Designing Optimal Robotic Process Automation Architectures

Geeta Mahala[1(✉)], Renuka Sindhgatta[2], Hoa Khanh Dam[1], and Aditya Ghose[1]

[1] University of Wollongong, Wollongong, Australia
gm168@uowmail.edu.au, hoa@uow.edu.au, aditya.ghose@gmail.com
[2] Queensland University of Technology, Brisbane, Australia
renuka.sr@qut.edu.au

Abstract. The design and implementation of Robotic process automation (RPA) requires an architecture where there is seamless coordination between humans, robotic agents, and intelligent agents automating information acquisition tasks and decision-making tasks. Effective coordination of agents would need to consider the efficiency of different types of resources in completing tasks, the quality when handling complex tasks, and the cost of resources executing the task. In this work, a novel approach for generating an optimal architecture considering distinct types of resources that include human, intelligent and robotic agents is proposed. An optimal architecture is the optimal enactment of process instances executed by a combination of human and automation agents based on their characteristics. The architecture considers resources, resource types, and their characteristics that meet multiple objectives of process execution.

Keywords: Robotic process automation · Multi-objective optimization · Genetic algorithm · Optimal resource architecture

1 Introduction

The idea of Robotic Process Automation, where some (or all) tasks in a business process are automated by deploying software agents, intelligent agents or conversational agents (chatbots) to execute tasks which would have traditionally been executed by human operators, has steadily grown in popularity over the recent past. By various accounts, the size of the global market for RPA products runs into the billions of dollars.

One of the problems that has been highlighted by researchers in this space is that of identifying which tasks to automate [15]. This assumes a simple dichotomy in the available agent types: *human agents* and *robotic agents*. The use of only the former type of agents would represent the old approach to executing processes while the use of the latter would represent an emphasis on automation. More recently, the repertoire of available agent types has expanded to also include *intelligent agents* which are endowed with the ability to learn from past experience and make more complex decisions.

E. Kafeza et al. (Eds.): ICSOC 2020, LNCS 12571, pp. 448–456, 2020.
https://doi.org/10.1007/978-3-030-65310-1_32

This greater variety of resource/agent types also leads to greater diversity in performance characteristics. Human agents tend to be expensive and are typically slower than robotic agents or intelligent agents. Intelligent agents sometimes turn out to be not adept at taking certain decisions (for instance due to limitations in their machine learning routines), necessitating the deployment of human agents to cross-check their decisions. Robotic and intelligent agents take very little time to complete their tasks, relative to human agents. We can also sometimes assume that there is an unlimited supply of robotic and intelligent agents, while we have to contend with a fixed number of expensive-to-deploy human agents. There are clearly, at the very least, three non-functional factors at play: time, cost, and performance (i.e., the extent to which an agent is able to deliver correct outcomes, measured on a numeric scale). We aim to minimize time and cost and maximize performance. It is important to note that all of these factors are measurable and easily monitored. We rate agent *types* on these factors for different *task types* for simplicity in this paper, although our approach could easily be extended to agent *instances*. Rating would involve assessing agents' time, cost, and performance on a comprehensive set of benchmark problems.

This leads us to the problem of identifying an *optimal RPA architecture*. Recall that the traditional notion of a software or system architecture involves the specification of how system components connect and interact to realize overall system functionality. In our setting, an *RPA architecture* specifies how the various agents involved in the execution of a (partially or fully) automated process connect and interact to realize overall process functionality. An RPA architecture specifies, for each process task in each process instance, the agent (type) allocated for executing that task. There is a vast space of possible allocations, each of which leads to different outcomes in terms of overall time taken, overall cost incurred, and performance (or quality of work) achieved in each of the tasks. This suggests a multi-criteria optimization problem where there are at least three distinct (and *incommensurable*) objective functions at play: cost minimization, time minimization, and performance maximization.

In the next section, we provide a background of the over-arching problem. Section 3 provides details of the multi-criteria optimization problem and motives the problem with an example. Section 4 presents related work. We present our conclusions and discuss future work in Sect. 5. It is instructive to take a step back and observe that the solution we have presented also solves the more general problem of devising *process architectures* (i.e., deciding which resource will execute each task in all process instances).

