Mixed integer programming and adaptive problem solver learned by landscape analysis for clinical laboratory scheduling

Keyao Wang \textsuperscript{a,b}, Bo Liu \textsuperscript{b*}

\textit{a} School of Economics and Management, Beihang University, Beijing, China  
\textit{b} Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

**Abstract:** This paper attempts to derive a mathematical formulation for real-practice clinical laboratory scheduling, and to present an adaptive problem solver by leveraging landscape structures. After formulating scheduling of medical tests as a distributed scheduling problem in heterogeneous, flexible job shop environment, we establish a mixed integer programming model to minimize mean test turnaround time. Preliminary landscape analysis sustains that these clinics-orientated scheduling instances are difficult to solve. The search difficulty motivates the design of an adaptive problem solver to reduce repetitive algorithm-tuning work, but with a guaranteed convergence. Yet, under a search strategy, relatedness from exploitation competence to landscape topology is not transparent. Under strategies that impose different-magnitude perturbations, we investigate changes in landscape structure and find that disturbance amplitude, local-global optima connectivity, landscape’s ruggedness and plateau size fairly predict strategies’ efficacy. Medium-size instances of 100 tasks are easier under smaller-perturbation strategies that lead to smoother landscapes with smaller plateaus. For large-size instances of 200-500 tasks, extant strategies at hand, having either larger or smaller perturbations, face more rugged landscapes with larger plateaus that impede search. Our hypothesis that medium perturbations may generate smoother landscapes with smaller plateaus drives our design of this new strategy and its verification by experiments. Composite neighborhoods managed by meta-Lamarckian learning show beyond average performance, implying reliability when prior knowledge of landscape is unknown.

**Keywords:** healthcare, clinical laboratory, scheduling, mixed integer programming, adaptive problem solver, distributed scheduling, landscape analysis.

**Highlights**

- Formulate scheduling of medical tests as a distributed scheduling problem in heterogeneous, flexible job shop environment, and establish a mixed integer programming model
- Design an adaptive problem solver under the guidance of landscape analysis
- Define a new search strategy to be well suited to the instance by leveraging landscape characteristics

\textsuperscript{*} Corresponding author. \textit{E-mail address:} bliu@amss.ac.cn (B. Liu)
1. Introduction

Clinical laboratories in hospitals must perform medical tests on tube-clinical specimens in a timely way so that medical personnel are enabled to make decisions for patients. Prompt testing of specimens and test results can be urgent. Clinical lab results can support as much as 70% of clinical decisions [1]. In one study, laboratory results were responsible for up to 80% of patient facts required for real-time medicine for critical care in emergency departments [2]. Poor laboratory efficiency delays results reporting, significantly impacts follow-up diagnoses, and hazards patients’ well-being at life-threatening moments in their lives [3].

Leveraging clinical laboratory efficiency, especially to decrease test turnaround time, has gone far but without fully implementing optimal scheduling. Innovation in rapid testing methods and in stand-alone high-throughput instruments and their layout has been successfully incorporated, and yet less than optimal scheduling could impede whole laboratory efficacy.

In our field investigations in a clinical laboratory at China-Japan Friendship Hospital in Beijing, we observed how staff laboriously schedule tasks daily with minimal automated assistance. Situations challenged them on how to optimally operate their complex, busy laboratory to achieve higher throughput. Extremely dynamic testing volume, ranging into hundreds of tubes per shift, was common. Round-the-clock lab operation meant multiple shifts needing to be completed. Instruments arrayed in a complex instrument-function topology required staff to figure out a time-saving travel path for each tube passing from pre-analysis to results reporting.

Improving turnaround by means of better clinical laboratory scheduling, and doing so without incurring capital costs from new hardware, seems a worthy startpoint. If sophisticated scheduling tools can automatically arranged a feasible sequence for instrument use with testing tubes, it could improve throughput and prevent medical burnout.

We, consequently, sought a mathematical model and an adaptive problem solver. However, to date, a mathematical formulation expressing the quantifiable relationship among task, resource and capacity in a whole clinical laboratory does not exist [4]. Some steps were confined to scheduling tests on a solo workstation [5], or on staff assignments across workstations [6].

As domain-independent problem solving, less dependent on human intervention, is needed for alleviating medical staff’s cognitive burden and for guaranteeing convergence, we searched for an adaptive problem solver that could match designated problems with effective algorithms. However, unawareness of relatedness between the search strategies and the problem’s landscape structures could impede identification of effective strategies. A quantifiable characterization of landscape for the clinic-orientated scheduling problem is rather unclear; how landscape topology changes under different strategies, i.e., reflecting instance hardness, remains largely unknown; landscape features that highly influence performance are undeterminable. The central issue was that relatedness from characteristics of the best search strategy to characteristics of landscape is not transparent.

To answer challenges in formulation for a real-practice clinical laboratory scheduling, and to present an adaptive problem solver by leveraging instances’ landscape features were the direction of our study. On the way to this destination, we took up three challenging questions.

Can an explicit mathematical formulation be derived to quantitatively portray complicated relational structures among process, instrument, and capacity in clinical laboratory?
Can a generic solver be designed under the guidance of landscape analysis to make an informed decision on which search strategy performs better on a particular instance?

When an instance is hard for extant approaches at hand, can a new search strategy be defined to be well suited to the instance by leveraging its landscape characteristics?

Our contributions are four-fold, and Fig. 1 illustrates our approach.

1. Mathematical formulation for a real-practice clinical laboratory scheduling
2. Benchmark for normal and extreme scenarios
3. Learning an adaptive problem solver by landscape analysis
4. Validation on large-scale instances

For the first question, we formulated optimal decisions in clinical laboratory in a framework of Distributed scheduling in Heterogeneous and Flexible Job Shop Problem environment (D-HFJSP) (Section 3), and established a mixed integer programming model with the objective of minimizing mean of turnaround time (Section 4). To our knowledge, this is the first attempt to formulate a whole laboratory scheduling problem and provide a mixed integer programming model.

For the second question, by embodying syntactic and generative approaches, we presented a complementary adaptive problem solver within a framework of memetic algorithm. Several landscape analysis tools [7-9] were resorted to in the learning domain’s syntactic characteristics to suggest suitable strategies for a family of instances, while meta-Lamarckian learning [10] was relied upon for generating adaptable strategies for a single instance at runtime (Section 6). Here, strategies were referred to different neighborhoods in combinatorial space, which were deployed in memetic algorithm. The memetic algorithm was a collection of computationally found strategies based on scatter search, called SSMA (Section 5). SSMA’s novelty lies in (1) an efficient dispatching heuristic that represents solutions into feasible schedules, (2) an economic diversification generation that generates diversified, high-quality initial solutions through a new block-based Nawaz-Enscore-Ham heuristic, and (3) a simulated annealing-based improvement method that deploys meta-Lamarckian learning to choose a neighborhood from among composite neighborhoods. Utilization of both inter- and intra-instance features to infer the choice of a strategy is rather scarce. To our knowledge, our
work is the first attempt to understand the nature of landscapes for optimization in healthcare applications.

For the third question, a new search strategy for large-size instances was defined by exploiting what we have learned about search spaces. We theoretically proved expected inter-neighbor distances that measured each strategy’s competences in exploitation over search space. After experimentally investigated landscape structural changes under each strategy that imposed perturbations of different magnitudes, we found that, for large-size instances (200 to 500 tasks), the extant strategies at hand with either larger-perturbations or smaller-perturbations were faced with more rugged landscapes, including multiple scattered funnels (big valleys) with larger attractor or sink plateaus that made the search stagnate. Under moderate perturbations, landscapes were smoother and had smaller attractor plateaus, suggesting large-size instances were easily optimizable under moderate perturbations. This advocated for designing a new strategy, Block-based Insert, to search for optima with moderate momentum and to easily escape from attractor plateaus and traverse multiple funnels.

Through comprehensive experiments, the adaptive problem solver’s efficacy was studied (Section 7). Our benchmark of 20 toy instances (2, 4, 6, and 8 tasks) and 150 realistic instances (100, 200, 300, 400, and 500 tasks) reflects normal and extreme scenarios. Toy and medium-size instances (100 tasks) were easier under Swap-based strategy that generated smaller perturbations and led to smoother landscape with smaller plateaus. It was validated that Swap achieved the same or better results as Gurobi Optimizer on toy instances but was performing best on medium-size instances. For large-size instances (200 to 500 tasks), the new strategy that allowed a quick step over multiple funnels and plateaus performed best. On nearly all instances, meta-Lamarckian learning showed beyond average performance, suggesting its reliability when no prior knowledge exists about the problem landscape.

While optimal scheduling is an obvious need in clinical decision support systems, a scheduling model and optimizer that are too complex for physicians is not a workable answer. Our work in applied modeling and generic solver is vital towards automated reasoning and scheduling for artificial intelligence applied to critical clinical testing. The potential for real-world implementation in the clinical laboratory would seem to lie in a mixed integer programming model, together with adaptive problem solver where landscape analysis and meta-Lamarckian learning tuned a memetic algorithm.

2. Related work

To actually derive a mathematical formulation and a generic solver for scheduling in clinical laboratory, our research was extended to a wide scope. Besides investigating scheduling applied to the clinical laboratory, we reviewed topics related to distributed scheduling, adaptive problem solving, and landscape analysis. Herein, we indicate what we adopted from the literature and what we proposed to extend.

**Scheduling in clinical laboratory** research finds test sequences processed on machine with respect to one or more objectives [11]. Since Aarts et al. [5] introduced scheduling theory to clinical chemistry, 25 years’ of effort seem confined to scheduling a solo workstation, or staff assignments across workstations. Aarts et al. [5] formulated pipettor dispatch in a chemistry workstation as a no-wait flow shop to minimize turnaround time, and Kim et al. [12] adopted a job shop model to improve throughput for a portable platform. van Merode et al. [13] described physician assignments as a mathematical programming model allowing physician preemption. Boyd and Savory [14] matched by genetic algorithm physicians of different skills with tasks. Elena et al. [6] presented a multi-period integer programming model to decrease total idle time and the number of physicians needed. To date,
mathematical formulation expressing quantifiable relationships among task, resource and capacity in a whole clinical laboratory does not exist [4]. By contrast, we described a mixed integer programming model for this purpose.

**Distributed scheduling** occurs in multiple separate factories or production lines. In addition to assignment of jobs to machines and sequencing of jobs on machine, the existence of separate factories has prompted studies on distribution of jobs to factories [15-21]. Related work modeled the separate entity as *homogenous* flexible job shop [22-24]. To our knowledge, ours is the first report to formulate a *heterogeneous* flexible job shop in distributed scheduling literature. In healthcare, distributed scheduling has hardly been introduced except for efforts on collaborative operating room scheduling [25].

**Adaptive problem solving**, which learns generalized heuristics over a distribution of problems by exploring space of possible heuristics, evolved in the path of syntactic, generative, and statistical learning approaches [26]. *Syntactic approaches* identify those domain’s syntactic structures that influence heuristics’ efficacy, such as tightness of constraints [27], refinement cost [28], or number of feasible serialization orderings [29]. After finding several efforts had resorted to landscape features as syntactic structures, we elaborated its advance in landscape analysis. *Generative approaches* conjecture heuristics for a single problem based on past problem-solving experiences, examples being PRODIGY [30], case-based reasoning [31], meta-Lamarckian learning [10], and adaptive memetic algorithm [32]. Unlike syntactic and generative approaches, statistical learning approaches [26, 33-36] acknowledge the distribution of problems, and learn policies by imitation (supervised learning) or through experiences (reinforcement learning). Among the challenges that adaptive problem solving encounters is that true distribution of learned and generalized instances cannot be mathematically characterized. We did not address these open questions and, instead, charted an alternative route by proposing a complementary adaptive problem solver embodying syntactic and generative approaches. Our rationale was that via landscape analysis and meta-Lamarckian learning, the most suitable strategy would make the search easier, and consequently can be learned for an instance.

**Landscape analysis**, since it reflects topology of combinatorial search space [37-39], helps identify landscape features that highly influence search performances, that characterize problem hardness under different strategies, and that motivate new strategy construction [40]. Hutter et al. [41] modeled algorithm runtime as a function of problem-specific features, where autocorrelation coefficient quantified TSP’s landscape ruggedness. Watson et al. [42] found that mean distance between random local optima and the nearest optima highly correlates with problem hardness for tabu search on job shops. Streeter and Smith [43] characterized job shop’s landscape as a big valley for low ratio of job to machine, and as many separate big valleys for high ratio of job to machine. Several studies showed how hypothesis on landscapes could motivate design of new algorithms. Merz and Katayama [44] upon seeing that local optima for quadratic programming problems were concentrated in a small fraction of the solution domain, designed large-jump mutation to traverse the basins of attraction. Zhang [45] estimated and utilized backbone structures to guide search for Boolean satisfiability. Barbulescu et al. [46] found plateaus dominated search spaces for satellite control network scheduling, and constructed algorithms making larger perturbations to solutions. However, characterization of landscapes for distributed scheduling is rather unclear to-date; how landscape structure changes under different strategies remains unknown. Once we analyzed landscapes for a family of clinical laboratory scheduling instances, comprehensively learned that domain’s syntactic structures and acknowledged relatedness between landscape structure and search strategies, we could then define a new search strategy.
An explicit mathematical formulation, portraying the complicated relations among task, resource and capacity in a whole clinical laboratory, does not exist. An adaptive problem solver taking advantage of landscape analysis is highly needed. These motivate us to present a sophisticated, automated scheduling tool, thereby achieving the best clinical practice and enhancing healthcare quality in a time-critical situation.

