Stochastic Resource Allocation with Time Windows

Yang Li[1] and Bin Xin[2](✉)

[1] Beijing Institute of Electronic System Engineering, Beijing 100854, China
[2] School of Automation, Beijing Institute of Technology, Beijing 100081, China
brucebin@bit.edu.cn

Abstract. The stochastic resource allocation problem with time windows (SRAPTW) refers to a class of combinatorial optimization problems which are aimed at finding the optimal scheme of assigning resources to given tasks within their time windows. In SRAPTW, the capability of resources to accomplish tasks is quantitatively characterized by probability. The expected allocation scheme should include not only the task-resource pairings but also their allocation time. This paper formulates SRAPTW as a nonlinear mixed 0–1 programming problem with the objective of maximizing the reward of completing specified tasks. Then, a general encoding/decoding method is proposed for the representation of solutions, and several different problem-solving methodologies are presented and compared. Results of computational experiments show that the utilization of SRAPTW-specific knowledge can bring in excellent performance, and a constructive heuristic combining maximal marginal return strategy and maximal probability strategy has remarkable advantages, especially in larger-scale cases.

Keywords: Stochastic Resource Allocation · Time Windows · Constructive Heuristic

1 Introduction

The stochastic resource allocation problem with time windows (SRAPTW) refers to a class of combinatorial optimization problems which are aimed at finding the optimal scheme of assigning resources to given tasks within certain time windows. As a complex variant of the stochastic resource allocation problems (SRAP) [1, 2], a prominent feature of SRAPTW is that the capability of resources to accomplish tasks is quantitatively characterized by probability within certain time windows.

SRAPTW widely exists in the control of complex systems which involve the allocation of multiple sensors and actuators for various control tasks. When sensors or actuators are used for moving targets especially those time-critical ones, the time windows of resource allocation are one of the key factors to check the feasibility of task scheduling and execution.

Previous studies on SRAP, such as sensor-target assignment problem and weapon-target assignment problem, were focused on the allocation of resources to tasks without time windows [1–8]. In fact, time windows widely exist in the resource allocation problems from diverse fields, e.g., emergency and rescue [9], communication channel management [10], traffic resources management [11], transportation and logistics [12–14], defense resources allocation [15]. Unlike common resource allocation problems involving time windows, the success probability of resources to accomplish tasks is time-dependent in SRAPTW. Therefore, the time windows in SRAPTW not only bring constraints, but also relate to the distribution of success probability, which means the task performance of resources is sensitive to the execution time of tasks. In this sense, solving SRAPTW involves the matching of resources and tasks as well as determining the execution time of tasks, so as to optimize the overall task performance.

This paper is aimed at formulating SRAPTW and proposing efficient strategies and algorithms for solving it. The contribution of the paper is summarized as follows:

1) Formulate the SRAPTW as a nonlinear mixed 0–1 programming problem.
2) Provide a general encoding/decoding method for representing solutions and designing search algorithms to solve SRAPTW.
3) Propose four different SRAPTW algorithms which combine knowledge-dependent constructive heuristic and random search. Conduct comparative computational experiments to identify their pros and cons.

The rest of the paper is structured as follows. Section 2 presents the problem formulation. Section 3 presents the proposed algorithms. Section 4 presents computational experiments. Section 5 concludes the paper.

2 Problem Formulation

2.1 Problem Description

Stochastic resource allocation (SRA) refers to the scenarios in which the capability of a resource (e.g., a person or a robot) to complete tasks is characterized by probabilities. An allocation scenario of m resources and n tasks is considered. The probability of the ith resource ($i = 1, 2, \cdots, m$) to complete the jth task ($j = 1, 2, \cdots, n$) also depends on the allocation time. So, if the probability, denoted by Pij(t), is lower than certain threshold or even becomes 0, the allocation will be abandoned. Generally, the allocation of resources to tasks needs to be handled within specified time windows. Thus, the problem is an SRA problem with time windows (SRAPTW). Assume that each task, once accomplished, will bring a reward. Denote the reward of the jth task by v_j. The objective of SRAPTW is to maximize the total reward of allocating resources to tasks within specified time windows. Besides, resources when assigned usually have to satisfy some constraints. For example, for one resource, there will be some time interval between one allocation and its next, as well as an allowable maximal number of tasks to be allocated. For each task, it is allowed that multiple resources can be used but each resource can be used only once.