2 Background

An RPA implementation would require human agents and robotic agents working together with suitable coordination. The coordination between agents has been suggested in prior human-automation [12,21] and RPA [19] studies. Such a coordination between the agents can generally be broken down into three broad categories: (i) levels where the task is primarily performed by a human, (ii) levels

where the human-agent interaction is high during task execution, and (iii) levels with low human involvement.

Agents executing tasks can be broadly categorised as: (i) *Human Agent (HA)* capable of executing all types of (manual) tasks of the process, (ii) *Robotic Agent (RA)* or specialised software program, that automates *information acquisition* tasks or information gathering tasks, and (iii) *Intelligent Agent (IA)* that automates *information analysis or decision-making* tasks and improves its performance through learning [17].

Agents can have distinct resource characteristics such as performance, experience or suitability when executing different tasks [19]. For example, an IA may have good performance when performing a task to verify the details of the loan application document, but may have low performance in computing the credit risk of the applicant having large number of financial transactions in a different country. In such scenarios, an HA would be required to validate the task done by an IA. Thus, in certain scenarios referred to as a lower level of automation, an HA would often be required to execute the task again after its completion by an IA or RA. At higher levels of automation, HAs exercise a supervisory role intervening only if necessary (failures, errors, or poor execution quality), or further have tasks fully executed by an IA or RA. Human verification tasks may be added dynamically during process execution based on resource characteristics of the IAs or RAs.

The ability of distinct types of resources to automate various tasks of a process leads to increased execution alternatives. The choice of alternatives available needs to account for multiple and often conflicting objectives such as: i) minimizing the *cost* of execution as each of these types of resources have a cost associated with each type of agent executing a task, ii) maximizing the *performance* or quality of the work done by the resource, and iii) minimizing the *makespan* or the shortest possible time for all the process instances to complete execution. For example, an HA can be expensive and may take longer to complete the tasks but will be able to handle any task with minimal errors (or high quality). An IA or RA can take lower time to complete tasks but may execute certain tasks with lower quality. Choosing the right types of resources and the tasks executed by the resources results in a trade-off between different objectives. Hence, selecting an RPA architecture that considers resources, resource types, resource characteristics, and the necessary objectives is complex, time consuming, and crucial to avoid sub-optimal and error prone process executions.

3 Problem Formulation of an Optimal RPA Architecture

In this section we describe the optimization model to enable decisions on the suitability of an RPA solution that meets the required objectives. In this paper, we consider three important objectives. However, our approach is generic and can support multiple and additional objectives. The objectives are to minimize the cost and the overall time taken to execute the process instances (or makespan) while maximizing the performance. The optimization problem is formalized.

The inputs are as follows:

- W, be the maximum number of tasks, $i \in \{1, \ldots, W\}$
- M, be the maximum number of resources $j \in \{1, \ldots, M\}$
- C, be the maximum number of process instances, $k \in \{1, \ldots, C\}$
- ϕ_j, the type of resource j where $\phi_j \in \{HA, IA, RA\}$
- π_j, the cost of a resource of type ϕ_j
- σ_{ij}, is 1 if the the resource of type ϕ_j is *suitable* to perform task i match and zero otherwise
- α_{ij}, the effort on task i by a resource of type ϕ_j
- β_{ij}, the performance of resource of type ϕ_j on task i
- τ_{perf_i}, an acceptable performance threshold for task i
- $d_{ii'}$, is one if there is sequential dependency and tasks i precedes i'

The key inputs to the model are the resources and their characteristics such as suitability, performance, and cost.

Suitability is the inherent quality of a resource j to perform a task i. The suitability of a resource is considered to be a binary value. Suitability can be determined based on agent specification and implementation, or can be determined based on the organization model attributes such as role or department of the resource.

Performance: Automation agents are more susceptible to resource specific errors i.e. errors made by resources when performing a task [16]. Performance measure of an IA can be computed based on the algorithms implemented such as F1-score, root mean square error, precision, or precision@k [20]. These measures can be computed during the training and testing of the algorithms. The performance resource types ϕ_j, on task i can be computed using the measure specific to the implemented algorithms. A threshold τ_{perf_i} indicates an acceptable performance necessary for any agent. If the performance of an agent is below this threshold, the task is either re-executed by an HA or verified by an HA.