3. Workflows and modeling aspects for scheduling in clinical laboratory

We introduce workflows for processing medical specimens in the clinical laboratory and characterize the scheduling model.

3.1. Workflows for processing medical specimens in clinical laboratory

A clinical laboratory offers a variety of testing services. We chose two exceedingly common ones, blood biochemical and immunologic tests. In Fig. 2, a 4-stage workflow for processing blood specimens, involving testing order recognition, specimen preparation, analysis, and results validation and reporting, is illustrated in a schematic of a real laboratory’s workflow and instrument layout. Upon receiving specimens from outpatient/inpatient drawing centers, lab staffs distribute specimens to one of two lines, Region 1 or Region 2. Automated scanners (Region 1) or staff members (Region 2) recognize test requests by reading tube barcodes, and group them into lots. Next, blood specimens will be prepared by centrifuges to separate serum and plasma, and by decapper or staff to remove lids. Third, tubes grouped in batches are loaded to biochemistry or immunoassay analyzers. Last, testing results from analyzers are validated and reported by physicians. Afterwards, tubes are sealed and stored in refrigerator for later use.

![Fig.2. Schematic representation of workflows for processing medical specimens, and instrument layout in a real clinical laboratory.](image)

We interviewed the staff at Clinical Laboratory of China-Japan Friendship Hospital in Beijing to obtain data needed for scheduling. Lower and upper bounds of processing time, maximum capacity, and number of parallel instruments are listed in Table 1. Maximum capacity refers to the upper bound of the instrument that can simultaneously process tubes.
Table 1 Operations, processing time, capacity and quantity of instruments/physicians in a real-practice clinical laboratory.

| Region | Instrument / Staff | Operation                  | Processing time (second) | Maximum Capacity | Number of parallel instruments |
|--------|-------------------|----------------------------|--------------------------|------------------|-------------------------------|
| 1      | Centrifuge        | Centrifugation             | [480, 600]               | 84               | 2                             |
|        | Decapper          | Decapping                  | 2                        | 1                | 1                             |
|        | Biochemistry Analyzer | Biochemical test           | [480, 600]               | 84               | 2                             |
|        | Immunoassay Analyzer | Immunologic test           | [1080, 1800]             | 84               | 2                             |
|        | Staff             | Result validation & reporting | [4, 6]                | 1                | 1                             |
| 2      | Centrifuge        | Centrifugation             | [300, 360]               | 32               | 4                             |
|        | Physician         | Decapping                  | [4, 6]                   | 1                | 1                             |
|        | Biochemistry Analyzer | Biochemical test           | [300, 720]               | 60               | 2                             |
|        | Immunoassay Analyzer | Immunologic test           | [900, 2700]             | 48               | 2                             |
|        | Staff             | Result validation & reporting | [4, 6]                | 1                | 1                             |

3.2. Modeling aspects for scheduling in clinical laboratory

A variety of aspects can be considered when developing scheduling models. The roadmap proposed by Mendez et al. [47] was used; it considers process topology, equipment issues, and time and demand-related constraints. See Mendez et al. [47] for details. The following modeling aspects were considered.

**Process topology:** Modelled as Distributed scheduling in Heterogeneous and Flexible Job Shop Problem environment, it is considered to be distributed in that there are two separate lines, heterogeneous in that lines can differ in instruments’ quantity, capacity and speed (processing time), and flexible job shop in that each line has parallel instruments at certain stages and each tube has its own pre-determined operational sequence.

**Equipment connectivity and assignment:** Each tube can only be assigned to one line. Once assigned, tubes cannot transfer across lines, which imposes hard constraints on equipment allocation.

**Specimen transportation:** Movement of specimens between instruments is performed by guide way in Region 1 or by human action in Region 2. As transport time between instruments is rather short as compared with specimens’ processing time, transport time is neglected in this study.

**Batch size:** Batch size cannot exceed an instrument’s maximum capacity.

**Processing time:** Processing time is considered to be fixed. Processing time for a tube in a batch equals processing time for the batch.

**Demand patterns:** Requested testing orders in short-term scheduling are assumed to be fixed.

**Changeovers:** Consumables (including reagents and probes) are assumed to be sufficient for short-term scheduling. Thus, changeover time is not considered.

**Resource constraints:** Resource constraints focus on instruments while other constraints (including labor, power, reagents, and other utilities) are not considered.

**Time constraints:** Time constraints such as non-working periods on the weekend or maintenance periods are not considered in short-term scheduling.

**Costs:** In our field investigation, because the mean in-lab to reporting turnaround time (MTAT) is crucial, minimization of MTAT is considered.
The overall situation is characterized as a distributed scheduling problem in which each separate line is a flexible job shop and different. The objective is to minimize mean test turnaround time. The problem is abbreviated as D-HFJSP. According to the $\alpha/\beta/\gamma$ notation for classifying scheduling problems [48], D-HFJSP is denoted as $D_{\alpha}, F_{\beta}/batch/MTAT$. It is NP-hard since it is more complex than the NP-hard flexible job shop [49].

4. Problem formulation

In this section, we establish a mixed integer programming model for D-HFJSP. The indices, parameters and variables are explained in Table 2.

| Table 2 Indices, parameters and variables in mixed integer programming model. |
|-----------------------------|-----------------------------|
| Indices                     | Parameters                  | State Variables            | Decision Variables |
| $i$ Index of specimens      | $J_i$ The $i$-th specimen   | $B_{d}^{i,k}$ The $d$-th batch processed on machine $M_{i,k}$ in job shop line $F_i$ | $X_{i}^{d}$ Binary variable, taking value 1 if specimen $J_i$ is processed in job shop line $F_j$; 0 otherwise |
| $j$ Index of operations     | $o_i$ Number of operations of specimen $J_i$ | $T_{d}^{i,k}$ Processing time of batch $B_{d}^{i,k}$ | $Y_{i,j,k}^{d}$ Binary variable, taking value 1 if the $j$-th operation $O_{i,j}$ of specimen $J_i$ belongs to the $d$-th batch $B_{d}^{i,k}$ processed on the $k$-th machine $M_{i,k}$; 0 otherwise |
| $l$ Index of job shop lines | $O_{i,j}$ The $j$-th operation of specimen $J_i$ | $E_{i,j}$ Available time of operation $O_{i,j}$ of specimen $J_i$ | $Z_{d}^{i,k,r}$ Binary variable, taking value 1 if the $d$-th batch $B_{d}^{i,k}$ is processed in the $r$-th position on machine $M_{i,k}$; 0 otherwise |
| $k$ Index of machines in job shop line | $f$ Number of job shop lines | $C_{r}^{i,k}$ Completion time of the $r$-th batch on machine $M_{i,k}$ | |
| $d$ Index of batches        | $F_l$ The $l$-th job shop line | $TAT_i$ Turnaround time of specimen $J_i$ | |
| $r$ Position of batch in the processing sequence on machine | $m_l$ Number of machines in job shop line $F_l$ | | |

The model is represented as
\[
\min \left\{ MTAT = \frac{1}{n} \sum_{l=1}^{B} TAT_l \right\} \tag{1}
\]

Subject to:

\[
\sum_{i=1}^{f} X_i^l = 1, \forall i \tag{2}
\]

\[
\sum_{k=1}^{m} \sum_{d=1}^{n} \sum_{j=1}^{o} W_{i,j}^{l,k} \cdot Y_{i,j,d}^{l,k} = X_i^l, \forall i, l \tag{3}
\]

\[
\sum_{k=1}^{m} \sum_{d=1}^{n} \left( 1 - W_{i,j}^{l,k} \right) \cdot Y_{i,j,d}^{l,k} = 0, \forall i, j, l \tag{4}
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{o} Y_{i,j,d}^{l,k} \leq c_{l,k}, \forall d, k, l \tag{5}
\]

\[
\sum_{d=1}^{n} Z_d^{l,k,r} \leq 1, \forall k, l, r \tag{6}
\]

\[
\sum_{r=1}^{n} Z_d^{l,k,r} = 1, \forall d, k, l \tag{7}
\]

\[
T_d^{l,k} = \max_{i,j} \left\{ Y_{i,j,d}^{l,k} \cdot P_{i,j}^{l,k} \right\}, \forall i, j, d, k, l \tag{8}
\]

\[
C^{l,k}_1 \geq T_d^{l,k} + \Gamma \cdot \left( Z_d^{l,k,r} - 1 \right), \forall d, k, l \tag{9}
\]

\[
C^{l,k}_r \geq C^{l,k}_{r-1} + T_d^{l,k} + \Gamma \cdot \left( Z_d^{l,k,r} - 1 \right), \forall d, k, l, r = 2, \ldots, n \tag{10}
\]

\[
C^{l,k}_r \geq C^{l,k}_{i,j} + \Gamma \cdot \left( Y_{i,j,d}^{l,k} + Z_d^{l,k,r} - 2 \right), \forall d, i, j = 1, \ldots, o_i, k, l, r \tag{11}
\]

\[
E_{i,l} \geq 0, \forall i \tag{12}
\]

\[
C^{l,k}_r \geq 0, \forall k, l, r \tag{13}
\]

\[
TAT_l \geq T_{d+1}^{l,k} + \Gamma \cdot \left( Y_{i,j,d}^{l,k} + Z_d^{l,k,r} - 2 \right), \forall d, i, l, k, r \tag{14}
\]

\[
X_i^l \in \{0, 1\}, \forall i, l \tag{15}
\]

\[
Y_{i,j,d}^{l,k} \in \{0, 1\}, \forall d, i, j, k, l \tag{16}
\]

\[
Z_d^{l,k,r} \in \{0, 1\}, \forall d, k, l, r \tag{17}
\]

- Objective (1) is to minimize mean test turnaround time for all specimens \( J_i \in \{J_1, \ldots, J_n\} \).

- Constraint (2) guarantees that each specimen \( J_i \) can only be assigned to one job shop line \( F_l \), \( F_l \in \{F_1, \ldots, F_f\} \). Once distributed, specimens are not allowed to transfer across lines.

- Constraints (3) and (4) guarantee that the \( j \)-th operation \( O_{i,j} \), \( O_{i,j} \in \{O_{i,1}, \ldots, O_{i,o_i}\} \) for specimen \( J_i \) can only be allotted to one of the batches on machine \( M_{l,k} \) in the dedicated line, and they guarantee that \( O_{i,j} \) is assigned to the eligible machine \( M_{l,k} \), \( M_{l,k} \in \{M_{l,1}, \ldots, M_{l,m_l}\} \) where \( W_{i,j} = 1 \).

- Constraint (5) ensures that lot size does not exceed the machine’s capacity \( c_{l,k} \).

- Constraints (6) and (7) determine each lot’s position in the machine’s processing sequence by guaranteeing each lot can only be placed in one position, and each position can hold at most one lot.

- Constraint (8) implies that the processing time of the \( d \)-th batch is equal to the longest processing time of the specimen in the batch.
• Constraints (9) and (10) imply that the completion time of the \( r \)-th batch on machine \( M_{l,k} \) should be not less than the completion time of the \( (r-1) \)-th batch on machine \( M_{l,k} \) plus the processing time of the \( r \)-th batch.

• Constraint (11) specifies that batch completion time is not less than the available time of all specimens allotted to the batch plus the processing time of the batch, i.e., the batch’s processing cannot start on the current machine until all specimens of that batch complete their preceding operations and are released.

• Constraint (12) implies that no specimen becomes available for processing on the machine of the next processing stage until the batch to which it belongs has been processed.

• Constraint (13) implies that at time 0 all specimens are available for processing.

• Constraint (14) sets batch completion times to be non-negative.

• Constraint (15) defines the turnaround time for specimen \( J_i \).

To conclude, we derived an explicit mixed integer programming model to portray the complicated relationship in clinical laboratory. By solving it, optimal decisions are obtained that boil down to four consecutive decisions: distribution of tubes to lines, assignment of tubes to instruments, sequencing, and time of tubes on instrument. The next section illustrates the solver’s main features.