2.2 Optimization Model

The SRAPTW can be formulated as an optimization problem. Assume that the execution of tasks about each resource is mutually independent. Then, the objective of the problem can be presented as follows:

$$\max J_1(X,T) = \sum\nolimits_{j=1}^{n} v_j\Big(1 - \prod\nolimits_{i=1}^{m}\big(1 - p_{ij}(t_{ij})x_{ij}\big)\Big) \tag{1}$$

where $J_1(X,T)$ represents the objective function, X and T are the decision matrices, $X = \left[x_{ij}\right]_{m\times n}$ represents the resource-task allocation relation, $T = \left[t_{ij}\right]_{m\times n}$ represents the task execution time for each resource-task pair when allocated, and $P = \left[p_{ij}(t_{ij})\right]_{m\times n}$ represents the success probability that a resource can accomplish a task at specific time. A detailed description of other parameters is presented in Table 1.

The objective shown in Eq. (1) can be equivalently converted to the following

$$\min J_2(X,T) = \frac{\sum_{j=1}^{n} v_j \prod_{i=1}^{m}\big(1 - p_{ij}(t_{ij})x_{ij}\big)}{\sum_{j=1}^{n} v_j} \tag{2}$$

where $J_2(X,T)$ is a normalized objective function.

Table 1. Explanation of Model Parameters.

Symbol	Explanation
v_j	Reward of accomplishing task j
x_{ij}	$x_{ij} = 1$ means that task j is allocated to resource i; $x_{ij} = 0$ otherwise
t_{ij}	The task execution time for resource i to execute task j
$p_{ij}(t_{ij})$	The success probability that resource i can accomplish task j at the time t_{ij}
TW_{ij}	The time interval for resource i to execute task j
t_{ij}^{-}	The lower bound of the time interval for resource i to execute task j
t_{ij}^{+}	The upper bound of the time interval for resource i to execute task j

The following constraints have to be satisfied.

Range of Decision Variables. It is obvious that the allocation variable is of 0–1 type and the allocation time should be determined within specified time window.

$$x_{ij} \in \{0, 1\}, \forall i \in \{1, 2, \cdots, m\}, \forall j \in \{1, 2, \cdots, n\} \tag{3}$$

$$t_{ij} \in [t_{ij}^{-}, t_{ij}^{+}], \forall i \in \{1, 2, \cdots, m\}, \forall j \in \{1, 2, \cdots, n\} \tag{4}$$

Maximal Number of Tasks to be Allocated for Each Resource. For resource i, the maximal number of tasks which can be allocated to it should not exceed n_i.

$$\sum\nolimits_{j=1}^{n} x_{ij} \le n_i, \forall i \in \{1, 2, \cdots, m\} \tag{5}$$

Maximal Number of Resources to be Allocated for Each Task. For task j, the maximal number of resources which can be allocated to it should not exceed m_j.

$$\sum\nolimits_{i=1}^{m} x_{ij} \leq m_j, \forall j \in \{1, 2, \cdots, n\} \tag{6}$$

Time Interval for Each Resource to Execute Two Tasks. For resource i, the execution of any two tasks assigned to it should be separated at least τ_i.

$$|t_{ij} - t_{ik}| \geq \tau_i, \forall i \in \{1, 2, \cdots, m\}, \forall j, k \in \{1, 2, \cdots, n\}, j \neq k \tag{7}$$

To sum up, the optimization model for SRAPTW can be formulated as follow:

$$\min J_2(X, T), \text{s.t.}(3) \sim (7).$$

Obviously, SRAPTW is a nonlinear mixed-variable programming problem.

