay, G7H 2B1, Canada*
[b] *Group for Research in Decision Analysis (GERAD), 3000, ch. de la Côte-Sainte-Catherine, Montréal, H3T 2A7, Canada*

**Abstract**

Robotic Process Automation (RPA) is used in various fields of human activity in order to implement faster and more secure processes through a reduction in the risks or errors but also an increase in the productivity rates. The increase of its use and importance calls for evermore efficient solution methods for this problem. In this paper, the RPA is addressed in the context of a financial institution. The problem consists in assigning transactions to software robots, where each transaction type has a different clearance date and a different processing time. First, four heuristics are used to compute an upper bound on the number of required software robots. Then, this bound is given as a parameter to an integer linear program, which is used to assign the transactions to the different robots. The quality of the solutions is assessed by an extensive experimental study on a set of 39,000 instances. The results show that two heuristics outperform the others and allow for a faster resolution by the integer linear program which in turn finds the optimal solution for most of the instances within a timeout of 60 seconds.

*Keywords:*
Robotic process automation; linear integer programming; bin packing; heuristics; upper bound.

## 1. Introduction

Throughout history, new inventions have allowed mankind to achieve ever-better productivity. Nowadays, advanced software that completes complicated tasks is widely used and available. This allows to automate certain processes in several fields of application. This so-called RPA is often achieved by implementing codes that can directly interact with the data and process it in the background while other software is running. This represents an interesting alternative to traditional Graphical User Interfaces (GUI) software where users are required to directly interact with machines, often

[*] Corresponding author. Tel.: +1-418-545-5011 x 5604
*E-mail address:* sara.seguin@uqac.ca

resulting in human errors. Robotic Process Automation increases productivity in such situations and is considered a highly promising approach with an increasing number of real-life applications [18]. Indeed, more and more companies rely on this technology to complement their working teams and optimize their internal processes [13]. This form of automation relies on software that mimics human mouse movements and other inputs when interacting with a GUI. These software robots are programmed to perform specific tasks faster and with more accuracy than humans [19, 1]. Furthermore, the software runs continuously and does not feel boredom or exhaustion, which are human risk factors. Many companies now use RPA successfully [3, 11, 14, 7]. A specific field in which RPA is used to automate critical operations is banking. Financial institutions must handle a large volume of transactions daily, some of which are short and repetitive. When processing the latter type of transaction, humans would typically get bored and lose focus rather fast. Thus, using software robots instead seems like a more effective solution as their processing times and productivity are not affected by contingencies such as exhaustion and boredom. Each transaction is associated with a type, which has specific market hours, volumes, clearance dates and processing times. In the context of RPA implementation, these transactions are to be assigned to software robots that will process them. While RPA can lead to better efficiency and accuracy, it comes with a cost. Indeed, each software robot requires a paid license to operate. Therefore, there is a significant incentive to minimize the number of software robots necessary to process a certain number of transactions. As stated by the authors of [8], even though industry is interested in the RPA problem, academic literature is lacking on this specific problem, but recently gained attention [17, 10]. In this paper, the technical implementation of such a problem is proposed.

Several recent papers deal with applying RPA on real case studies [15, 9]; all of them highlight a given gain in productivity achieved by the company after implementing RPA. In [2], the authors compare assigning back-office tasks to robots and to humans. Back-office tasks being often tedious and repetitive, when they are assigned to humans, the latter tend to get exhausted or bored after a given time which increases the risks of error and extends the processing times while the robots do not have these drawbacks. Indeed, the research showed a gain of up to 20% in the number of cases treated by the company with the implementation of RPA in complement of their existing teams. In [11], Lacity *et al.* showed that using RPA at the company Telefònica O2 both enhanced consumer experience and lowered the costs incurred. Another study by Lamberton *et al.* [12] showed that the number of people required to process claims of an insurance company would be up to 300% larger without the implementation of RPA. In [16], an integer linear program is proposed to minimize the total costs. Periods are defined at the intersections of market operating hours for all of the transactions, leading to a reduced variable space. A first phase computes the total number of robots required, whereas a second phase assigns the transactions to the robots. Preliminary results on only two test cases led to feasible results, but further experimentation's showed that transactions could not be divided between robots, therefore leading to a higher number of actual robots required. Also, the actual use of a period is not considered, which could lead to having periods where robots are only active, for example, 5% of the available time. Another approach [4] solves a network flow problem in the first phase and an assignment problem in the second phase. The network flow allows to find a lower bound on the number of robots. The complexity of the problem, but also the computational time to obtain a solution is significantly lower than the first proposed approach [16]. Extensive experiments also show that this method suffers from the indivisibility of the transactions between the robots.