5. Scatter search based memetic algorithm

Our complementary adaptive problem solver is within a framework of memetic algorithm that is a collection of computationally found strategies based on scatter search, called SSMA. In this section, we present SSMA in detail.

5.1. Scatter search

Scatter search was introduced to solve integer programming [50] and was adopted as an effective algorithm for scheduling [51], including for distributed scheduling [52, 53]. Glover [50] identified a template for scatter search, as shown in Fig. 3.

```plaintext
BEGIN
Initialize:
   Use Diversification Generation Method to generate a population of trial solutions
   Apply Improvement Method to enhance trial solutions
   Use Reference Set Update Method to build and maintain a reference set with best solutions
While (Stopping conditions are not satisfied)
   Generate new subsets with Subset Generation Method by operating on the reference set
   Apply Solution Combination Method to a selected subset to generate new trial solutions
   Apply Improvement Method to improve trial solutions
   Apply Reference Set Update Method to update the reference set
End While
   Output reference set
END
```

Fig. 3. Basic procedure for scatter search.

Though scatter search could generate solutions with certain quality, both “no free lunch theorems” [54] and “conservation of information in search” [55] posit the importance of incorporating domain
knowledge to improve convergence. Consequently, a collection of strategies were sophisticatedly
designed for scatter search. Next, we detail implementations.

5.2. Solution representation and evaluation

We adopt a well-known permutation-based encoding scheme [56] to represent solution in the form of
vector of specimen sequence (VSS). The element’s position represents processing precedence among
all elements. To generate feasible schedules, a novel dispatching heuristic named First Available
Batch Machine (FABM) is proposed to (1) distribute specimens to lines, (2) group specimens into
batches, and (3) assign batches to eligible machines. FABM is illustrated in Fig. 4.

BEGIN
Input a vector of specimen sequence (VSS).
(1): Distribute specimens to lines. In order left to right, select one element at a time from VSS, and
implement Step (1).
(1.1): For selected specimen, determine a set of eligible machines that can process its first
operation.
(1.2): Determine a set of available machines from the eligible set in Step (1.1).
(1.2.1): If the set of available machines is not empty, go to Step (1.3).
(1.2.2): If the set of available machines is empty, suggesting all eligible machines are
occupied, then according to constraints (9) – (11), an eligible machine that has an earliest
available time can be determined. Put the specimen on that machine’s waiting list.
(1.3): Determine the machine from the available set obtained in Step (1.2).
(1.3.1): If there is only one machine in an available set, assign the specimen to this machine.
(1.3.2): If there are multiple machines, assign the specimen to the first machine whose
capacity is closest to its maximum capacity after adding the specimen into its processing
batch.
(1.3.3): If there are multiple machines with the same remaining capacity, randomly assign
specimen to any machine.
(1.4): Specimen is distributed to the job shop line that contains the machine.
Remark: Hereafter, the specimen’s remaining operations can only be processed on machines in the
same job shop line, and it is not allowed to transfer across lines. This complies with Constraint (2).
(2): Group specimens for batch processing on machines.
(2.1): For each machine, specimens assigned to this machine are sorted in the ascending of arriving
time that is calculated from Constraint (12). If more than one specimen has the same arriving
time, then they are ranked according to the precedence specified in VSS.
(2.2): Split ranked specimens into batch(es) by size equal to machine capacity. If the number of
remaining specimens is less than the machine capacity, these specimens are regarded as a
single batch.
(2.3): For each batch assigned to the machine, the machine can only begin processing once this
batch’s specimens are all available. Their arriving times are computed by Constraint (11).
Remark: FABM is proposed for batch processing machines (e.g., centrifuge, biochemistry analyzer
and immunoassay analyzer), but is also applicable for non-batch machines simply by limiting
machine capacity to one (e.g., decapper).
(3): Dispatch the successive operation for each specimen on machine according to Steps (1.1)–(1.3),
and group specimens for batch processing on that machine according to Step (2).
(4): Repeat Step (3) until all operations of all specimens are assigned.
(5): Output the sequencing and timing of operations to machines, that is, the schedule.
END

Fig. 4. First Available Batch Machine (FABM) generates feasible schedules.

We provide an instance to explain FABM. In the illustrative case, 6 specimens are processed by
two job shop lines. Specimens 1-3 are for biochemical tests and specimens 4-6 for immunologic tests.
Processing time for each operation by eligible machines is listed in Appendix A. Time is sampled
within the lower and upper bounds of processing time in Table 1. The sign ‘-’ means the machine is ineligible for the operation. For instance, a value of 545 at the intersection of row $O_{1,3}$ and column \((M_{1,4}, M_{1,5})\) means that the 4th and 5th machines are both eligible for the 3rd operation of specimen $J_3$, and the processing time is 545 seconds. For illustrative purpose, we set maximum capacity to be \([2, 1, 2, 2, 1]\) for centrifuges, decappers, biochemistry analyzers, immunoassay analyzers, and results validation & reporting, respectively.

Assuming VSS to be \([3, 1, 6, 4, 5, 2]\), for the first specimen $J_3$, its first operation $O_{3,1}$ could be randomly assigned to any machine in eligible machine set \(\{M_{1,1}, M_{1,2}, M_{2,1}, M_{2,2}\}\) according to Steps (1.1)-(1.3) in FABM. More specifically, according to Step (1.3.3), $O_{3,1}$ can be assigned to any machine because the four machines’ remaining capacities are the same. Here, $O_{3,1}$ is assigned to $M_{2,1}$. For the second specimen $J_1$, its first operation $O_{1,1}$ is assigned to $M_{2,1}$ according to Steps (1.1), (1.2.1) and (1.3.2), so that $M_{2,1}$ can reach its maximum capacity and begin processing \(\{O_{1,1}, O_{3,1}\}\) as soon as possible. Till now, specimens $J_3$ and $J_1$ have been assigned to a machine in line $F_2$, and their remaining operations can only be processed on machines in line $F_2$ according to Step (1.4). For the remaining specimens in VSS, by implementing Step (1) on an element-by-element basis, $O_{6,1}$ and $O_{4,1}$ are assigned to $M_{1,2}$, and $O_{3,1}$ and $O_{2,1}$ are to $M_{2,2}$. To sum up, specimens \(\{J_6, J_4\}\) and \(\{J_3, J_1, J_4, J_3\}\) are distributed to $F_1$ and $F_2$, respectively.

According to Step (2) in FABM, $J_3$ and $J_1$ are grouped as the first batch on machine $M_{2,1}$ for batching processing, and the batch’s processing time is the maximum processing time of $O_{3,3}$ and $O_{1,1}$ on $M_{2,1}$, i.e., 332 seconds. Following Step (2), $J_6$ and $J_4$ are grouped as the first batch on $M_{1,2}$, and batch processing time is 564 seconds. $J_5$ and $J_2$ are grouped as the 1st batch on $M_{2,2}$, and the processing time is 356 seconds.

The successive operation for each specimen is dispatched to machines by Step (3). For the second operation of each specimen, only one eligible machine in each job shop line, i.e., $M_{1,3}$ and $M_{2,3}$, is available. Once $J_3$ and $J_1$ are released from batching machine $M_{2,1}$, they arrive at the successive eligible machine $M_{2,3}$ simultaneously. Since $M_{2,3}$’s maximum capacity is 1, $O_{3,2}$ is first processed according to Steps (2.1)-(2.2). $J_1$ is on the waiting list of $M_{2,3}$, and $O_{1,2}$ can be processed after $O_{3,2}$ is finished. By the same means, \(\{O_{3,2}, O_{2,2}\}\) and \(\{O_{6,2}, O_{4,2}\}\) are dispatched and processed by machine $M_{2,3}$ and $M_{1,3}$, respectively. We repeat Step (3) until all operations for all specimens are assigned. By implementing FABM, we obtain the schedule, i.e., the sequencing and timing of the machines’ operations. Fig. 5 illustrates the resultant Gantt chart of the schedule. Decision variables \((X^i_t, Y^j_{i,f,a}, Z^k_{j,t,r})\) corresponding to this schedule are valued in Appendix B.

![Fig. 5. Gantt chart depicting the assignment of operations to machines, sequencing of operations, and timing of operations.](image-url)
5.3. Diversification generation method: block-based NEH heuristic

Diversification generation method generates an initial population of trial solutions. We develop a block-based Nawaz-Enscore-Ham (NEH) heuristic. Block-based NEH is detailed in Fig. 6. By implementing it multiple times, we obtain multiple diversified high-quality solutions that will be fed to SSMA as initial solutions.

BEGIN
(1): Randomly generate a vector of specimen sequence (VSS).
(2): Split the VSS into \( b \) blocks by the block size. The blocks are numbered in order left to right, and the serial number of each block remains unchanged.
Remark: (a) The block size equals the greatest common divisor of capacities of the batch machines (see Batch size definition in Section 3.2); (b) If the remainder, that is, the length of VSS modulo the block size, is nonzero, then the remainder is also a block.
(3): Select the first two blocks from VSS, and evaluate two partial possible schedules. Select the better sequence as current sequence.
Remark: FABM evaluates partial possible schedules.
(4): Take block \( \zeta \), \( \zeta = 3, \ldots, b-1 \), and find the best partial schedule(s) by placing it in all possible \( \zeta \) positions in the sequence of blocks that are already scheduled. Set the best one as current sequence.
Step (4.1): If there is more than one best partial schedule, that is, when two possible positions give the same objectives, then the block is inserted in the position which has less total idle time for all machines. The total idle time for all machines is defined as
\[
\sum_{d=1}^{f} \sum_{k=l}^{m_{i}} I_{T_{i,k}} = \sum_{d=1}^{f} \sum_{k=l}^{m_{i}} \sum_{l=1}^{n} \sum_{r=1}^{n} \left( C_{r}^{l,k} - C_{r-1}^{l,k} - Z_{d}^{l,k} - T_{d}^{l,k} \right)
\]  
(19)
(5): Repeat Step (4) until the \((b-1)\)-th block is assigned.
(6): For the \( b \)-th block, if its size is less than the block size, put it at the end of the partial schedule obtained after Step (5); otherwise, perform Step (4) to find the best schedule.
END

Fig. 6. Block-based NEH heuristic for Diversification Generation.

NEH heuristic uses a priority rule to gradually build a complete solution, by positing that jobs with higher total processing time should be given higher priority [57]. Due to its simplicity and efficacy, NEH has been listed among the best heuristics for flow shops, and is used to construct high-quality solutions fed as initial solutions to memetic algorithms [58].

To make NEH heuristic suitable for our case, and to balance diversification and intensification in search, two enhancements are made to construct the block-based NEH heuristic. First, considering alternation of specimen’s priority within a batch cannot improve the optimization criterion. We split testing sequence into blocks, regard each block as an entity, and perform NEH operation on these entities rather than on single specimens, avoiding useless moves within the entity. Computational cost is reduced from \( O(mn^{2}) \) in NEH to \( O(mb^{2}) \) in block-based NEH, where \( m \), \( n \) and \( b \) denote number of machines, number of specimens and number of blocks, respectively, with \( b < n \). Second, we adopt the idea of Ribas et al. [59] which is: If two possible positions give the same criterion, job is inserted in the position that has less total idle time for all machines. It could save extra cost induced from idle machines and potentially produce solutions with high quality.

5.4. Subset generation

Subset Generation generates a new subset by operating on reference set. Our method generates a two-element subset comprising the best solution and the solution furthest away from the best one. The
inter-solution distance is measured by Job Precedence Rule-based distance, JPR distance [60]. It counts
the number of job pairs with identical elements but different precedence in two solutions. JPR distance
metric is detailed in Appendix C. The guaranteed diversity benefits breeding diversified solutions for
the next Solution Combination procedure.

5.5. Solution combination: min-max construction based on votes

Solution Combination generates a new trial solution by splicing the reference solutions in subset. We
adopt min-max construction based on votes by Glover [61], as shown in Fig. 7.

BEGIN

(1): Initialize vote vector \(v' = [v'_1, v'_2]'\) as \([0, 0]'\) at \(t = 0\), where \(v'_1\) and \(v'_2\) are votes for reference
solutions \(\pi_1\) and \(\pi_2\) for the \(t\)-th position in trial solution (from left to right, \(t = 1, 2, ..., n\)).
Assign \([MTAT(\pi_2), MTAT(\pi_1)]'/(MTAT(\pi_1) + MTAT(\pi_2))\) to influence weight \(w = [w_1, w_2]'\)
where \(MTAT(\pi_1)\) and \(MTAT(\pi_2)\) are objective values for \(\pi_1\) and \(\pi_2\), respectively.

(2): For each reference solution, from left to right, select the first element not included in trial
solution as a candidate.

(3): Fill the first unassigned position left to right in trial solution by the min-max construction based
on votes.