2.3 Problem Analysis

From the problem formulation shown above, the following properties can be derived:

Property 1: The more resources are allocated without violating any constraints, the better the objective value will be. Stated another way, for any X, if more x_{ij} can become 1, $J_2(X,T)$ can be improved.

Property 2: For any X, the larger $p_{ij}(t_{ij})$ is, the better the objective value.

Property 3: If all $t_{ij}^* = \text{argmin}(p_{ij}(t_{ij}))$,$\forall i \in \{1, 2, \cdots, m\}$, $\forall j \in \{1, 2, \cdots, n\}$, do not have conflicts w.r.t. constraints (7), then $t_{ij} = t_{ij}^*$ will be a necessary condition for any $x_{ij} = 1$.

As demonstrated later, these properties are beneficial to design efficient problem-specific strategies or operators to solve SRAPTW.

3 Algorithm Design

3.1 Solution Representation

The decision variables X (binary valued) and T (real valued) are straightforward representation of solutions to SRAPTW. However, it will be more convenient to utilize the problem-domain knowledge in SRAPTW by use of permutation-based encoding and decoding schemes.

According to Property 1, for a feasible X which does not violate the constraints (3), (5) or (6), changing more x_{ij} from 0 to 1 will bring the improvement of the objective value. In this sense, for an empty X (all zero elements), the order of making each x_{ij} become one while ensuing constraint satisfaction can be a key factor to generate a feasible and even high-quality solution. In other words, the permutation of all resource-task pairs can be used as a main component of solution representation.

For the execution time regarding each pair, we can determine the value of each t_{ij} from its time window according to certain rules, e.g., unbiased random sampling or biased heuristic selection.

To sum up, the permutation of all resource-task pairs and corresponding execution time can be used to represent a solution to SRAPTW. The detailed encoding-decoding scheme is described as follows.

Encoding Scheme. For a resource-task pair (briefly called an RT pair) denoted by *i-j* ($i \in \{1, 2, \cdots, m\}, j \in \{1, 2, \cdots, n\}$), we use the number $n \times (i-1) + j$ to represent it. Then, the permutation of all RT pairs is formally a permutation of the integers from 1 to $m \times n$. The matrix of the execution time regarding each RT pair, together with the permutation, constitutes the encoding scheme.

For example, for an SRAPTW of 2 resources and 2 tasks with time windows $TW_{11} = [0, 0.1]$, $TW_{12} = [0.3, 0.4]$, $TW_{21} = [0.2, 0.3]$ and $TW_{22} = [0.5, 0.9]$, the permutation 4-1-3-2 and the matrix of execution time [0.05, 0.38; 0.25, 0.78] represent a solution.

Decoding Scheme. For a given permutation Pe and a matrix of execution time$T = \left[t'_{ij}\right]_{m \times n}$, $x_{ij} = 1$ for each RT pair will be checked, in the order indicated by Pe, to see if it violates constraints (5), (6) or (7). If no constraints violation occurs, then let $x_{ij} = 1$ and $t_{ij} = t'_{ij}$; otherwise, let $x_{ij} = 0$ and $t_{ij} = \infty$.

Based on the above representation scheme, a feasible solution can be generated by decoding after determining a permutation of all RT pairs and a execution time matrix. Both the permutation and the execution time matrix can be generated in different ways. For example, they can be generated by random sampling or constructive heuristics.