It is also worth mentioning that since RPA is a relatively new technology, many companies still tend to have some difficulty to grasp its effects, the means to implement it in practice and also its challenges. In a attempt to fill that gap, Wewerka and Reichert present in [18] a systematic literature review and a framework to analyze and compare publications on RPA. That paper highlights the differences between RPA and other similar technologies and details where RPA can be implemented, the tools that are used to implement it, its effects and the future improvements that should be considered in future studies.

The RPA problem presented in this paper consists in determining the minimum number of robots required in order to complete several concurrent financial transactions. Each transaction type is constrained by market operation hours and a clearance date. It also has a volume (i.e. number) of transactions to be completed, each of which require a given processing time. The solution to the problem provides a schedule for the robots that minimizes the maximal number of active robots at a given period for different transactions combinations. The main contribution of this paper is an improvement of the method described in [16] for solving the RPA assignment problem. More precisely, four different heuristics are introduced to compute an upper bound on the total number of required robots. In a second phase, an integer linear program uses the computed upper bound to find an optimal assignment of the transactions to the robots.

Numerical results show that the upper bounds found by the heuristics are of good quality and computed very fast. This new approach addresses some drawbacks that were discovered when attempting to solve a large number of instances with the original approach. For example, one transaction could be divided between different robots, and the proposed optimization model consisted of two phases, whereas a single phase is proposed in this paper, allowing a more efficient solution in practice.

The paper is organized as follows. Section 2 describes the RPA problem and the proposed linear integer formulation. Section 3 details the proposed heuristics to compute the upper bound on the number of robots. Numerical results are presented in Section 4 and final remarks are presented in Section 5.

**Nomenclature**

| | |
|---|---|
| P | set of transaction types |
| K | set of periods |
| R | set of robots |
| $v_p$ | volume of transactions of type $p \in P$ |
| $t_p$ | process time of transaction type $p \in P$ |
| $l_k$ | length of period $k \in K$ |
| $w_{kp}$ | 1 if transaction type $p \in P$ can be processed in period $k \in K$, 0 otherwise. |
| $N^{ub}$ | upper bound on the number of robots required (calculated with heuristic) |
| $x_{rkp}$ | number of transactions of type $p \in P$ processed by robot $r \in R$ in period $k \in K$ |
| $N$ | number of robots |
| $y_{rk}$ | 1 if robot $r \in R$ is active in period $k \in K$, 0 otherwise. |

## 2. Problem Description

The RPA assignment problem is stated as follows. Several transactions are to be completed using the lowest possible number of robots for each period. A period is defined by the different types of operations that can be processed between the beginning and the end of that period. In other words, periods are defined by the intersections of market operating hours for all of the transactions; if the set of transaction types that can be processed changes at a certain time, a new period is defined.

Each transaction type is characterized by its market hours, clearance date, processing time and volume. The available market hours of a type is what determines the set of periods during which it can be processed. Transactions that share the same characteristics are grouped under a type, and the number of such operations of the same type is referred to as the volume.

The aim is to assign each transaction to a given period. The number of transactions completed in a period is constrained by the length of the period, the number of robots assigned to the period, as well as the processing time of each operation. Every robot has the time limit of the period it is assigned to at any moment. The objective of the RPA assignment problem is to find an assignment that minimizes the largest number of robots required among all periods, for any given set of operations to complete under their respective clearance dates.

In this paper, the operations are financial transactions and each robot is a software license that incurs a fixed cost. Therefore, minimizing the largest number of robots among all periods reduces license costs. In what follows, a linear integer program is presented.

### 2.1. Mathematical model

This integer linear program assigns the transactions to the robots and requires an upper bound on the total number of robots. The upper bound $N^{ub}$ is computed using heuristics and is discussed in Section 3.