(3.1): If two candidates are identical, put any element into the \(t\)-th position of trial solution,
update vote as \(v' = v'^{-1}\), and go to Step (4).

(3.2): If two candidates are different, select one with respect to minimizing
\[
\text{arg min}_{j=1,2} \| (v'^{-1} + I_j) \cdot \sum_{i=1}^{n} (v'^{-1} + I_j(i)) - w \|_1
\]
where \(I_1 = [1, 0]'\), \(I_2 = [0, 1]'\), and \(\| \cdot \|_1\) is \(l^1\)-norm, meaning the summation of abstract
values.

(3.2.1): If both \(I_j\), \(j \in \{1, 2\}\), lead to the same departure in Eq. (20), then randomly select
a reference solution’s candidate. Update vote as \(v' = v'^{-1} + I_j\) when \(j\)-th reference
solution is selected, and go to Step (4).

(3.2.2): Select the candidate from the \(j\)-th reference solution, leading to the minimum
departure in Eq.(20), and update vote as \(v' = v'^{-1} + I_j\).

(4): Repeat Steps (2) – (3) until all positions in trial solution are filled.

END

Fig. 7. Min-max construction based on votes for Solution Combination.

Its basic idea is to minimize the maximum deviation from the trial solution to the two reference
solutions. Two vectors are defined, influence weight vector and vote vector. Influence weight,
reflecting reference solution’s influence on the trial solution, takes reference solution’s fitness value.
Here, fitness value is the reciprocal of the objective value. The higher the fitness value of the reference
solution, the greater its influence on the trial solution. Vote is defined as accumulated contribution of
elements from a reference solution to the trial solution. The more elements the reference solution
contributes to the trial solution, the greater the vote. Min-max construction based on votes optimally
selects element from reference solutions to construct the trial solution with respect to minimizing the
departure of votes (actual contribution) from weights (expected influence): see optimization problem
(20) in Fig. 7. \(l^1\)-norm is used as the distance for measuring the departure of votes from weights. The
trial solution is constructed in a position-by-position way in which the first unassigned position from
left to right is filled by an element chosen from one of the reference solutions. Votes are updated in the
way that the reference solution whose element has been selected would be accredited with one vote. Repeat the procedure until all positions in trial solution are filled with elements.

We illustrate Solution Combination using a concise example in Fig. 8. Two reference solutions \( \pi_1 \) and \( \pi_2 \) are \([3, 1, 6, 4, 5, 2]\) and \([3, 4, 6, 1, 5, 2]\), and their objective values \( MTAT(\pi_1) \) and \( MTAT(\pi_2) \) are 1569.50 and 1829.17, respectively according to FABM. Influence weight is \( w = [w_1, w_2] = [1829.17, 1569.50]/(1569.50 + 1829.17) = [0.54, 0.46]' \), and vote is \( v^0 = [0, 0]' \).

For each reference solution, at the beginning, the first elements that could fill the first position in trial solution are 3. By Step (3.1), element 3 enters into the first position in trial solution, and vote vector is updated as \( v^1 = [0, 0]' \), as shown in Fig. 8(a). Then, element 1 in \( \pi_1 \) and 4 in \( \pi_2 \) are the first elements not included in trial solution, and are candidates for the second position in trial solution. According to Step (3.2), 1 is selected from \( \pi_1 \), as shown in Fig. 8(b). To be specific, when 1 in \( \pi_1 \) is selected, vote vector is updated as \( v^2 = v^1 + I_1 = [1, 0]' \), and the difference between votes and weights in \( l^1 \)-norm is 0.92; and if 4 in \( \pi_2 \) was selected, the vote vector was updated as \( v^2 = v^1 + I_2 = [0, 1]' \), and the difference was 1.08. Thus, 1 is selected from \( \pi_1 \) to enter into the second position of trial solution. For the third position in trial solution, 6 in \( \pi_1 \) and 4 in \( \pi_2 \) are the first elements not ever having been included in trial solution. A choice of 4 in \( \pi_2 \) leads to a minimum departure from votes to weights, as shown in Fig. 8(c). To be specific, when 6 in \( \pi_1 \) was selected, vote vector was updated as \( v^3 = v^2 + I_1 = [2, 0]' \), and the difference between votes and weights in \( l^1 \)-norm was 0.92; and if 4 in \( \pi_2 \) is selected, vote vector is updated as \( v^3 = v^2 + I_2 = [1, 1]' \), and the difference is 0.08. Repeat the selection until all positions in trial solution are filled with elements. The final combined solution is shown in Fig. 8(d).

5.6. Improvement: meta-Lamarckian learning enhanced simulated annealing

Improvement method performs fine search by exploiting the promising region around the trial solution with a goal of achieving enhanced search quality. We develop a meta-Lamarckian learning enhanced simulated annealing as the Improvement method. Our method features two aspects. First, to avoid
stagnancy in simulated annealing, perturbations to the solutions are made by multiple neighborhoods, 
Insert, Swap, Inverse, and our proposed Block-based Insert. Second, to attenuate the negative impact 
of improper neighborhood, meta-Lamarckian learning [10] is employed to adaptively select the most 
promising neighborhood and then the selected neighborhood perturbs the trial solution. 

5.6.1. Simulated annealing

In combinatorial optimization, simulated annealing usually achieves state-of-the-art performance 
despite its simplicity [62]. The simulated annealing starts from an initial state, and randomly generates 
a new state in the neighborhood of the current state, causing a change $\Delta E$ in objective value. In our 
case, the initial state is the trial solution from Solution Combination, and a new state is typically 
sampled from its neighbors. The new candidate is accepted to replace the current state with a 
probability $P_T = \min\{1, \exp(-\Delta E / T)\}$, where $T$ is the temperature. Since the probability to accept an 
inferior candidate gradually decreases with the temperature drops, simulated annealing has potential to 
escape local optima, and has a speed-up convergence.

Parameters, such as initial temperature, annealing schedule, neighborhood and acceptance 
probability, are related to its search efficacy [63]. We set an initial temperature $T_0$ by trial and error, 
use an exponential annealing schedule $T_k = \lambda \cdot T_{k-1}$ where $\lambda \in (0, 1)$ is the cooling coefficient, and $T_k$ 
and $T_{k-1}$ are the temperatures at generations $k$ and $k-1$, respectively.

5.6.2. Neighborhoods (Strategies)

Strategies in adaptive problem solver are referred to different neighborhoods in combinatorial space. 
For our five neighborhoods, Insert, Swap, Inverse, Block-based Insert and composite neighborhood, 
the first four are shown in Fig. 9 and the composite neighborhood in Section 5.6.3.

![Fig. 9. Illustration of Neighborhoods: (a) Insert, (b) Swap, (c) Inverse, (d, e) Block-based Insert.](image-url)

1. Insert: Choose randomly two distinct elements from vector of specimen sequence which is a $n$ - 
length permutation. Insert the back one before the front. As illustrated in Fig. 9(a), specimens 1 
and 5 are selected, and specimen 5 (the latter) is inserted in front of specimen 1.

2. Swap: Select randomly two distinct elements from the sequence and swap. Specimens 1 and 5 are 
selected and swapped, as shown in Fig. 9(b).

3. Inverse: Invert the subsequence between two different random positions in the sequence. 
Specimens 1 and 5 are selected. The inverse, the partial permutation between them, is shown in 
Fig. 9(e).
4. Block-based Insert: Choose randomly two distinct blocks that are generated in Section 5.3. Insert the back block before the front one without changing specimens’ priority within each block. As depicted in Fig. 9(d), Blocks 2 and 3 are selected, and Block 3 is inserted in front of Block 2 while keeping specimens’ priority within each block unchanged. Another illustration is shown in Fig. 9(e) where Block 3 is in front of Block 2, since the first element of Block 3 is before the first element of Block 2. All elements of Block 2 are inserted before the first element of Block 3.

Our initiative to design Block-based Insert is to impose medium perturbations to generate a smooth landscape for large-size instances (200-500 tasks). We find in Section 7 that for large-size instances, extant strategies at hand generate either larger or smaller perturbations, resulting in rugged landscapes with large plateaus, thereby impeding search. Our hypothesis is that medium perturbations may generate smoother landscapes with smaller plateaus. Compared with Insert, Swap and Inverse, the Block-based Insert has a moderate inter-neighbor distance; see experiments in Section 7.2.1 and theoretical analysis in Appendix F. Under Block-based Insert, landscapes are smoother and have smaller attractor plateaus, suggesting large-size instances are easily optimizable. We will experimentally investigate changes in landscape structure under each strategy that imposed perturbations of different magnitudes, and show how search performances are affected in Section 7.

5.6.3. Meta-Lamarckian learning to manage composite neighborhood

Diverse neighborhoods lead to double-edged sword effects. While perturbation behaviors avoid stagnancy, a dilemma occurs over how to select the most appropriate neighborhood at runtime to eliminate negative effects of improper neighborhood. Besides the syntactic approach, in our complementary adaptive problem solver, we adopt a generative approach and relied on meta-Lamarckian learning [10] to generate adaptable strategies from composite neighborhood runtime by learning historical search traces for a single instance.

Meta-Lamarckian learning, divided into training and non-training stages, has in the training stage four simulated annealing algorithms, each algorithm deploying a neighborhood to perform perturbations with the same size of Metropolis sampling [64]. Reward \( \eta_i \) of the \( i \)-th neighborhood is determined as

\[
\eta_i = \frac{\left| MTAT_{b,i} - MTAT_{a,i} \right|}{\theta}
\]  

(21)

where \( MTAT_{b,i} \) and \( MTAT_{a,i} \) are objective values before and after the search using the \( i \)-th neighborhood, and \( \theta \) is the size of Metropolis sampling. To compromise between solution quality and search efficiency, the size of Metropolis sampling is set as the same as the number of blocks.

Utilizing probability for each neighborhood is calculated as

\[
p_{ut,i} = \frac{\eta_i}{\sum_{j=1}^{K} \eta_j}
\]

(22)

where \( p_{ut,i} \) is the utilizing probability for the \( i \)-th neighborhood, and \( K \) is the total number of neighborhoods.

At the non-training stage of meta-Lamarckian learning, based on utilizing probability a roulette wheel rule [65] decides which neighborhood to be selected. If the \( i \)-th neighborhood is selected, its reward will be updated by \( \eta_i = \eta_i + \Delta \eta_i \), where \( \Delta \eta_i = \left| MTAT_{b,i} - MTAT_{a,i} \right| / \theta \) is the reward from the \( i \)-th neighborhood during the non-training phase. Utilizing probability for each neighborhood is updated again by Eq. (22). Repeat the non-training stage until stopping conditions are satisfied.
5.6.4. Framework for meta-Lamarckian learning enhanced simulated annealing

In simulated annealing, meta-Lamarckian learning adaptively selects neighborhood from the composite neighborhood, as depicted in Fig. 10. Its input is a trial solution. It performs fine search by exploiting the promising region around the trial solution. Finally, an improved trial solution is output to update the Reference Set.

![Fig. 10. Meta-Lamarckian learning enhanced simulated annealing for Improvement.](image)

5.7. Reference set update

Reference set update method builds and maintains a reference set consisting of best solutions found so far. The improved trial solution from Improvement enters into the reference set, and the solution furthest away from the best one in the reference set is removed. Update the best solution once the improved trial solution outperforms the current best one.

5.8. Framework of scatter search based memetic algorithm

We sophisticatedly designed all components prerequisite for instantiation of a scatter search. An overall framework for SSMA is illustrated in Fig. 11. First, to quickly generate feasible schedules, a novel and efficient dispatching heuristic, First Available Batch Machine (FABM), was proposed. Second, to initialize trial solutions with high quality and rich diversity, a new block-based NEH strategy was proposed for Diversification Generation. Third, to guarantee diversity in breeding, Subset
Generation Method adopted the JPR distance metric to find the solution furthest away from the best one. Fourth, the min-max construction based on votes was adopted in Solution Combination to generate a new trial solution that was close to parents and inherited more from the parent with higher fitness value. Fifth, to enrich search behaviors and prevent premature convergence, simulated annealing with single- and composite-neighborhood was proposed as Improvement Method. To avoid negative effects of incorrect neighborhood, meta-Lamarckian learning adaptively chose a suitable neighborhood that won over rivals due to its past successful search experiences.

Five algorithms can be derived by specifying one neighborhood in SSMA framework. SSMA-SWP (-INS, -INV, -INB, or -ML) is a memetic algorithm where Swap (Insert, Inverse, Block-based Insert, or composite neighborhoods under meta-Lamarckian learning) is selected respectively. Next, we present an adaptive problem solver where landscape analysis is resorted to in recommending a suitable strategy for a family of instances.