3.2 Constructive Heuristics

Constructive heuristics represent a philosophy of straightforwardly generating a solution to a complex problem by determining its components step by step, rather than implementing samplings in solution space as widely adopted in various search algorithms such as improvement heuristics and metaheuristics [16–18]. In comparison with search algorithms, constructive heuristics do not need function evaluations to improve solutions in an iterative way. To design efficient constructive heuristics, the permutation of all RT pairs and the execution time matrix can be generated by utilizing SRAPTW-specific knowledge, e.g., the three properties shown in Subsect. 2.3. To utilize Properties 2 and 3, the execution time will be set to t^*_{ij} if it does not violate any constraint. The permutation can be constructed by using the rule-based method based on maximal marginal return (MMR) [6, 19]. The marginal return means the improvement of the objective value with reference to its current value brought by choosing one RT pair. MMR is a rule for constructing a solution incrementally, and in each step, the RT pair with maximal marginal return without constraint violation will be added into the allocation scheme. In fact, since the MMR-based procedure can generate a complete solution directly, it is unnecessary to get the permutation as a preliminary which is de facto implicated in the procedure. To save space, the rationale of MMR will not be presented here in more detail, and interested readers may refer to the reference [6]. Instead, we provide the pseudo-code of the procedure in **Procedure 1**.

Procedure 2. MMR(T)

```
Input: The initial execution time matrix T
Output: allocation scheme X, final execution time matrix T, objective value F
X = $O_{m\times n}$ ;    % All zeros
CS1 = $O_{m\times 1}$ ; % The flag of constraint 5
CS2 = $O_{n\times 1}$ ; % The flag of constraint 6
Temp = inf * ones(m,n) ; % All infinities
cnt = 0 ;
For i = 1 to m
   For j = 1 to n
      r(i,j) = $v_j * p_{ij}(t_{ij})$ ; % Calculate the initial marginal return for each RT
pair
   End
End
For k = 1 to m*n
    [w,g] = max(r) ; % Get the maximum for each column of the matrix r
                  % w records the maximum values
                  % g records the row number of each maximum in each column
    [u,h] = max(w) ; % Get the maximum of the vector w
                  % u records the maximum value
                  % g records the column number of the maximum in w
    i = g(h) ; % the $h$th element of the vector $g$
    j = h ;       % i-j indicates the RT pair with the maximal marginal return
    If CS1(i) < $n_i$ && CS2(j) < $m_j$ && min(|$t_{ij}$-Temp(i,:)|) $\geq \tau_i$
         X(i,j) = 1; CS1(i) = CS1(i) + 1 ; CS2(j) = CS2(j) + 1 ;
          Temp(i,CS2(j)) = $t_{ij}$ ;
          If CS1(i) == $m_i$  cnt = cnt +1; end
         If CS2(j) == $n_j$  cnt = cnt +1; end
         If cnt == m+n break ; end
         r(:,j) = r(:,j) * (1 - $p_{ij}(t_{ij})$) ;
    End
    r(i,j) = - inf ;
End
F = 0 ; q = ones(n,1) ;
For j=1 to n
    For i = 1 to m
         q(j) = q(j) * ( 1 - X(i,j) * $p_{ij}(t_{ij})$ ) ;
    End
    F = F + $v_j$ * q(j) ;
End
T = (1./X) * T ; F = F / sum(v) ;
Return X, T, F
```

Since the core of this constructive heuristic is the MMR-based procedure combined with the execution time setting based on maximal probability, we name it by **MMR-MP**. The time complexity of determining all the execution time with maximal probability is

$O(mnl)$ where l represents the number of time samplings in each time window. From the pseudo-code of MMR-MP, the worst-case time complexity of MMR procedure is $O(m^2n^2)$. So, the time complexity of MMR-MP is $O(m^2n^2 + mnl)$.

3.3 Random Search

Different from the constructive heuristic MMR-MP, the random search here refers to certain unbiased search process to find better solutions. Since any solution involves two parts, i.e., the permutation and the execution time matrix, we have three different strategies to implement random search:

Strategy 1: MMR for permutation (*implicit*) & Random sampling of execution time
Strategy 2: Random permutation & Execution time with maximal probability
Strategy 3: Random permutation & Random sampling of execution time