The model minimizes the maximal number of robots required for all periods:

$$\min \quad N \tag{1}$$

$$s.t. \quad N \geq \sum_{r \in R} y_{rk}, \qquad \forall k \in K, \tag{2}$$

$$\sum_{k \in K} \sum_{r \in R} x_{rkp} = v_p, \qquad \forall p \in P, \tag{3}$$

$$\sum_{r \in R} x_{rkp} \leq w_{pk} v_p, \qquad \forall p \in P, \forall k \in K, \tag{4}$$

$$\sum_{p \in P} x_{rkp} t_p \leq l_k y_{rk}, \qquad \forall k \in K, \forall r \in R, \tag{5}$$

$$\sum_{p \in P} x_{rkp} \geq y_{rk}, \qquad \forall k \in K, \forall r \in R, \tag{6}$$

$$N \leq N^{ub}, \tag{7}$$

$$y_{rk} \in \mathbb{B}, \qquad \forall k \in K, \forall r \in R, \tag{8}$$

$$x_{rkp} \in \mathbb{Z}^+, \qquad \forall k \in K, \forall r \in R, \forall p \in P. \tag{9}$$

In this model, the sets $K$ and $P$ are given by the set of transactions to be completed (the input data). On the other hand, the set of robots $R$ contains $N^{ub}$ robots. Without a heuristic algorithm computing this upper bound, the length of $R$ would have to be either infinite or arbitrary. Eq. (2) are used to minimize the maximal number of robots $N$. If a period requires $N$ robots and it is the highest number of robots among all periods, then $N$ licenses will need to be paid, regardless of the other periods. Minimizing the total number of robots is thus equivalent to minimizing the number of active of robots in the most demanding period. Eq. (3) are used to assign every transaction of each type. Eq. (4) set the value of $x_{rkp}$ to 0 when $w_{pk}$ is 0, so that no transaction is assigned outside of its market hours. Eq. (5) are used to enforce the respect of the period lengths. They also ensure that $x_{rkp}$ is 0 if $y_{kr}$ is 0, which means that transactions can only be assigned to active robots. Eq. (6) state that $y_{kr}$ must equal 0 if $x_{rkp}$ equals 0, in other words, a robot can only be active if transactions are assigned to that robot. Without these constraints, the periods that do not require the highest number of robots could uselessly contain active robots without increasing $N$. Finally, the bound on the number of robots is given by Eq. (7). Variables' domains are given by Eq. (8) -(9). The result is an assignment of every transaction of type $p \in P$ to robot $r \in R$ in period $k \in K$ given by $x_{rkp}$.

## 3. Heuristics to compute the upper bound $N^{ub}$

This section first demonstrates that the RPA assignment problem is NP-complete, meaning any exact method for its resolution would have exponential complexity. Removing the upper bound $N^{ub}$ in the mathematical model of Section 2.1 would indeed require an enormous amount of time given the complexity of the problem.

In order to calculate the value of this upper bound $N^{ub}$, four heuristics are developed which provides close to optimal solutions rather fast. Then, using the obtained upper bound, the mathematical model is solved to find an optimal assignment of the transactions to the robots.

### *3.1. Proof of NP-Hardness*

In this section, it is shown that the RPA assignment problem is NP-hard. In order to do so, a reduction from the well-known bin packing problem to the RPA assignment problem is proposed.

The bin packing is an optimization problem that is known to be NP-hard. In this problem, $p$ indivisible packages of sizes $s_1, s_2, ..., s_P$ must each be assigned to bins. The objective is to assign all packages using the least possible number of bins of equal capacity $c$. Fig. 1 presents an example of such a problem where five packages must be placed in three bins.

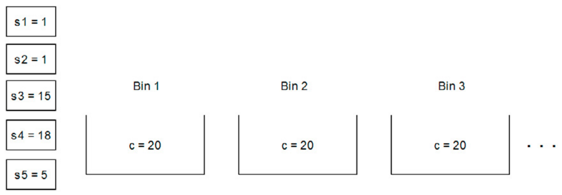


Fig. 1. Example of a bin packing problem

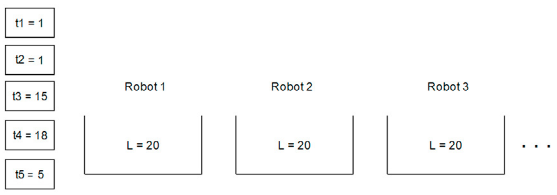


Fig. 2. RPA problem equivalent to the bin packing example

It is clear the RPA assignment problem is in *NP*. Indeed, any solution can be verified in polynomial time. Now, consider a restriction of the RPA assignment problem where every transaction type has a volume of one and can be processed in a single period of length $L$. Then, the objective is to find the minimum number of robots in the period that can complete all transactions. Each robot can work for at most $L$ units of time, and each transaction has an associated processing time requirement $t_1, t_2, ..., t_P$. This restriction of the RPA assignment problem is equivalent to a bin-packing problem for which each package of size $s_p : p \in P$ is a transaction with a processing time $t_p : p \in P$, and where a new bin with capacity $c$ is equivalent to a robot active in the period if $L = c$.