BEGIN
Initialize
(1) Use Diversification Generation Method (see Section 5.3: Block-based NEH heuristic) to initialize a population of trial solutions.
(2) Apply Improvement Method (see Section 5.6: Meta-Lamarckian learning enhanced simulated annealing) to enhance trial solution.
Remarks: Each of the five different neighborhoods (Swap, Insert, Inverse, Block-based Insert, and the composite neighborhood managed by meta-Lamarckian learning) can be used. Each strategy is referred to one neighborhood.
(3) Use Reference Set Update Method (see Section 5.7) to build and maintain a reference set with best solutions.
While (Stopping conditions are not satisfied)
(4) Generate new subsets with Subset Generation Method (see Section 5.4) by operating on the reference set.
(5) Apply Solution Combination Method (see Section 5.5: Min-max construction based on votes) to a selected subset to generate new trial solutions.
(6) Apply Improvement Method (see Section 5.6) to improve trial solutions.
(7) Apply Reference Set Update Method (see Section 5.7) to update the reference set.
End While
(8) Output reference set.
END

Fig. 11. Framework for SSMA.

Adaptive problem solver: landscape analysis to learn algorithm

The no free lunch theorem sustains that algorithm performance is problem-dependent, and no algorithm performs well across all possible problems [54]. A search for a domain-independent problem solver, to match designated problems with effective strategies, is non-trivial and challenging mainly because landscape topology that reflects instance hardness under a search strategy is unknown [66, 67].

For our complementary adaptive problem solver embodying syntactic and generative approaches, we presented the generative approach in Section 5.6, where meta-Lamarckian learning [10] generated adaptable strategies at runtime for an instance. In this section, we resort to landscape analysis in learning domain’s syntactic characteristics for a family of instances.

6.1. Landscape analyzers to characterize landscape
Characteristics of landscape are critical indicators of search space topology, determining instance hardness under search strategies. We use three analyzers, namely fitness distance correlation, autocorrelation coefficient, and local optima network. Through these analyzers, landscape can be quantitatively characterized in terms of connectivity from local to global optima, ruggedness, and the number and size of plateaus. These landscape characteristics are recognized as the domain’s syntactic characteristics, and we hypothesize they influence search strategies’ efficacy.

6.1.1. Fitness distance correlation

Fitness distance correlation [7] reveals connectivity from local optima to global optima, reflecting search difficulties towards global optima. Large fitness distance correlation indicates strong local-global optima connectivity, implying easier movement from local optima to global optima (or the best-so-far solutions), as estimated by their correlation with respect to their distance, as in

$$FDC = \sum_{i=1}^{m} \frac{(f(x_i) - \bar{f})(d(x_i) - \bar{d})}{m \sigma_f \sigma_d}$$

(23)

where $m$ is the size of set of solutions $x_i$, $f(x_i)$ is the objective value, $d(x_i)$ is the distance from $x_i$ to the global optima (or the best-so-far solution), $\bar{f}$ and $\bar{d}$ are the mean of $f(x_i)$, and $d(x_i)$, $\sigma_f$ and $\sigma_d$ are the standard deviations of $f(x_i)$ and $d(x_i)$.

6.1.2. Autocorrelation coefficient

Autocorrelation coefficient [8] reveals landscape ruggedness, reflecting search difficulties around its neighbors. A zero autocorrelation value sustains a rugged landscape. Conversely, a one autocorrelation value suggests a smooth landscape. It is estimated by the correlation of solutions that are several steps apart along a random walk.

$$AC(s) = \sum_{t=1}^{m-s} \frac{(f(x_t) - \bar{f})(f(x_{t+s}) - \bar{f})}{\sigma_f^2 (m-s)}$$

(24)

where $m$ is random walk length, $x_t$ is the $t$-th solution along the random walk, and $s$ is the number of steps apart from two sequences along the random walk.

6.1.3. Local optima network

Local optima network is modeled with nodes (local optima) connected by directed, weighted edges (transition probability from a node to another) [9]. The network reflects risks of getting stuck in local optima and difficulties in traversing basins of attraction. Via local optima network, Daolio et al. [68] characterized flow shop’s landscape in terms of network size, clustering coefficient, transition probabilities, link heterogeneity, path length, and mixing patterns. These features fairly explained instance hardness under iterated local search and predicted performances.

To our knowledge, for modeling real-practice scheduling problems, local optima network has not yet been used. In view of its successful application in flow shop, we believe it has potential in speculating the landscapes although our clinics-orientated scheduling problems are more complex. Appendix D gives a brief explanation about concepts in local optima network, and Fig. 12 illustrates the concepts.
To visualize local optima network, we develop an algorithm to obtain nodes and edges (see Appendix E: Recording nodes and edges for local optima network), and then adopt a force-directed method [9] to assign nodes to positions in a metric space.

![Local optima network diagram](image)

Fig. 12. An illustrative local optima network on a landscape for a minimization problem, showing the two funnels, one connected to red colored nodes and the other to yellow-colored nodes.

6.2. Matching among strategies, landscapes and performances

Awareness of relatedness between landscape structures and search strategies would benefit the identification of an effective strategy. A schematic of the adaptive problem solver is illustrated in Fig. 13. The adaptive problem solver operated as follows.

First, representative instances are solved by different strategies, and search traces are recorded for each representative instance under each strategy. In our preliminary investigation, the running time increased exponentially with instance size. We partitioned instances into subgroups by size (100, 200, 300, 400, and 500), simply selecting instances from each subgroup as representative. We solved representative instances of each subgroup with five algorithms proposed in Section 5.8.

Second, through three landscape analyzers (see Section 6.1), instances’ landscape characteristics were explicitly calculated, revealing how landscapes change under different strategies (see Section
Through experimental observations, we constructed a chain relatedness starting from search strategy’s exploitation competence over search space (measured by inter-neighbor distance), going through landscape topology, and then to search performance. We determine landscape characteristics that highly influence search performance. We relate characteristics of the best search strategy to characteristics of landscape, and infer the choice of a strategy that made search easier for each subgroup.

Finally, the recommended strategy was tested on large-scale instances (see Section 7.3). Statistical analysis is performed to validate performances of the selected strategy (see Section 7.4).

7. Experiments

7.1. Benchmark

We design a benchmark that contains 20 toy instances (2, 4, 6, and 8 specimens) and 150 realistic instances (100, 200, 300, 400, and 500 specimens). In our field investigation in a clinical laboratory attached to China-Japan Friendship Hospital in Beijing, average number of tubes per shift is around 200, but for deeper analysis, we enlarged tube number to 500.

To generate toy instances, several modifications are made in settings in Table 1. The number of centrifuge machines in Region 2 is changed to 2, and capacities for centrifuge, biochemical analyzer, and immunoassay analyzer are all set to be 2. The ratio of biochemical tests to immunologic tests is one to one. For each test size (2, 4, 6, 8), five instances are generated, generating a total of 20 toy instances. To generate realistic instances, the settings in Table 1 are adopted. Ratios of biochemical tests to immunologic tests are 1:4, 1:1, and 4:1, respectively. For each combination of test size (100, 200, 300, 400, 500) and each test ratio, ten instances are generated, for a total of 150 instances. The term “INSTANCE_BT_IT_Idx” denotes BT specimens for biochemical tests, IT specimens for immunologic tests, and Idx as the index number for each combination of BT and IT. Benchmark is summarized in Table 3. Instances are publicly available at [69]. All experiments were performed on a Intel Xeon Gold 6254 3.10GHz machine with 128 GB memory running Ubuntu 18.04.3 LTS. We experimented with Gurobi v9.5.0 and MATLAB 2020b.

| Instances     | Number of specimens | Ratio of biochemical tests to immunologic tests | Number of instances per combination | Total instances |
|---------------|---------------------|-----------------------------------------------|-------------------------------------|-----------------|
| Toy instance  | [2, 4, 6, 8]        | {1:1}                                         | 5                                  | 20              |
| Medium-size   | {100}               | {1:4, 1:1, 4:1}                               | 10                                 | 30              |
| Large-size    | [200, 300, 400, 500]| {1:4, 1:1, 4:1}                               | 10                                 | 120             |

7.2. Preliminary landscape analysis and tune-ups of adaptive problem solver

In this section, we measure inter-neighbor distances under different neighborhoods, reveal how landscapes change under different strategies, determine landscape characteristics that highly influence search performance, and relate characteristics of the best search strategy to landscape characteristics. We speculate about a chain relatedness starting from search strategy’s competence (measured by inter-neighbor distance), then landscape structure, and ending at search performance. Consequently, strategy choice could be inferred as the one that makes search easier.

7.2.1. Inter-neighbor distance under different neighborhoods
Inter-neighbor distance distinguishes the ability of neighborhood’s exploitation over search space. We theoretically measure inter-neighbor distance under each neighborhood. To save space, proofs are put in Appendix F. Results are given in Table 4.

Table 4 Theoretical expectations and variances of the inter-neighbor distance under different neighborhoods.

| Neighborhood     | Expectation of Inter-neighbor Distance | Variance of Inter-neighbor Distance |
|------------------|----------------------------------------|-------------------------------------|
| Inverse          | \((n - 2)(n - 3)/(6n(n - 1))\)         | \((n + 1)(n - 2)(n - 3)(7n - 18)/(180n^2(n - 1)^2)\) |
| Insert           | \(2(n + 1)/(3n(n - 1))\)               | \(2(n + 1)(n - 2)/(9n^2(n - 1)^2)\) |
| Swap             | \(2(2n - 1)/(3n(n - 1))\)              | \(8(n + 1)(n - 2)/(9n^2(n - 1)^2)\) |
| Block-based Insert | \(2n(b + 1)/(3b^2(n - 1))\)         | \(2(b + 1)(b - 2)n^2/\)\((9b^4(n - 1)^2)\) |

We conduct experiments on representative instances and verify that experimental values are consistent with theoretical values; see Table 5. For medium- and large-size instances, Inverse produces the largest inter-neighbor distance, suggesting a potential largest exploration over the solution space. Swap and Insert have the smallest inter-neighbor distances, favoring smaller exploitation. Block-based Insert is at a moderate perturbation level among the four neighborhoods.

Table 5 Theoretical and experimental inter-neighbor distances under different neighborhoods on representative instances.

| Instance          | Neighborhood     | Theoretical values | Experimental values |
|-------------------|------------------|--------------------|---------------------|
|                   |                  | Mean               | Variance            | Average             | Variance            |
| Medium-size       | Inverse          | 0.1600             | 0.0371              | 0.1678              | 0.0381              |
| (INSTANCE_20_80_5)| Insert           | 0.0068             | 2.24E-05            | 0.0075              | 2.54E-05            |
|                   | Swap             | 0.0134             | 8.98E-05            | 0.0148              | 1.05E-04            |
|                   | Block-based Insert | 0.0280          | 3.47E-04            | 0.0259              | 3.24E-04            |
| Large-size        | Inverse          | 0.1633             | 0.0380              | 0.1714              | 0.0407              |
| (INSTANCE_40_160_5)| Insert           | 0.0034             | 5.58E-06            | 0.0035              | 5.74E-06            |
|                   | Swap             | 0.0067             | 2.23E-05            | 0.0072              | 2.35E-05            |
|                   | Block-based Insert | 0.0137          | 8.79E-05            | 0.0132              | 8.60E-05            |
| Large-size        | Inverse          | 0.1644             | 0.0383              | 0.1645              | 0.0373              |
| (INSTANCE_60_240_5)| Insert           | 0.0022             | 2.48E-06            | 0.0025              | 2.66E-06            |
|                   | Swap             | 0.0045             | 9.91E-06            | 0.0045              | 9.70E-06            |
|                   | Block-based Insert | 0.0090          | 3.92E-05            | 0.0087              | 3.82E-05            |
| Large-size        | Inverse          | 0.1650             | 0.0384              | 0.1635              | 0.0390              |
| (INSTANCE_80_320_5)| Insert           | 0.0017             | 1.39E-06            | 0.0019              | 1.57E-06            |
|                   | Swap             | 0.0033             | 5.57E-06            | 0.0035              | 5.99E-06            |
|                   | Block-based Insert | 0.0068          | 2.21E-05            | 0.0069              | 2.29E-05            |
| Large-size        | Inverse          | 0.1653             | 0.0385              | 0.1436              | 0.0302              |
| (INSTANCE_100_400_5)| Insert           | 0.0013             | 8.91E-07            | 0.0013              | 7.90E-07            |
|                   | Swap             | 0.0027             | 3.56E-06            | 0.0026              | 3.27E-06            |
|                   | Block-based Insert | 0.0054          | 1.42E-05            | 0.0054              | 1.40E-05            |

7.2.2. Fitness distance correlation measuring local-global optima connectivity

We investigate the local-global optima connectivity under each strategy via fitness distance correlation. Two representative instances, INSTANCE_20_80_5 (medium-size) and INSTANCE_60_240_5 (large-size) were selected for illustrative analysis. Number of iterations for SSMAs was set as 20,000. Objective value relative to the best-so-far solution (y-axis) is depicted against its distance from the best-so-far solution (x-axis) for medium-size instance in Fig. 14 and for large-size instance in Fig. 15.