Random permutation can be achieved in MATLAB by the command randperm(n $*$ m) while the random sampling of execution time can be conducted by implementing $t_{ij} = t_{ij}^- + (t_{ij}^+ - t_{ij}^-) \times rand$ where *rand* is a random number following the uniform distribution in (0,1). The three versions of random search resulting from these strategies are named **MMR-RS**, **RP-MP** and **RP-RS**, respectively. In fact, similar to MMR-MP, both MMR-RS and RP-MP utilize SRAPTW-specific knowledge while maintaining certain randomness. In contrast, RP-RS is a pure random search without search bias. Denote the allowable number of random samplings (function evaluations) by N. Then, the time complexity of MMR-RS, RP-MP and RP-RS is $O(Nm^2n^2)$, $O((N + l)mn)$ and $O(Nmn)$, respectively.

4 Computational Experiments

The goal of the computational experiments is to validate the performance of the proposed algorithms including MMR-MP, MMR-RS, RP-MP and RP-RS. All tests are conducted on a Lenovo Laptop X1 with Intel Core i7 CPU (1.8GHz) and 16GB RAM. For each SRAPTW instance, the random search algorithms including MMR-RS, RP-MP and RP-RS will run 25 times for making a statistical analysis of results.

4.1 Experiments Configuration

Test case generator: The task rewards are generated by following a uniform distribution $v_j \sim U(1{,}1 + \delta)$ where $U(a{,}b)$ represents a uniform distribution in the interval $(a{,}b)$, and δ is a control parameter. All time windows TW_{ij} are randomly generated from the interval (0,1). The success probability $p_{ij}(t_{ij}) \sim N(\mu_{ij}, \sigma_{ij}^2)$ where $N(\mu, \sigma^2)$ represents a Gauss distribution with mean μ and standard deviation σ. Let $\mu_{ij} \sim U(0{,}1)$ and $(\sigma_{ij} - \varepsilon) \sim U(0, t_{ij}^+ - t_{ij}^-)$ where ε is a positive number for regulating the distribution of success probability. A larger ε implies a smaller change of success probability within corresponding time window. For constraints (5) and (6), $m_i = \lceil \alpha \times rand_i \rceil$, $n_j = \lceil \beta \times rand_j \rceil$ where $\lceil \cdot \rceil$ is the ceiling operator, and α and β are control parameters, and $rand_i, rand_j \sim U(0{,}1)$. For constraint (7), $\tau_i \sim U(\tau_i^-, \tau_i^+)$ where τ_i^- and τ_i^+ are control parameters.

The following cases are generated.

Case 1: m=5, n=5, $\delta = 0, \varepsilon = 1, \alpha = 1, \beta = 1, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 2: m=5, n=5, $\delta = 1, \varepsilon = 1, \alpha = 1, \beta = 1, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 3: m=5, n=5, $\delta = 1, \varepsilon = 0.7, \alpha = 1, \beta = 1, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 4: m=5, n=5, $\delta = 9, \varepsilon = 1, \alpha = 1, \beta = 1, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 5: m=5, n=5, $\delta = 9, \varepsilon = 0.7, \alpha = 2, \beta = 2, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 6: m=5, n=10, $\delta = 9, \varepsilon = 0.4, \alpha = 2, \beta = 2, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 7: m=10, n=5, $\delta = 9, \varepsilon = 0.1, \alpha = 2, \beta = 2, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 8: m=10, n=5, $\delta = 9, \varepsilon = 0.4, \alpha = 2, \beta = 2, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 9: m=10, n=10, $\delta = 9, \varepsilon = 0.1, \alpha = 2, \beta = 2, \tau_i^- = 0.1, \tau_i^+ = 0.4$
Case 10: m=20, n=10, $\delta = 9, \varepsilon = 0.1, \alpha = 2, \beta = 2, \tau_i^- = 0.1, \tau_i^+ = 0.4$

In each case, ten different instances are generated by following the probability distribution preset in the test case generator. Detailed data about the total 100 instances and related results will be released online and can be acquired by contacting the authors.