A visual representation of this reduction is presented in Fig. 2. By comparison with Fig. 1, it is obvious that the two problems are equivalent. Bins correspond to robots and packages to different transactions. Therefore, the RPA assignment problem is NP-hard.

## 3.2. *Heuristic algorithms*

Four heuristic methods are tested. Each are inspired by the *First fit* [6] and *First Fit Decreasing* [5] algorithms used to solve the bin packing problem. To the authors knowledge, these heuristics have not been used in the context of a RPA problem. The first heuristic is explained in greater details since the other three are very similar.

1. **First Fit (FF) Algorithm.** (**Algorithm 1**)
   This algorithm is very similar to the FF used in the offline bin packing problem found in the literature [6]. Transaction types are considered in the order they are given, and periods are considered in chronological order. The entire volume of a certain transaction type is assigned before moving on to another type. Each transaction is assigned to the first possible period that is in the set of valid periods and has enough time to complete it. If there is no period in which a particular transaction can be executed, a new robot is added to all periods. However, a new robot for any period is only used if all previously added robots in all other possible periods are full. A schematic representation of these explanations is visible in Fig. 3.

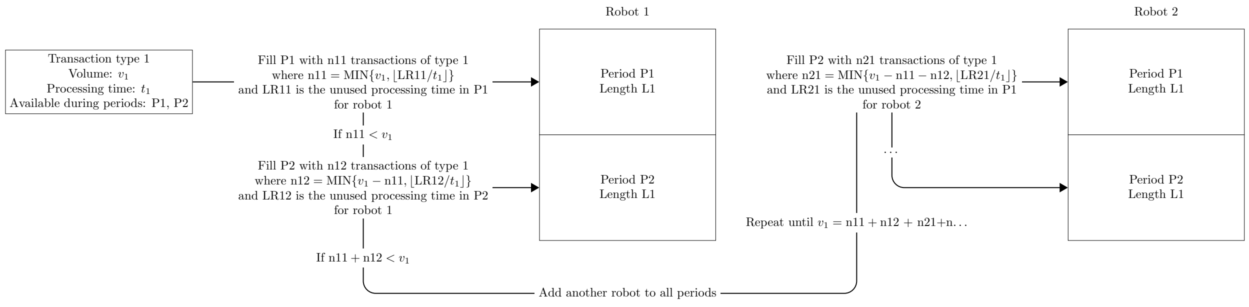

Robot 1
Robot 2
Transaction type 1
Volume: $v_1$
Processing time: $t_1$
Available during periods: P1, P2
Fill P1 with n11 transactions of type 1
where n11 = MIN{$v_1$, ⌊LR11/$t_1$⌋}
and LR11 is the unused processing time in P1
for robot 1
If n11 < $v_1$
Fill P2 with n12 transactions of type 1
where n12 = MIN{$v_1$ − n11, ⌊LR12/$t_1$⌋}
and LR12 is the unused processing time in P2
for robot 1
If n11 + n12 < $v_1$
Period P1
Length L1
Period P2
Length L1
Fill P2 with n21 transactions of type 1
where n21 = MIN{$v_1$ − n11 − n12, ⌊LR21/$t_1$⌋}
and LR21 is the unused processing time in P1
for robot 2
...
Repeat until $v_1$ = n11 + n12 + n21+n...
Period P1
Length L1
Period P2
Length L1
Add another robot to all periods

Fig. 3. Algorithm 1 explained

2. **FF with Decreasing Transaction Processing Time Algorithm.** (**Algorithm 2**)
   The only difference with this algorithm and Algorithm 1 is that the different transaction types are ordered in decreasing order of processing time before being assigned to a period.
3. **FF with Decreasing Period Length Algorithm.** (**Algorithm 3**)
   This algorithm is also similar to Algorithm 1. However, the periods are ordered in decreasing order of length. This means that, for any given transaction, this algorithm tries to assign it to the longest period in which it can be executed, before checking the other ones.
4. **FF with Decreasing Transaction Processing Time and Decreasing Period Length Algorithm.** (**Algorithm 4**)
   This algorithm is essentially a mix of both previously described algorithms. It is similar to the first one described, except both the transactions and the periods are sorted in decreasing order.