It is observed that distances between local and global optima are often less than 0.6 on medium-size instance and 0.5 on large-size instance, indicating that local optima are clustered in a small area of solution space, and local optima of large-size instance are more concentrated. Second, fitness distance correlation under each strategy is between [0.13, 0.47] on the medium-size instance, while in between [0.39, 0.79] on the large-size instance, indicating a stronger connectivity on large-size instance. In
particular, on medium-size instance, SSMA-INV (-SWP) have weaker connectivity than SSMA-INS (-INB, -ML). On the large-size instance, except for SSMA-INV, all SSMAs have a fitness distance correlation larger than 0.6, featuring a strong connectivity among optima. On both instances SSMA-ML has high fitness distance correlation, suggesting that it can build a strong connectivity from local to global optima over the landscape by selecting the most competitive search strategy at runtime.

Fig. 14. Fitness distance correlation on a medium-size instance (INSTANCE_20_80_5).

Fig. 15. Fitness distance correlation on a large-size instance (INSTANCE_60_240_5).

7.2.3. Autocorrelation coefficient measuring landscape’s ruggedness

We utilize autocorrelation coefficient to reveal landscape’s ruggedness. Two representative instances are solved by five SSMAs. Autocorrelation coefficient under each strategy is calculated according to
Eq. (24). Number of iterations for SSMAs is set as 20,000. Descending curves and autocorrelation coefficient are depicted for medium-size instance in Fig. 16 and for large-size instance in Fig. 17.

Fig. 16. Autocorrelation coefficient to measure ruggedness of landscape on a medium-size instance (INSTANCE_20_80_5).
Top: Descending curves; Bottom: Autocorrelation coefficient along a random walk of 500 steps.

In Fig. 16, SSMA-SWP (-INS, -ML) have larger autocorrelation coefficients than SSMA-INV (-INB), suggesting Swap, Insert, and composite neighborhood lead to smoother landscapes which support a continuously improved objective value, whereas Inverse and Block-based Insert lead to more rugged landscapes. Strategies perform well on a smooth landscape and a fast convergence is expected. This is validated by faster descending curves under SSMA-SWP (-INS, -ML). Besides, as revealed in Table 5, Swap and Insert generate smaller perturbations to solutions than Inverse and Block-based Insert. It can be concluded that medium-size instance is easier under Swap and Insert than Inverse and Block-based Insert since Swap and Insert generated smaller perturbations and led to a relatively smooth landscape.

Fig. 17. Autocorrelation coefficient to measure ruggedness of landscape on a large-size instance (INSTANCE_60_240_5).
Top: Descending curves; Bottom: Autocorrelation coefficient along a random walk of 500 steps.
In Fig. 17, SSMA-INB (-ML, -SWP) have larger autocorrelation coefficients than SSMA-INV (-INS), indicating Swap, Block-based Insert, and composite neighborhoods lead to smoother landscape. SSMA-INB achieved the fastest convergence, followed by SSMA-ML (-SWP). As revealed in Table 5, compared with other strategies Block-based Insert generates moderate perturbations to solutions in terms of mean inter-neighbor distance from which we concluded that large-size instance is easier for Block-based Insert that leads to smooth landscapes. Furthermore, SSMA-ML deploys meta-Lamarckian learning to adaptively select the most-rewarded neighborhood from composite neighborhood at runtime.

7.2.4. Local optima network revealing difficulty in jumping out of local optima

Local optima networks under SSMA-INV (-INS, -SWP, -INB) are visualized in Fig. 18 for toy instance, and in Fig. 19 for medium-size instance. The heat colors palette, a sequential color scheme skewed to reds and yellows, is used. Red identifies the funnel with the best-so-far optima, and a yellow color gradient reflects decrease in solution performance.

Since plateaus harass search, it is visually observed from Fig. 18 and 19 that even toy instance’s local optima networks contain multiple sub-optimal plateaus (attractor plateaus), implying the search complexity for distributed scheduling problems. In Section 7.3.1 we show toy instances’ hardness by comparative investigations between SSMAs and Gurobi Optimizer.

For the toy instance in Fig. 18, SSMA-INS (-SWP) discovers escape edges from plateaus to promising regions more easily than SSMA-INV (-INB) did. It is observed that local optima networks under SSMA-INV (-INB) are structured as multiple funnels. The considerable amount of attractor plateaus are large-sized, exhibiting great gravitational force, which make searches less likely to escape them. Additionally, SSMA-INV (-INB) have more scattered sink plateaus, meaning their searching behaviors at the end of search remain diversified, impeding convergence. It is validated by a relatively poor solution under SSMA-INV (-INB) in Section 7.3.1. It allows the conclusion that toy instances are easier under Swap and Insert (than under Block-based Insert and Inverse) that lead to local optima networks with less funnels and smaller-sized attractor plateaus.

For a medium-size instance in Fig. 19, compared with SSMA-INV (-INB), SSMA-INS (-SWP) have smaller plateaus; see plateau average size in Table 6. Smaller average size of plateaus in a local optima network suggests smaller basins of attraction which impose fewer impediments on jumping out of plateaus and support a better exploration over solution space. SSMA-INS (-SWP) could discover escape edges from attractor plateaus to more promising regions. Search traces under SSMA-SWP are restricted in small search areas as shown in Fig. 19(c), while search traces under SSMA-INV (-INS, -INB) are more disperse. In this regard, SSMA-SWP showed aggressive exploitation leading to the fastest convergence.

For a large-size instance, since under different strategies it becomes hard to visually discern differences among local optima networks, statistics abstracted from these networks are used to speculate landscape structures. Compared with SSMA-INV (-INS), SSMA-INB (-SWP) have smaller plateaus (see plateau average size in Table 6), indicating less attraction from attractor plateaus and facilitating escape from them. This agrees with observations in autocorrelation coefficients in Section 7.2.3, where SSMA-INV (-INS) have smaller autocorrelation coefficients, leading to more rugged landscapes and slower convergence rate. Because, as shown in Table 5, compared with Insert and Inverse, Block-based Insert generates moderate perturbations in search space, it can be concluded that large-size instances are easier under moderate-disturbance neighborhood, e.g., Block-based Insert.
Fig. 18. Local optima networks under SSMAIs on toy instance (INSTANCE_4_4_5).
Fig. 19. Local optima networks under SSMAs on medium-size instance (INSTANCE_20_80_5).
7.2.5. Strategy inferred from landscape analysis

Preliminary experiments on representative instances reveal several interesting findings, as summarized in Table 6.

Table 6 Relationship between inter-neighbor distances, landscapes, and performances. * We do not provide mean inter-neighbor distance and local optima network’s statistics for SSMA-ML, since a composite neighborhood is used.

| Instance          | Algorithm | Neighborhood   | Mean Inter-neighbor Distance | Landscape Analyzers | Performance Rank |
|-------------------|-----------|----------------|-----------------------------|---------------------|------------------|
|                   |           |                | Fitness Distance Correlation | Autocorrelation, AC(1) | Plateau average size (local optima network) |
| INSTANCE_20_80_5  | SSMA-INS  | Insert         | 0.0068                      | 0.4708              | 0.7473           | 27.60            | 3               |
| (100 tasks)       | SSMA-SWP  | Swap           | 0.0134                      | 0.1335              | 0.8548           | 17.34            | 1               |
|                   | SSMA-INV  | Inverse        | 0.1600                      | 0.1521              | 0.1255           | 32.24            | 5               |
|                   | SSMA-INB  | Block-based Insert | 0.0280                  | 0.4625              | 0.1462           | 47.40            | 4               |
|                   | SSMA-ML   | Composite      | -                           | 0.4073              | 0.7297           | -                | 2               |
| INSTANCE_40_160_5 | SSMA-INS  | Insert         | 0.0034                      | 0.8286              | 0.5142           | 23.34            | 5               |
| (200 tasks)       | SSMA-SWP  | Swap           | 0.0067                      | 0.7985              | 0.5973           | 16.81            | 3               |
|                   | SSMA-INV  | Inverse        | 0.1633                      | 0.1813              | 0.1112           | 26.91            | 4               |
|                   | SSMA-INB  | Block-based Insert | 0.0137                  | 0.7639              | 0.5411           | 23.04            | 2               |
|                   | SSMA-ML   | Composite      | -                           | 0.8721              | 0.5158           | -                | 1               |
| INSTANCE_60_240_5 | SSMA-INS  | Insert         | 0.0022                      | 0.7899              | 0.4602           | 22.67            | 5               |
| (300 tasks)       | SSMA-SWP  | Swap           | 0.0045                      | 0.6667              | 0.8753           | 18.40            | 3               |
|                   | SSMA-INV  | Inverse        | 0.1644                      | 0.3886              | 0.0487           | 19.22            | 4               |
|                   | SSMA-INB  | Block-based Insert | 0.0090                  | 0.6203              | 0.7101           | 16.91            | 1               |
|                   | SSMA-ML   | Composite      | -                           | 0.7334              | 0.6567           | -                | 2               |
| INSTANCE_80_320_5 | SSMA-INS  | Insert         | 0.0017                      | 0.7222              | 0.2809           | 13.79            | 5               |
| (400 tasks)       | SSMA-SWP  | Swap           | 0.0033                      | 0.8023              | 0.8886           | 10.89            | 2               |
|                   | SSMA-INV  | Inverse        | 0.1650                      | 0.5334              | 0.0157           | 13.53            | 3               |
|                   | SSMA-INB  | Block-based Insert | 0.0068                  | 0.5190              | 0.7215           | 17.32            | 1               |
|                   | SSMA-ML   | Composite      | -                           | 0.8168              | 0.8764           | -                | 4               |
| INSTANCE_100_400_5| SSMA-INS  | Insert         | 0.0013                      | 0.4095              | 0.9171           | 18.18            | 5               |
| (500 tasks)       | SSMA-SWP  | Swap           | 0.0027                      | 0.8170              | 0.9775           | 12.65            | 3               |
|                   | SSMA-INV  | Inverse        | 0.1653                      | 0.3640              | 0.0234           | 15.35            | 4               |
|                   | SSMA-INB  | Block-based Insert | 0.0054                  | 0.5009              | 0.8599           | 13.74            | 2               |
|                   | SSMA-ML   | Composite      | -                           | 0.5263              | 0.4488           | -                | 1               |

For medium-size instances, SSMA-SWP showed the best performance. It has the lowest fitness distance correlation, suggesting that fine-grained search is more conducive to enhancing search. Swap neighborhood produces smaller disturbances, favoring smaller exploitation over the solution space. SSMA-SWP has the highest autocorrelation coefficient (smoothest landscapes) and the smallest-sized attractor plateaus (fewest impediments on jumping out of plateaus), supporting a fast descending speed. In contrast, SSMA-INV performed worst. It also has very low fitness distance correlation, but lacks the ability to perform fine searches due to its largest disturbances to solutions. SSMA-INV was faced with the most rugged landscapes (smallest autocorrelation coefficient), including larger attractors (larger plateau) that made the search stagnate.