Parameter Setting: For all random search algorithms, the maximal number of function evaluations (samplings) is set as $10 \times m \times n$. For determining the time for the maximal probability in each time window, even samplings within [0,1] are conducted and the minimal spacing is set to 0.001.

4.2 Performance Comparison

For each instance in each case, the results of all algorithms, i.e., obtained objective values in 25 runs, will be compared by Wilcoxon ranksum test with 0.05 significance level. For two algorithms A and B, the indicator of the ranksum test $h = 1$ means the performance of A is significantly different from that of B. In this case, if the mean of the results obtained by A is smaller (larger) than that by B, then A gets a score of $+1$ (-1) and B gets a score of -1 (+1); otherwise, both A and B get zero. In each case, the highest and lowest scores are 30 and –30, respectively. The scores of four algorithms in 10 cases (totally 100 instances) are summarized in Table 2. To save space, detailed results for each instance will not be presented. The average runtime of the four algorithms in different cases is shown in Fig. 1.

Table 2. Statistical results of algorithm comparison (scores)

Algorithm	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
RP-RS	−16	−18	−24	−26	−24	−29	−30	−30	−30	−30
RP-MP	**25**	**28**	**27**	**25**	**15**	−11	9	12	−4	−10
MMR-RS	8	4	−3	−5	9	**22**	−7	−3	12	10
MMR-MP	−17	−14	0	6	0	18	**28**	**21**	**22**	**30**

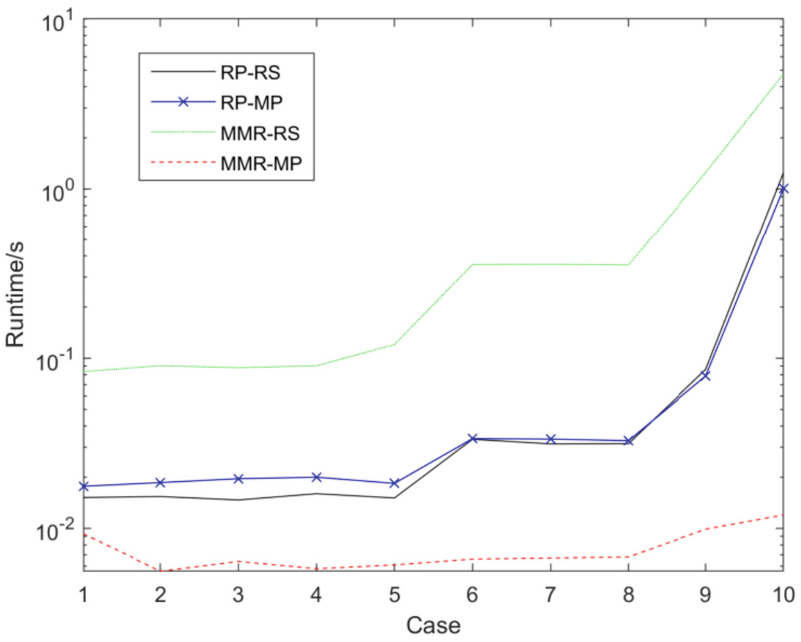

Fig. 1. The average runtime of four algorithms in different cases.

From Table 2, the following results can be found:

1) RP-MP has obvious advantages in Cases 1, 2, 3 and 4; in contrast, it is not a good solver in Cases 6 and 10;
2) MMR-MP takes the first place in Cases 7, 8, 9 and 10 (especially wins all in the larger-scale Case 10). However, it was defeated by RP-MP in Cases 1, 2, 3, 4 and 5 and performs poorly in Cases 1 and 2.
3) MMR-RS gets the highest score in Case 6, and shows its advantage over RP-MP and RP-RS but loses to MMR-MP in Cases 9 and 10.
4) RP-RS has no advantage in almost all cases. In Cases 7, 8, 9 and 10, it was completely defeated by the other algorithms. This implies that pure random search without use of problem-specific knowledge is inefficient, though it can theoretically cover the solution space and find the optimal solution with sufficient samplings.
5) It seems that the MMR strategy generally has dominant advantages in larger-scale cases, which is supported by (a) the superiority of MMR-MP over RP-MP in Cases 6 to 10, (b) the superiority of RP-MP over MMR-MP in Cases 1 to 5, and (c) the superiority of MMR-RS over RP-RS. The comparison implies that the permutation of RT pairs will play a more important role in determining the quality SRAPTW solutions as the scale of SRAPTW increases.
6) It seems that the MP strategy has advantages in smaller-scale cases, which is supported by (a) the superiority of MMR-MP over MMR-RS in Cases 7 to 10, (b) the superiority of MMR-RS over MMR-MP in Cases 1, 2 and 5, and (c) the superiority of RP-MP over RP-RS. Obviously, MP plays a crucial role in generating high-quality