Effort: The time taken to complete a task i varies for different resource types. An IA or RA has higher processing power and hence takes significantly lower effort as compared to an HA.

The *process model* provides inputs on the dependencies between the tasks represented by $d_{ii'}$ specifying that to execute task i', the task i needs to be executed before.

The decision variables are as follows:

- x_{ij}, which is true if task i is assigned to resource j, and false otherwise
- e_i, the end date of task i
- s_i, the start date of task i

This leads to the following optimization objectives.

$$\min(\max(e_i) - \min(s_{i'}))\ \forall i, i' \quad \text{(Minimize makespan)}$$

$$\min \sum_{i,j} \pi_j x_{ij} \quad \text{(Minimize cost)}$$

$$\max \sum_{i,j} \beta_{ij} x_{ij} \quad \text{(Maximize performance)}$$

The following constraints are also imposed.

s.t.

$$\sum_j x_{ij} = 1 \ \forall i \text{ where } \beta_{ij} \geq \tau_{perf_i} \quad \text{(ONE TASK TO ONE RESOURCE IF PERFORMANCE IS HIGH)}$$

$$\sum_j x_{ij} x_{ij'} = 1 \ \forall i \text{ where } \beta_{ij} < \tau_{perf_i}, \phi_{j'} = HA$$

(ONE TASK IS SUPERVISED BY HA IF PERFORMANCE IS LOW)

$$\sum_j x_{ij}(e_i - s_i) \geq \alpha_{ij} \text{ for all } i \quad \text{(PLANNED END TIME OF TASK CONSIDERS EFFORT)}$$

$$\sum_j (x_{ij} x_{i'j} = 1) \Rightarrow (e_i < s_{i'} \vee s_i > e_{i'}) \text{ for all } i \neq i' \quad \text{(ONE TASK AT A TIME)}$$

$$\sum_{i,j} x_{ij} \sigma_{ij} > 0 \quad \text{(TASK SUITABLE BY RESOURCE)}$$

$$\sum_{ii'jj'} d_{ii'}(x_{ij} x_{i'j'} = 1) \Rightarrow e_i < s_{i'} \quad \text{(SEQUENTIAL DEPENDENCY)}$$

$$x_{ij} \in \{0, 1\}$$

Running Example. We illustrate the need for such an optimization architecture with the help of a simple example. Figure 1 (a) presents a business process with 3 tasks to process a claim application. The three types of resources HA, IA, and RA have different resource characteristics on each of these three tasks. An HA has highest performance but requires higher effort and comes with a higher cost. An IA can support the decision making task of evaluating a claim having lower performance than an HA but better cost and effort parameters. An RA can perform the task of notifying the status of the application with high performance but is incapable of performing the first two tasks of the process (namely checking the claim application and evaluating the claim). In this example, we consider all resources of a given resource type with the same resource characteristics. However, our problem formulation does not make any such assumption. The values for effort (α_{ij}), performance(β_{ij}), and cost (π_j) considered for the three tasks and three resource types is presented in Fig. 1 (a).

The allocation of task to resources that meets multiple objectives while satisfying the constraints can result in multiple solutions. One such solution of task allocation with 3 process instances and 6 resources (2 HA, 2 IA, and 2 RA) is shown in Fig. 1 (b). Solutions that meet multiple objectives form a Pareto front of minimum cost, minimum makespan, and maximum performance. A solution on a Pareto front does not dominate another solution on the same front, i.e. by moving along the curve, you could minimize makespan at the expense of reducing performance or maximize performance at the expense of increasing makespan. For example, in Fig. 1 (c) the three solutions on the blue line are non-dominating solutions. The solutions vary with the performance threshold set for the supervisory control of an HA ($\tau_{perf_i} = 5$ for the blue line), i.e. if the performance of an agent is lower than the threshold, it needs to be followed by another task that is performed by an HA to validate or supervise the task executed by an IA or an RA. If the performance threshold is lower (e.g. $\tau_{perf_i} = 3$, orange line), then an IA or RA can do the task without HA supervision. Thus, the makespan reduces, and so does the performance of the entire allocation. Similarly, if the