For large-size instances, SSMA-INB achieved the best performance. It has a moderate fitness distance correlation, sustaining that the search in between fine-grained and coarse-grained searches would be preferable. Block-based Insert could generate moderate disturbances, meeting the requirement. Under the moderate disturbances, it generates smoother landscapes with smaller plateaus, thereby fostering search. In contrast, SSMA-INS performed worst. It has a very high fitness distance correlation, supporting that large-disturbance exploration is an obvious need. Nevertheless, SSMA-INS has the smallest disturbance, which can not fulfill the need. SSMA-INS can hardly traverse the search space manifested in rugged landscapes with large-sized plateau due to lack of larger momentum. Next, we will test the adaptive problem solver’s performance on large-scale instances.
7.3. Comparisons

7.3.1. Toy instances

We investigated the comparative efficiency between our proposed algorithms and Gurobi Optimizer [70] on small-size toy instances. Gurobi solver is called by YALMIP in MATLAB [71]. CPU time for Gurobi runs is set to 3000 seconds. SSMAs are terminated either if they reach the maximum iterations 10,000, or if the best-so-far solution has been unimproved after 1,000 iterations. Each variant of SSMAs is independently run 30 times on each instance. Best objective values and average computing time are listed in Tables 7 and 8, respectively. Gurobi only obtains optima on INSTANCE_1_1 but fails to find optima on the rest of the instances in 3000 seconds. For instances with 8 specimens, SSMAs outperform Gurobi. We noticed in the distributed job shop literature that Naderi and Azab [72] optimally solved small sized instances up to 12 jobs using CPLEX, another state-of-the-art MIP solver. The distributed scheduling with heterogeneous and flexible job shops in our study is more difficult to solve than the distributed job shop. Next, we will test the adaptive problem solver’s performance on medium- and large-size instances.

| Instance | Gurobi Optimizer | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|------------------|----------|----------|----------|----------|----------|
| 1_1_{[1-5]} | 1363.70 | 1460.50 | 1460.50 | 1460.50 | 1460.50 | 1460.50 |
| 2_2_{[1-5]} | 1380.06 | 1440.92 | 1440.92 | 1440.92 | 1472.20 | 1440.92 |
| 3_3_{[1-5]} | 1400.76 | 1542.98 | 1542.98 | 1542.98 | 1543.12 | 1542.98 |
| 4_4_{[1-5]} | 1403.98 | 1399.08 | 1398.96 | 1398.96 | 1403.18 | 1398.96 |

7.3.2. Medium- and large-size instances

We examine SSMAs’ performances on 30 medium-size instances and 120 large-size instances. SSMAs are terminated if they reach the maximum iterations 10,000. Each algorithm is independently run 30 times on each instance. The four performance metrics considered for comparison are Best Relative Error to $C^*$ (BRE), Average Relative Error to $C^*$ (ARE), Worst Relative Error to $C^*$ (WRE), and CPU time, where $C^*$ is the best-so-far value. Results are summarized in Appendixes G and H.

| Instance | Gurobi Optimizer | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|------------------|----------|----------|----------|----------|----------|
| 1_1_{[1-5]} | 1.05 | 0.76 | 0.75 | 0.74 | 0.75 | 0.77 |
| 2_2_{[1-5]} | 3008.14 | 0.80 | 0.80 | 0.80 | 0.80 | 0.84 |
| 3_3_{[1-5]} | 3021.38 | 0.92 | 0.94 | 0.94 | 0.94 | 0.96 |
| 4_4_{[1-5]} | 3005.16 | 1.08 | 1.15 | 1.14 | 1.10 | 1.17 |

In Section 7.2, it is learned that SSMA-SWP is the most effective algorithm for medium-size instances, and SSMA-INB is the most effective algorithm for large-size instances. Results in Appendixes G and H agree with speculations. For medium-size instances, SSMA-SWP achieved the best BRE in nearly 57% instances and the best ARE in nearly 73% instances. For large-size instances, SSMA-INB achieved the best BRE in nearly 54% instances and the best ARE in nearly 95% instances. SSMA-ML that managed composite neighborhoods by meta-Lamarckian learning achieved competitive performances. For medium-size instances, SSMA-ML had the best BRE in nearly 43% instances and the best ARE in 7% instances. For large-size instances, SSMA-ML had the best BRE in nearly 36% instances and the best ARE in 2% instances. Meta-Lamarckian learning’s impacts on performances for INSTANCE_60_240_5 are visually investigated in Fig. 20. After the training stage, Block-based Insert is rewarded by the highest utilization probability, yet it is not always chosen along the search. Convergence is faster as Block-based Insert is utilized, while convergence is slower, even kept stagnation as Block-based Insert is not applied.
7.4. Statistical comparisons

7.4.1. Pairwise comparisons

We conduct two pairwise statistical tests, a Sign test and Wilcoxon test [73], to show statistical significations of comparative performances between any pairs of algorithms. The criterion used is the average performance of multiple independent runs for each algorithm on each instance.

We perform Sign test to statistically demonstrate performance difference between SSMA-ML and other algorithms, as summarized in Table 9. Wins (Loses or Ties) count the times when SSMA-ML performs superior (inferior or identically) to its counterpart algorithm. The detected difference indicates that SSMA-ML is outperformed by SSMA-SWP on medium-size instances and by SSMA-INB on large-size instances.

A Wilcoxon test is utilized to illustrate the degree of difference among algorithms. The difference between two algorithms will be ranked according to absolute value among all test instances. The sum of ranks that one algorithm is superior (inferior) to another is indicated by $R^+$ ($R^-$). As observed in Table 10, for medium-size instances, $p$ values that SSMA-SWP performs better than SSMA-INV (-INS, -INB, and -ML) are at significance level of 0.001. For large-size instances, $p$ values that SSMA-INB performs better than others are at significance level of 0.001. Pairwise statistical comparisons agree with speculations about each algorithm’s performance.
Table 9 Results of Sign Test.

| Instances     | SSMA-ML | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB |
|---------------|---------|----------|----------|----------|----------|
| 100 specimens | Wins(+) | 30       | 21       | 7        | 25       |
|               | Loses(-)| 0        | 9        | 23       | 5        |
|               | Ties(=) | 0        | 0        | 0        | 0        |
|               | Detected differences | 0.000 | 0.045 | 0.006 | 0.001 |
| 200 specimens | Wins(+) | 30       | 29       | 25       | 0        |
|               | Loses(-)| 0        | 0        | 5        | 30       |
|               | Ties(=) | 0        | 1        | 0        | 0        |
|               | Detected differences | 0.000 | 0.000 | 0.001 | 0.000 |
| 300 specimens | Wins(+) | 30       | 30       | 30       | 0        |
|               | Loses(-)| 0        | 0        | 0        | 30       |
|               | Ties(=) | 0        | 0        | 0        | 0        |
|               | Detected differences | 0.000 | 0.000 | 0.000 | 0.000 |
| 400 specimens | Wins(+) | 30       | 30       | 30       | 0        |
|               | Loses(-)| 0        | 0        | 0        | 30       |
|               | Ties(=) | 0        | 0        | 0        | 0        |
|               | Detected differences | 0.000 | 0.000 | 0.000 | 0.000 |
| 500 specimens | Wins(+) | 27       | 22       | 24       | 3        |
|               | Loses(-)| 3        | 8        | 5        | 27       |
|               | Ties(=) | 0        | 0        | 1        | 0        |
|               | Detected differences | 0.000 | 0.018 | 0.001 | 0.000 |

Table 10 Results of Wilcoxon’s Test.

| Instances size | Comparison             | R⁺  | R⁻  | p value |
|----------------|------------------------|-----|-----|---------|
|                | SSMA-SWP versus SSMA-INV | 464.0 | 1.0 | 0.0000 |
|                | SSMA-SWP versus SSMA-INS | 458.0 | 7.0 | 0.0000 |
|                | SSMA-SWP versus SSMA-INB | 417.0 | 48.0 | 0.0000 |
|                | SSMA-SWP versus SSMA-ML | 401.5 | 63.5 | 0.0000 |
| 200 specimens  | SSMA-INB versus SSMA-INV | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-INS | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-SWP | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-ML | 465.0 | 0.0  | 0.0000 |
| 300 specimens  | SSMA-INB versus SSMA-INV | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-INS | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-SWP | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-ML | 465.0 | 0.0  | 0.0000 |
| 400 specimens  | SSMA-INB versus SSMA-INV | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-INS | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-SWP | 465.0 | 0.0  | 0.0000 |
|                | SSMA-INB versus SSMA-ML | 465.0 | 0.0  | 0.0000 |
| 500 specimens  | SSMA-INB versus SSMA-INV | 454.0 | 11.0 | 0.0000 |
|                | SSMA-INB versus SSMA-INS | 441.0 | 24.0 | 0.0000 |
|                | SSMA-INB versus SSMA-SWP | 453.0 | 12.0 | 0.0000 |
|                | SSMA-INB versus SSMA-ML | 442.5 | 22.5 | 0.0000 |

7.4.2. Multiple comparisons

We perform Friedman’s test to identify the existence of differences in average performance among a group of algorithms. The null hypothesis $H_0$ is that there is no significant difference among algorithms, while the alternative hypothesis $H_1$ indicates the presence of significant differences. Table 11 shows results of Friedman’s test, including average rank for each algorithm and $p$ value. The null hypothesis was rejected under the significance level of 0.001, suggesting that significant differences among algorithms exist. On medium-size instances SSMA-SWP performed best followed by SSMA-ML, while on large-size instances SSMA-INB performed best, also followed by SSMA-ML.
Table 11 Results of Friedman’s Test.

| Instance size | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML | p value |
|---------------|----------|----------|----------|----------|---------|---------|
| 100 specimens | 4.63     | 3.08     | 1.53     | 3.52     | 2.23    | 0.000   |
| 200 specimens | 4.23     | 4.53     | 3.05     | 1.00     | 2.18    | 0.000   |
| 300 specimens | 3.70     | 4.58     | 3.72     | 1.00     | 2.00    | 0.000   |
| 400 specimens | 3.73     | 4.80     | 3.47     | 1.00     | 2.00    | 0.000   |
| 500 specimens | 3.67     | 3.70     | 3.88     | 1.30     | 2.45    | 0.000   |

8. Conclusions and future work

Application of artificial intelligence in clinical medicine still faces daunting challenges. Our research is in the direction of narrowing the gap between artificial intelligence research and clinics-orientated operations practice. This paper formulated a mixed integer programming model for scheduling clinical laboratory testing, presented an adaptive problem solver by leveraging instances’ landscape features, and validated its efficacy in large-scale instances. We made several assumptions, and limitations exist, both of which should be noted.

Scheduling model for real-practice clinical laboratory: We formulated a mixed integer programming model for the heterogeneous and flexible job shop in distributed scheduling (D-HFJSP). This model focused on efficiency criterion in short-term scheduling, specifically deducing turnaround time. Other issues related to efficiency worth considering are tests’ priority, instrument recalibration, and staff changeover. In addition to efficiency, clinical laboratory should be robust enough to uncertainties in long-term scheduling, shortage of consumables (reagents and probes), breakdowns and other. Furthermore, a mathematical model is fragile when the situation changes, and incorporation of staff’s situational awareness helps to achieve expected-quality schedules. Initiatives related to improved patient care, such as specimen dividing for reduced blood draws, could be considered.

Adaptive problem solving: We presented an adaptive problem solver embodying syntactic (landscape characteristics) and generative (meta-Lamarckian learning) approaches to specify which search strategy to use. It reduced repetitive algorithm-tuning work as instances frequently change over time. Since our initiative was to provide an easy-to-use solver to the physician, it was essentially a preliminary lookup table. It is worth further bringing in machine learning by acknowledging problems’ distribution. It is, however, interesting and challenging to explicitly characterize a distribution.

Landscape analysis: It revealed landscape structures, guided strategy selection, and motivated the design of a new strategy. In this study, relatedness between characteristics of landscape and algorithm performance was established based on experimental observations but not on statistical analysis. It is worthwhile to further deploy regression model to validate the relationship by enrolling enough observations.

Local optima network: It visualized landscapes in nodes and edges. We used it as an offline approach to successfully reflect instance hardness and predict search performance. Since each algorithm is independently run multiple times to reveal the complex network, it is too expensive to guide search online. Realizing rapid prototyping of the local optima network could be a possible research direction.

Meta-Lamarckian learning: It adaptively selected the most promising strategy by rewards, and achieved beyond average performance. Since at the training stage every strategy needs to run for the same amount of perturbations, consuming a lot of computational budgets, it is interesting to see an
economic training strategy to save overhead, especially on expensive-to-evaluation instances. At the training stage, each strategy started the search from the same startpoint at which instance hardness might be different for each strategy, leading to “unfair” rewards. A reward strategy based on both multiple starting points and voting is expected to provide a robust measure.

Search strategy with adjustable disturbance amplitude: Changes in landscape were investigated under strategies that imposed different-magnitude perturbations. The relatedness observed motivated the design of a new search strategy with moderate perturbations. Search strategy with adjustable disturbance amplitude is highly needed to perform an expected exploitation over the search space.

Clinical practice will remain dependent on laboratory staff’s experience to draw up schedules even though the advance in applied modeling and an adaptive problem solver is vital towards sophisticated, automated scheduling for clinical decision-making. It helps to achieve a better clinical practice, enhance the quality of healthcare, and shape the laboratory of the future.