## 4. Experimental results

This section details how the synthetic test cases are generated and describes the testing environment for the different algorithms. Numerical results are then presented, followed by a discussion.

### *4.1. Synthetic test cases generation*

The RPA assignment problem in this study is applied in the context of a financial institution. A bank in North America provided a real test case with 12 transaction types. In order to further assess the quality of the proposed method, synthetic test cases are generated based on the distribution of data of the real test case. For each synthetic transaction type generated, the market operating hours and the clearance date are selected randomly from those of the set of transaction types of the real case. As for the processing time and volume, bounds are randomly selected from the real case, then random values are generated for both attributes, while being limited by the upper and lower bounds previously selected.

Synthetic cases with between 12 and 50 transaction types are generated in order to observe the impact of the instance sizes on both the solution quality and the running time of methods. In order to obtain meaningful results, 1,000 instances are generated for each number of different transaction types ranging from 12 to 50. A total of 39,000 instances are used to test the proposed methodology.

All tests are conducted on an "E2-Standard" Google Cloud Platform virtual machine. It consists of 4 virtual CPUs, 16 GB of main memory and an Intel Haswell platform. The operating system used is Debian 10 (Buster). The C programming language is used to implement the heuristics, without parallelism or multithreading. It is worth mentioning that computing time would be reduced if parallelism, multithreading or another sorting algorithm was used. For this paper, the qsort function is used. For the integer linear program, Python programming language and Xpress solver are used.

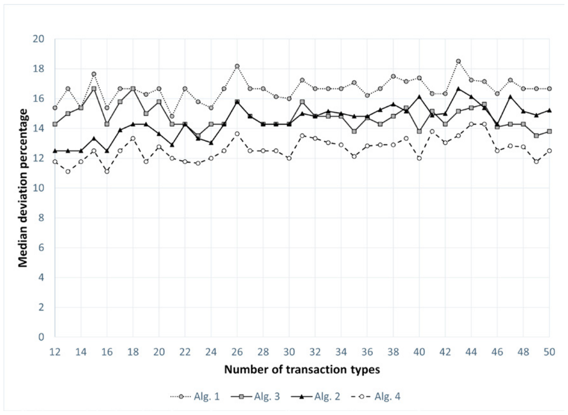
### *4.2. Numerical results and analysis*

The numerical results aim at determining if the bound calculated with the heuristics is close to the number of robots optimized with the integer linear program. Therefore, the number of robots and calculation times are of utmost importance and the numerical results focus on these elements.

First, the heuristics are used to determine an upper bound on the number of robots that is then used as a parameter in the optimization model. Each heuristic is run until it finds a solution and the linear integer program is first limited to a 60 seconds timeout in order to find a solution quickly. For the instances that did not reach optimality after 60 seconds, the timeout is set to 15 minutes before trying to solve them again.

The different heuristic methods provided good quality results in general. Fig. 4 shows the median deviation percentages of the heuristics compared to the Integer Linear Program (ILP), more precisely the difference, in percentage, between the number of robots found by the heuristics and the optimal number of robots in the ILP. For the sake of fairness, only the instances for which optimal solutions were obtained by the ILP were considered. It can be seen that the median deviation percentages does not exceed 20% which provides a rather good upper bound that allows to speed up the resolution of the ILP. The best heuristic is Algorithm 4, which is FF with decreasing transaction processing time and period length. The worst heuristic is Algorithm 1 and results show that Algorithm 4 greatly improves the bound.

Furthermore, the heuristics are, in some cases, able to find the optimal solution to several instances. Fig. 5 presents the success rates of the different methods, *i.e.* the percentage of instances for which optimal solutions are found by each method. Even if this rate decreases with the increase in the number of transaction types, heuristics are still able to provide good quality upper bounds that become good starting points for the resolution of the ILP. On all instances, the best algorithm is Algorithm 4 as it is the only one that is non-dominated.