solutions for smaller-scale cases, and it is also a necessary strategy for guaranteeing the performance of MMR-MP.

The average time cost of the four algorithms shown in Fig. 1 is consistent with the analysis about computational complexity in Sect. 3. Obviously, MMR-MP has the lowest time cost in all cases. The time costs of RP-RS and RP-MP are very close, which implies that the cost of obtaining the maximal probabilities (regulated by the parameter l) is negligible as compared to that of determining the permutation of RT pairs (regulated by the number of function evaluations N).

5 Conclusion

A mathematical programming model was built for the stochastic resource allocation problem with time windows. A general encoding/decoding scheme was proposed for representing SRAPTW solutions. Constructive heuristics and random search algorithms were proposed to solve SRAPTW. Comparative experiments show that MMR-MP and RP-MP have obvious advantages over MMR-RS and RP-RS in most cases. MMR-MP and RP-MP share the similarity in the use of preset time under maximal probability strategy, which implies the superiority and necessity of utilizing the problem-specific knowledge embodied in Properties 3 and 4. In some instances, MMR-RS is the single winner, which reflects that MMR is an effective strategy for determining the order of resource-task pairs in allocation process. In contrast, RP-RS did not take the first place in any instances, and also performs the worst in most cases. The failure of RP-RS as compared with its competitors confirms that pure random search without utilization of problem knowledge is not a satisfactory choice for algorithm design. Generally, misuse of inaccurate knowledge in problems may cause the loss of optimality or lead to local optima. On the other hand, no use of problem-specific knowledge usually results in slow convergence of iterative search process. A delicate balance between the two aspects may bring better strategies for solving complex SRAPTW. On the basis of the proposed general solution representation scheme, how to design advanced neighborhood search operator or meta-heuristics [20–22] also deserves further studies in the future.

Acknowledgement. We would like to thank the National Outstanding Youth Talents Support Program (Grant 61822304), the Basic Science Center Programs of NSFC (Grant 62088101), the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100), and the Shanghai Municipal Commission of Science and Technology Project (19511132101).

References

1. Castanon, D.A., Wohletz, J.M.: Model predictive control for stochastic resource allocation. IEEE Trans. Autom. Control **54**(8), 1739–1750 (2009)
2. Fan, G.M., Huang, H.J.: Scenario-based stochastic resource allocation with uncertain probability parameters. J. Syst. Sci. Complexity **30**(2), 357–377 (2017)
3. Li, J., Xin, B., Pardalos, P.M., Chen, J.: Solving bi-objective uncertain stochastic resource allocation problems by the s-based risk measure and decomposition-based multi-objective evolutionary algorithms. Ann. Oper. Res. **296**(1–2), 639–666 (2021)