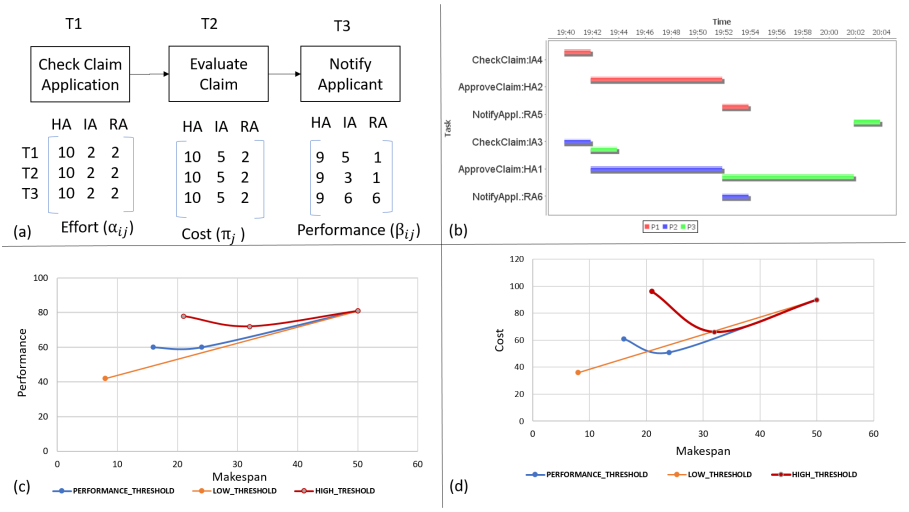

Fig. 1. (a) Example process with resource characteristics, (b) Optimal Solution to execute 3 process instances, (c) Performance vs. Makespan for different performance thresholds (τ_{perf_i}), and (d) Cost Vs. Makespan for different performance thresholds. (Color figure online)

performance threshold is set to a higher value (e.g. $\tau_{perf_i} = 6$), then most tasks performed by IA or RA will be supervised by an HA, resulting in increased performance and increased makespan. Figure 1 (d) presents the makespan and cost for the non-dominating solutions when different performance thresholds are set. The example motivates the need for considering distinct resources, their characteristics, multiple objectives, and constraints for selecting a suitable RPA architecture.

4 Related Work

Prior studies on RPA have focused on the design phase presenting techniques to identify candidate tasks for automation [15]. Studies have explored an increase in the scope of automation by the agents supported by Artificial Intelligence (AI) and Machine learning to do complex tasks [1,18]. Recent work has further distinguished types of resources and their characteristics in terms of suitability for execution of different types of tasks and presented a declarative specification to enable different levels of automation [19]. This work considers such a specifications as input where tasks can be executed by distinct types of resources permitting the generation and selection of an RPA architecture optimizing multiple objectives that are necessary for an effective implementation.

There have been extensive studies focusing on resources and their characteristics [2,7,9,13]. Resource characteristics are used for the allocation of tasks to resources [6,10,11]. The focus of these studies has been human participants and

their use for effective resource allocation. The need for robots and human participants to collaboratively work forms an important part of human-automation studies [12,21]. The need for such interactions has been discussed in BPA [1,19]. In this work, we consider distinct types of resources and their characteristics and their interactions with human participants to support effective automation and allocation.

Optimal allocation of tasks to human participants has been widely studied in different domains such as IT service delivery [4,5]. Task allocation to human participants supporting multiple objectives has been an important area of work for flexible business process executions [8,14]. In many of the previous studies, multi-objective optimization uses simulation or conventional constraint based optimization. In this work, we have defined our solution approach to account for the crucial interplay between human and automation agents. Further, we have explored the use of genetic algorithm to support optimization of multiple objectives.