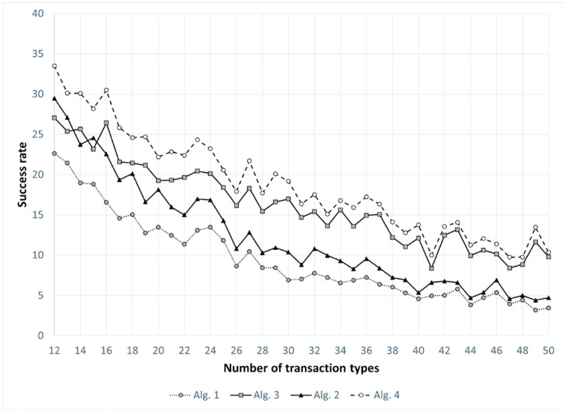


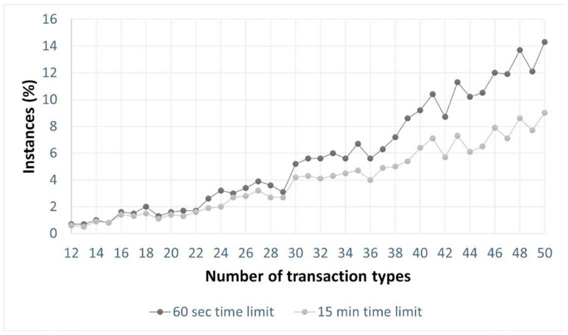
Fig. 4. Median deviation percentages for all solution methods

Fig. 5. Success rate of the different solution methods

It is worth mentioning that for a small percentage of instances, the ILP was not able to find optimal solutions and in rarer cases, was not able to find any solution, even after 15 minutes. The percentages are presented in Fig. 6 and 7. In the case of large instances, allowing a 15 minutes timeout reduces the percentage of instances with no optimal solution from 14 % to about 9 %. It is also worth mentioning that the ILP almost always finds a solution after 15 minutes, and that less than 0.4 % of instances are left without a solution.

In the cases where non-optimal solutions are found, the gap does not exceed 7% as shown in Fig. 8. For the cases where no solutions are found by the ILP, the heuristics are a good alternative since they provide good quality solutions in a very short time.

For the ILP, the running time is almost instant for 12 transaction types and 2.5 seconds for 50 transaction types. Of course, the time grows exponentially with the number of transactions. Concerning the running times for the heuristics, they can be viewed on Fig. 9. Algorithm 1 and Algorithm 2 are at least two times faster than Algorithm 3 and Algorithm 4. The heuristics are three to six orders of magnitude faster than the ILP.



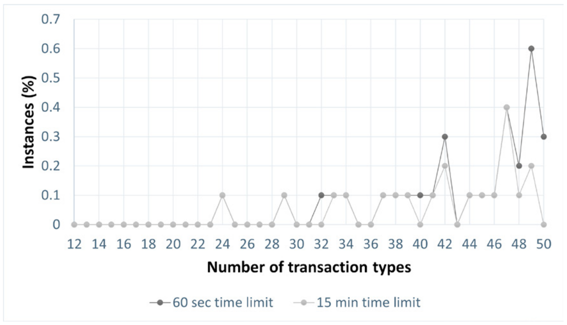


Fig. 6. Percentage of instances with non-optimal solutions found by the ILP solver. The dark line is after 60 seconds and the pale line after 15 minutes.

Fig. 7. Percentage of instances with no solutions found by the ILP solver after 60 seconds (dark line) and 15 minutes (pale line)

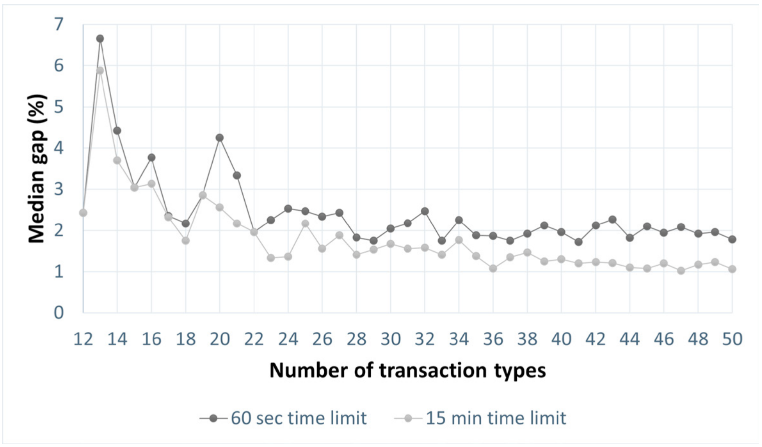


Fig. 8. Median optimality gaps for the ILP after 60 seconds (dark line) and 15 minutes (pale line)