4. Cassandras, C.G., Dai, L., Panayiotou, C.G.: Ordinal optimization for a class of deterministic and stochastic discrete resource allocation problems. IEEE Trans. Autom. Control **43**(7), 881–890 (1998)
5. Xin, B., Chen, J., Zhang, J., Dou, L.H., Peng, Z.H.: An efficient rule-based constructive heuristic to solve dynamic weapon-target assignment problem. IEEE Transactions on Systems Man and Cybernetics Part A - Systems and Humans **41**(3), 598–606 (2011)
6. Xin, B., Wang, Y.P., Chen, J.: An efficient marginal-return-based constructive heuristic to solve the sensor-weapon-target assignment problem. IEEE Transations on Systems, Man Cybernetics-Systems **49**(12), 2536–2547 (2019)
7. Asadpour, A., Wang, X., Zhang, J.: Online resource allocation with limited flexibility. Manage. Sci. **66**(2), 642–666 (2020)
8. Wang, Y.P., Xin, B., Chen, J.: An adaptive memetic algorithm for the joint allocation of heterogeneous stochastic resources. IEEE Transactions on Cybernetics **52**(11), 11526–11538 (2022)
9. Luan, D., Liu, A., Wang, X., Xie, Y., Wu, Z.: Robust two-stage location allocation for emergency temporary blood supply in postdisaster. Discrete Dynamics in Nature and Society **2022**, Article ID 6184170
10. Bankov, D., Khorov, E., Lyakhov, A., Famaey, J.: Resource allocation for machine-type communication of energy-harvesting devices in Wi-Fi HaLow networks. Sensors **20**, 2449 (2020)
11. Jiang, B., Fan, Z.P.: Optimal allocation of shared parking slots considering parking unpunctuality under a platform-based management approach. Transp. Res. Part E **142**, 102062 (2020)
12. Puglia Pugliesea, L.D., Ferone, D., Macrinac, G., Festa, P., Guerriero, F.: The crowd-shipping with penalty cost function and uncertain travel times. Omega **115**, 102776 (2023)
13. Wang, Y., Wang, X., Guan, X., Li, Q., Fan, J., Wang, H.: A combined intelligent and game theoretical methodology for collaborative multicenter pickup and delivery problems with time window assignment. Appl. Soft Comput. **113**, 107875 (2021)
14. Hoogeboom, M., Adulyasak, Y., Dullaert, W., Jaillet, P.: The robust vehicle routing problem with time window assignments. Transp. Sci. **55**(2), 395–413 (2021)
15. Almeida, R., Gaver, D.P., Jacobs, P.A.: Simple probability-models for assessing the value of information in defense against missile attack. Nav. Res. Logist. **42**(4), 535–547 (1995)
16. Meng, K., Chen, C., Xin, B.: MSSSA: a multi-strategy enhanced sparrow search algorithm for global optimization. Frontiers of Information Technol. Electronic Eng. **23**(12), 1828–1847 (2022)
17. Gao, G.Q., Mei, Y., Jia, Y.H., Browne, W.N., Xin, B.: Adaptive coordination ant colony optimization for multipoint dynamic aggregation. IEEE Trans. Cybernetics **52**(8), 7362–7376 (2022)
18. Guo, M., Xin, B., Chen, J., Wang, Y.P.: Multi-agent coalition formation by an efficient genetic algorithm with heuristic initialization and repair strategy. Swarm and Evolutonary Computation **55** (2020)
19. Gülpınar, N., Çanakoglu, E., Branke, J.: Heuristics for the stochastic dynamic task-resource allocation problem with retry opportunities. Eur. J. Oper. Res. **266**, 291–303 (2018)
20. Ding, Y.L., Xin, B., Zhang, H., Chen, J., Dou, L.H., Chen, B.M.: A memetic algorithm for curvature-constrained path planning of messenger UAV in air-ground coordination. IEEE Trans. Autom. Sci. Eng.Autom. Sci. Eng. **19**(4), 3735–3749 (2022)
21. Qi, M.F., Dou, L.H., Xin, B.: 3D smooth trajectory planning for UAVs under navigation relayed by multiple stations using Bezier curves. Electronics **12**(11), 2358 (2023)
22. Jiao, K.M., Chen, J., Xin, B., Li, L., Zhao, Z.X., Zheng, Y.F.: A framework for co-evolutionary algorithm using Q-learning with meme. Expert Systems With Applications **225** (2023)