5 Conclusion and Future Work

We offer a solution to the difficult problem of devising an RPA architecture. An approach based on genetic algorithms can be effective in generating useful design alternatives for an RPA architect. Evolutionary search can be employed to find optimal RPA architectures which simultaneously satisfies all objectives and constraints discussed in Sect. 3. Central to genetic algorithms is the representation of the solution. For example, binary representation can be used to constitute an RPA solution as a genotype of bit strings. This string represents a complete RPA setting, including all process instances, tasks in each process instance, and all resources including HA, IA, and RA. Assuming that we use non-dominated sorting algorithm (NSGA-II) [3] as evolutionary search at each generation, NSGA-II would sort the current population into a number of non-dominated fronts. The evolution process would continue until it would arrive at a specified number of generations. In the final generation, NSGA-II would return a set of Pareto optimal solutions as optimal RPA architectures. We also note that our approach is general enough to compute *optimal process architectures* (an allocation of resources to each task in a pool of process instances), but a detailed evaluation requires search through a somewhat different search space. This remains high on our list of priorities for future work.

References

1. van der Aalst, W.M.P., Bichler, M., Heinzl, A.: Robotic process automation. BISE, pp. 269–272 (2018)
2. Arias, M., Munoz-Gama, J., Sepúlveda, M.: Towards a taxonomy of human resource allocation criteria. In: Teniente, E., Weidlich, M. (eds.) BPM 2017. LNBIP, vol. 308, pp. 475–483. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_37

3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
4. Diao, Y., Heching, A.: Staffing optimization in complex service delivery systems. In: 7th International Conference CNSM 2011, Paris, France, October 24–28, 2011, pp. 1–9. IEEE (2011). http://ieeexplore.ieee.org/document/6104024/
5. Diao, Y., Heching, A.R., Northcutt, D.M., Wallace, R.: Service-delivery modeling and optimization. Interfaces **45**(3), 243–259 (2015)
6. Havur, G., Cabanillas, C., Mendling, J., Polleres, A.: Resource allocation with dependencies in business process management systems. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45468-9_1
7. Huang, Z., Lu, X., Duan, H.: Resource behavior measure and application in business process management. Expert Syst. Appl. **39**(7), 6458–6468 (2012)
8. Jiménez-Ramírez, A., Weber, B., Barba, I., Valle, C.D.: Generating optimized configurable business process models in scenarios subject to uncertainty. Inf. Softw. Technol. **57**, 571–594 (2015)
9. Kabicher-Fuchs, S., Mangler, J., Rinderle-Ma, S.: Experience breeding in process-aware information systems. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 594–609. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_38
10. Kumar, A., Dijkman, R., Song, M.: Optimal resource assignment in workflows for maximizing cooperation. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 235–250. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_20
11. Kumar, A., et al.: Dynamic work distribution in workflow management systems: How to balance quality and performance. J. Manage. Inf. Syst. **18**(3), 157–194 (2002)
12. Parasuraman, R., Sheridan, T.B., Wickens, C.D.: A model for types and levels of human interaction with automation. IEEE Trans. Syst. Man Cybern. Part A **30**(3), 286–297 (2000)
13. Pika, A., Leyer, M., Wynn, M.T., Fidge, C.J., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Mining resource profiles from event logs. ACM Trans. Management Inf. Syst. **8**(1), 1:1–1:30 (2017)
14. Jiménez-Ramírez, A., Barba, I., del Valle, C., Weber, B.: Generating multi-objective optimized business process enactment plans. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 99–115. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_7
15. Jimenez-Ramirez, A., Reijers, H.A., Barba, I., Del Valle, C.: A method to improve the early stages of the robotic process automation lifecycle. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 446–461. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_28
16. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies (2012)
17. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education (2010)
18. Scheepers, R., Lacity, M.C., Willcocks, L.P.: Cognitive automation as part of deakin university's digital strategy. MIS Q. Executive **17**(2), 89–107 (2018)
19. Sindhgatta, R., ter Hofstede, A.H.M., Ghose, A.: Resource-based adaptive robotic process automation. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 451–466. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_28

20. Sokolova, M., Japkowicz, N., Szpakowicz, S.: Beyond accuracy, f-score and ROC: a family of discriminant measures for performance evaluation. In: Sattar, A., Kang, B. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 1015–1021. Springer, Heidelberg (2006). https://doi.org/10.1007/11941439_114
21. Vagia, M., Transeth, A.A., Fjerdingen, S.A.: A literature review on the levels of automation during the years. what are the different taxonomies that have been proposed? Appl. Ergon. **53**, 190–202 (2016)