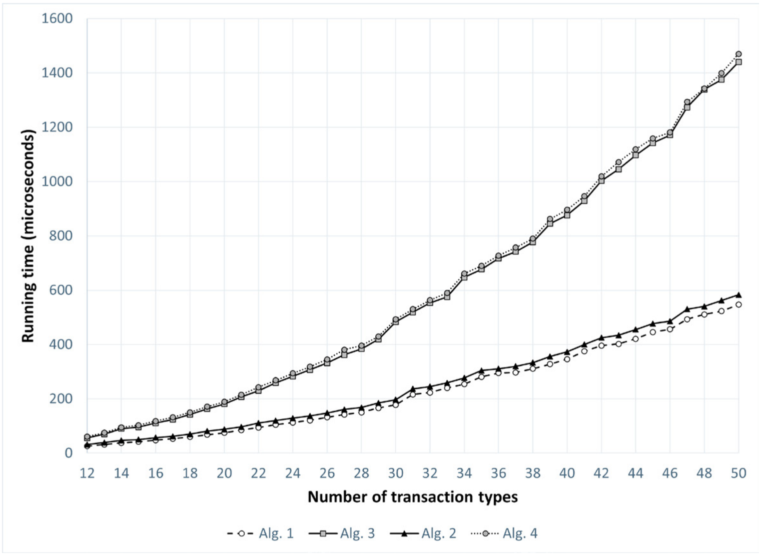


Fig. 9. The median running times of heuristics in microseconds.

Fig. 10 presents the median differences in the number of robots between the heuristics and the ILP for the instances where an optimal solution is found. As expected, this difference increases with the increase in the number of transaction types. Algorithm 4 is the best algorithm on large instances.

As previously mentioned, the solver for the ILP was first run with a 60 seconds timeout, if no optimal solution is found, the timeout is set to 15 minutes. In most cases, the number of robots is improved by only one unit which represents less than 0.5% improvement in average. This leads to the conclusion that 60 seconds of timeout is enough since the heuristics provide a good quality upper bound which, combined with the ILP, allows to obtain quite good solutions in a very short time.

Fig. 11 shows the median optimality gap differences between solutions found after 60 seconds and after 15 minutes. The majority of instances see a reduction of 0.5% in the optimality gap. This shows that running the ILP solver with a 60 seconds timeout is sufficient. Also note that the heuristic used to compute the bound is the Algorithm 4 for this comparison.

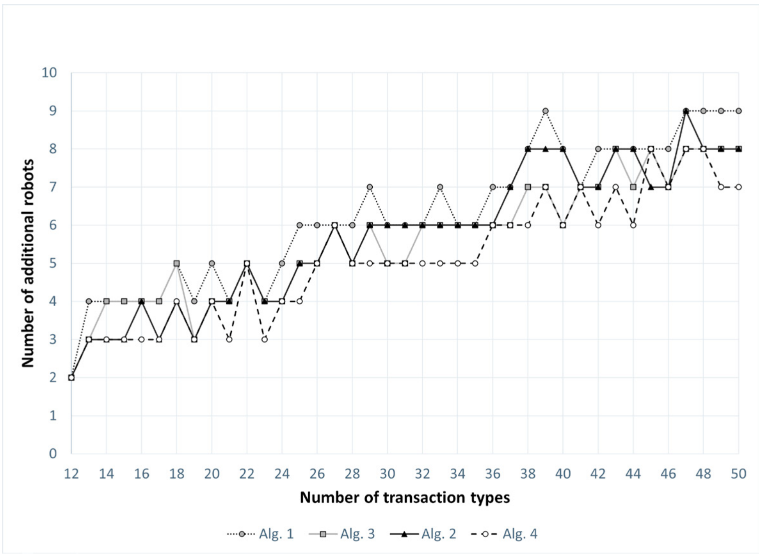


Fig. 10. The median difference of the number of robots with respect to the optimal solution.

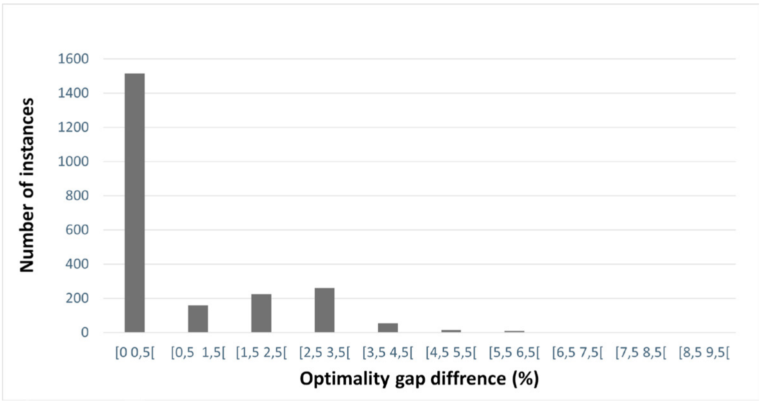


Fig. 11. The improvement in the optimality gap between the 60 seconds and the 15 minutes runs of the ILP solver.

### *4.2.1. Analysis*

Results show that the heuristics allow to obtain a good upper bound on the number of robots for the linear integer program. Since the heuristics run in microseconds, it is worthwhile to compute a bound before conducting the optimization. As for the ILP, setting a 60 seconds timeout is sufficient, given that only 5.75% of the instances are not solved to optimality in the given timeout. Of the total 39,000 instances, this represents about 2243 instances. For most of the instances without an optimal solution in 60 seconds, the optimality gap is around 2% for the ILP. The instances with no feasible solutions after both timeouts are rather rare, representing only 0.6 % of the total number of instances. Also, it is worth mentioning that using a 15 minute timeout leads to a reduction in the number of instances for which no feasible solution was found as shown in Fig. 7.

Algorithm 3 and Algorithm 4 are the preferred heuristics since they offer the best bounds when compared to the optimal solution. In an operational perspective, these heuristics should be used before solving the optimization model. Indeed, even if they are slower than Algorithms 1 and 2, they are still very fast as they always run well under one second, even for the largest instances.

## 5. Conclusion

This paper presents a methodology to solve the RPA problem. First, four heuristics are compared to determine the best algorithm in terms of quality of solution and second, this bound is used as a parameter in a linear integer optimization problem. A solution to the RPA problem is proposed to assign financial transactions to software robots and 39,000 instances are generated to test the methodology. The results show that the heuristics provide a good bound, allowing to solve the ILP problem with a timeout of 60 seconds. A single phase optimization model is proposed and allows to use efficiently all of the available time during the periods by dividing transactions between periods and robots. The results are conducted on static test cases, but in a dynamic mindset, where transactions arrive in real-time, this methodology could easily be adapted to find a good solution quickly.

## Acknowledgement

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for providing funds to conduct this research.

## References

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

[2] Aguirre, S., Rodriguez, A., 2017. Automation of a business process using robotic process automation (RPA), in: Applied Computer Sciences in Engineering. Springer International Publishing, pp. 65–71.

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

[4] Benkalaï, I., Séguin, S., Tremblay, H., Glangine, G., 2020. Computing a lower bound for the solution of a robotic process automation (RPA) problem using network flows, in: 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE. pp. 118–123.

[5] Dósa, G., 2007. The tight bound of first fit decreasing bin-packing algorithm is FFD (i) ≤ 11/9 OPT (i)+ 6/9, in: International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, Springer. pp. 1–11.

[6] Dósa, G., Sgall, J., 2013. First fit bin packing: A tight analysis, in: 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

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

[8] Hofmann, P., Samp, C., Urbach, N., 2020. Robotic process automation. Electronic Markets 30, 99–106.

[9] Huang, F., Vasarhelyi, M.A., 2019. Applying robotic process automation (rpa) in auditing: A framework. International Journal of Accounting Information Systems 35, 100433.

[10] Ivančić, L., Vugec, D.S., Vukšić, V.B., 2019. Robotic process automation: Systematic literature review, in: International Conference on Business Process Management, Springer. pp. 280–295.

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

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

[13] Madakam, S., Holmukhe, R.M., Jaiswal, D.K., 2019. The future digital work force: robotic process automation (rpa). JISTEM-Journal of Information Systems and Technology Management 16.

[14] Parcells, S., 2016. The power of finance automation. Strategic Finance 98(6), 40–45.

[15] Romao, M., Costa, J., Costa, C.J., 2019. Robotic process automation: A case study in the banking industry, in: 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6.

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

[17] Syed, R., Suriadi, S., Adams, M., W.Bandara, Leemans, S.J., Ouyang, C., ter Hofstede, A.H., van de Weerd, I., Wynn, M.T., Reijers, H.A., 2020. Robotic process automation: Contemporary themes and challenges. Computers in Industry 115, 103162.

[18] Wewerka, J., Reichert, M., 2020. Robotic Process Automation – A Systematic Literature Review and Assessment Framework. Technical Report. Cornell University.

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