Acknowledgements

This research greatly benefited from multiple field investigations hosted by Associate Professor Mei Wang (Clinical Laboratory at China-Japan Friendship Hospital in Beijing) whose questions and comments were incorporated in the final version. We gratefully acknowledge her trust and insight. Special thanks go to Professor Ann Marie Ross (University of Chinese Academy of Sciences) for her hard work on polishing the manuscript. The author Bo Liu received financial support from Frontier Science Key Research Program, Chinese Academy of Sciences (QYZDB-SSW-SYS020).
### Appendix A: Processing time for operations on eligible machines

**Table A.1**

Processing time for operations on eligible machines.

|   | $M_{1,1}$ | $M_{1,2}$ | $M_{1,3}$ | $M_{1,4}$ | $M_{1,5}$ | $M_{1,6}$ | $M_{1,7}$ | $M_{1,8}$ | $M_{2,1}$ | $M_{2,2}$ | $M_{2,3}$ | $M_{2,4}$ | $M_{2,5}$ | $M_{2,6}$ | $M_{2,7}$ | $M_{2,8}$ |
|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| $O_{1,1}$ | 545       | -         | -         | -         | -         | -         | -         | -         | 332       | -         | -         | -         | -         | -         | -         | -         |
| $O_{1,2}$ | -         | 2         | -         | -         | -         | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         |
| $O_{1,3}$ | -         | -         | 545       | -         | -         | -         | -         | -         | 530       | -         | -         | -         | -         | -         | -         | -         |
| $O_{1,4}$ | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         | -         | -         | -         | 5         | -         |
| $O_{2,1}$ | 593       | -         | -         | -         | -         | -         | -         | -         | 356       | -         | -         | -         | -         | -         | -         | -         |
| $O_{2,2}$ | -         | 2         | -         | -         | -         | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         |
| $O_{2,3}$ | -         | -         | 593       | -         | -         | -         | -         | -         | 695       | -         | -         | -         | -         | -         | -         | -         |
| $O_{2,4}$ | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         | -         | -         | -         | -         | 5         |
| $O_{3,1}$ | 530       | -         | -         | -         | -         | -         | 325       | -         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{3,2}$ | -         | 2         | -         | -         | -         | -         | -         | 4         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{3,3}$ | -         | -         | 530       | -         | -         | -         | -         | 475       | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{3,4}$ | -         | -         | -         | 4         | -         | -         | -         | -         | -         | -         | -         | -         | -         | -         | -         | 4         |
| $O_{4,1}$ | 597       | -         | -         | -         | -         | -         | 358       | -         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{4,2}$ | -         | 2         | -         | -         | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{4,3}$ | -         | -         | -         | 1787      | -         | -         | -         | -         | -         | -         | 2669      | -         | -         | -         | -         | -         |
| $O_{4,4}$ | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         | 5         | -         | -         | -         | -         |
| $O_{5,1}$ | 516       | -         | -         | -         | -         | -         | 318       | -         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{5,2}$ | -         | 2         | -         | -         | -         | -         | -         | 4         | -         | -         | -         | 1442      | -         | -         | -         | -         |
| $O_{5,3}$ | -         | -         | -         | 1297      | -         | -         | -         | -         | -         | -         | -         | -         | 4         | -         | -         | -         |
| $O_{5,4}$ | -         | -         | -         | 4         | -         | -         | -         | -         | -         | -         | -         | -         | -         | 4         | -         | -         |
| $O_{6,1}$ | 564       | -         | -         | -         | -         | 342       | -         | -         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{6,2}$ | -         | 2         | -         | -         | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         | -         |
| $O_{6,3}$ | -         | -         | -         | 1584      | -         | -         | -         | -         | -         | -         | 2161      | -         | -         | -         | -         | -         |
| $O_{6,4}$ | -         | -         | -         | 5         | -         | -         | -         | -         | -         | -         | -         | -         | 5         | -         | -         | -         | -         |
**Appendix B: Values for decision variables corresponding to a feasible schedule**

**Table B.1**

Sets of decision variables taking value of 1. Variables not listed in this table take values of 0.

| Job shop line | Sets of decision variables taking value of 1. Variables not listed in this table take values of 0. |
|---------------|----------------------------------------------------------------------------------------------------------|
| **#1**        | \( \{ X_1, X_6 \} \), \( \{ Y_{4,1}, Y_{4,2}, Y_{4,3,1}, Y_{4,4,1}, Y_{6,1,1}, Y_{6,2,2}, Y_{6,3,1}, Y_{6,4,2} \} \), \( \{ Z_1^{1,2,1}, Z_1^{1,3,2}, Z_2^{1,3,1}, Z_1^{1,6,1}, Z_1^{1,8,2}, Z_2^{1,8,1} \} \) |
| **#2**        | \( \{ X_1^2, X_2^2, X_3^2, X_5^2 \} \), \( \{ Y_{1,1,1}, Y_{1,2,1}, Y_{1,3,1}, Y_{1,4,1}, Y_{1,5,1}, Y_{1,6,1}, Y_{1,7,1}, Y_{1,8,1}, Y_{2,2,2}, Y_{2,3,1}, Y_{2,4,2}, Y_{3,3,3}, Y_{3,4,3}, Y_{3,5,2}, Y_{3,6,2}, Y_{3,7,2}, Y_{3,8,2}, Y_{4,2,2}, Y_{4,3,3}, Y_{4,4,3} \} \), \( \{ Z_1^{2,2,1}, Z_1^{2,3,1}, Z_2^{2,3,4}, Z_3^{2,3,1}, Z_4^{2,3,3}, Z_1^{2,4,1}, Z_1^{2,5,1}, Z_1^{2,6,1}, Z_1^{2,8,2}, Z_2^{2,8,3}, Z_3^{2,8,1}, Z_4^{2,8,4} \} \) |
Appendix C: Job precedence rule-based distance metric

To measure distance between solutions, Job Precedence Rule based distance measure, i.e., JPR distance [60] was adopted. JPR distance is the number of job pairs with identical elements but different precedence. Denote $\pi_1 = [3, 1, 6, 4, 5, 2]$ and $\pi_2 = [3, 4, 6, 1, 5, 2]$ as two permutations, and $D(\pi_1, \pi_2)$ as JPR distance between $\pi_1$ and $\pi_2$. For instance, we extract all pairs of jobs with precedence from $\pi_1$ and $\pi_2$. For $\pi_1$, we extracted (3,1), (3,6), (3,4), (3,5), (3,2), (1,6), (1,4), (1,5), (1,2), (6,4), (6,5), (6,2), (4,5), (4,2), and (5,2). For $\pi_2$, we extracted (3,4), (3,6), (3,1), (3,5), (3,2), (4,6), (4,1), (4,5), (4,2), (6,1), (6,5), (6,2), (1,5), (1,2), and (5,2). For the job pairs of 1 and 4, 1 precedes 4 in $\pi_1$, while 1 succeeds 4 in $\pi_2$, which adds a distance of 1. Similarly another two job pairs (i.e., 4 and 6), 4 and 6 show different precedence in $\pi_1$ and $\pi_2$. Thus, their JPR distance is 3. We normalized the JPR as $DIST(\pi_1, \pi_2) = D(\pi_1, \pi_2)/D_{\text{max}}$, where $D_{\text{max}} = n(n-1)/2$. Finally, the normalized JPR distance $DIST(\pi_1, \pi_2) = 3/(6 \times 5 / 2) = 0.2$. 
## Appendix D: Concepts in local optima network

**Table D.1**

Concepts in local optima network.

| Concepts                     | Explanation                                                                                                                                 |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Node                         | Represents local optimum.                                                                                                                   |
| Escape edge                  | Escape edge exists between two nodes if a better local optimum can be obtained by performing search from the starting point of a local optimum. Escape edge is directed from the starting node to the ending node. |
| Edge weight                  | For escape edge, it is the frequency at which the directed transition between connected nodes occurs.                                         |
| Edge width                   | Proportional to edge weight.                                                                                                                 |
| Node size                    | Proportional to weighted incoming degree, reflecting the node attraction.                                                                     |
| Funnel structure             | A group of local optima that converge to a single solution.                                                                                   |
| Big-valley structure (globally convex structure) | Local optima are clustered around a global optimum. It is a special case of funnel structure with only one funnel. |
| Plateau                      | A higher-level virtual node compressing a group of local optima with same objective values belonging to a connected component according to the escape edge. It can be further divided into attractor plateaus and sink plateaus. |
| Attractor plateau            | A plateau that can be improved.                                                                                                               |
| Sink plateau (Funnel bottom) | A plateau with the best objective value in the funnel.                                                                                       |
| Plateau average size         | Average number of local optima within each plateau.                                                                                         |
Appendix E: Recording nodes and edges for local optima network

Two basic elements for a visualized local optima network (LON) are nodes and edges. Nodes and edges found during the search on an instance by an optimization algorithm were recorded. For generating a LON, each optimization algorithm is independently repeated \( N \) times on each instance. The algorithm is terminated if the best-so-far solution has been unimproved after \( M \) iterations. We use node set \( L \) to store all newly-found improved local optima, and edge set \( E \) to record all transitions starting from a local optimum and ending at another optimum. If a local optimum/edge occurs multiple times, duplicates were deleted; only one was kept. The incoming degree for a node (node size in LON) was recorded, as well as frequency (edge width in LON) at which the directed transition between connected nodes occurs. Two parameters were set: for toy instances, the settings in Ochoa and Veerapen [9], meaning that the number of independent runs \( N = 1000 \) and maximum number of iterations \( M = 10,000 \); for medium- and large-size instances, \( N = 50 \) and \( M = 1000 \) were used to save computational cost.
Appendix F: Expectation and variance of inter-neighbor distance under different neighborhoods

We prove the expectation and variance of inter-neighbor distance under element-based perturbation (Swap, Insert and Inverse) and block-based perturbation (Block-based Insert). See Section 5.6.2 for neighborhood definitions.

F.1. Inter-neighbor distance under element-based perturbation

We assume that any neighbor could be reached by same probability, and define a discrete random variable \( \xi \) to represent the number of elements between any two distinct elements in a vector of specimen sequence (VSS). Two distinct elements \( J_1 \) and \( J_2 \), randomly selected from the VSS, are located in the \( s \)-th and \( (s + \xi + 1) \)-th position, respectively. \( d(\xi) \) is the inter-neighbor distance, and \( p(\xi) \) is the probability distribution function. The mean and variance of \( d(\xi) \) can be defined as

\[
ED = \sum d(\xi)p(\xi) \tag{F.1}
\]

\[
VarD = \sum p(\xi)(d(\xi) - ED)^2. \tag{F.2}
\]

F.1.1. Inter-neighbor distance under swap

That \( J_1 \) and \( J_2 \) are swapped will change \( (2\xi + 1) \) element pairs’ precedence, leading to

\[
d(\xi) = 2(2\xi + 1)/(n \cdot (n-1)) \tag{F.3}
\]

under the definition of inter-solution distance metric (see Appendix C). For any two randomly selected elements, the probability that the former is \( \xi \) positions ahead of the latter one is

\[
p(\xi) = 2(n - \xi - 1)/(n \cdot (n-1)) \tag{F.4}
\]

where \( \xi \in \{0, 1, ..., n-2\} \). We obtain

\[
ED_{SWP} = 2(2n-1)/(3n \cdot (n-1)) \tag{F.5}
\]

\[
VarD_{SWP} = 8(n + 1)(n - 2)/(9n^2(n - 1)^2) \tag{F.6}
\]

F.1.2. Inter-neighbor distance under insert

Inserting \( J_2 \) before \( J_1 \) leads to

\[
d(\xi) = 2(\xi + 1)/(n(n-1)) \tag{F.7}
\]

and \( p(\xi) \) remains the same as before. We obtain

\[
ED_{INS} = 2(n + 1)/(3n \cdot (n-1)) \tag{F.8}
\]

\[
VarD_{INS} = 2(n + 1)(n - 2)/(9n^2(n - 1)^2) \tag{F.9}
\]

F.1.3. Inter-neighbor distance under inverse

Inverting the subsequence between \( J_1 \) and \( J_2 \) leads to

\[
d(\xi) = \xi(\xi - 1)/(n(n-1)) \tag{F.10}
\]
and $p(\xi)$ remains the same as before. We obtain

$$ED_{INV} = (n-2)(n-3)/(6n(n-1))$$  \hspace{1cm} (F.11)

$$VarD_{INV} = (n+1)(n-2)(n-3)(7n-18)/(180n^2(n-1)^2)$$  \hspace{1cm} (F.12)

**F.2. Inter-neighbor distance under block-based insert**

The $n$-length sequence is split into $b$ blocks through diversification generation method (Section 5.3). Each block is composed of $n_c$ elements/specimens. Blocks are numbered in order, left to right, and each block’s serial number remains unchanged. Block-based Insert operates on these blocks, randomly choosing from the sequence two distinct blocks and inserting the latter block before the front block while keeping unchanged specimens’ priority within each block. When operating on the sequence, some elements belonging to the same block may be scattered in certain positions in the sequence, resulting in these elements not being connected with other elements in the same block, but the membership of these elements remains constant. We consider two situations, one is that specimens belonging to the same block are not scattered in the sequence, as shown in Fig. 9(d) of Section 5.6.2; the other is that specimens belonging to the same block are scattered, as shown in Fig. 9(e).

**F.2.1. Elements belonging to the same block are not scattered**

Two distinct blocks $B_g$ and $B_h$ are randomly selected. $B_g$ is before $B_h$. $\kappa$, a discrete random variable, represents that there are $\kappa$ blocks between $B_g$ and $B_h$. It leads to an inter-neighbor distance

$$d(\kappa) = 2(k+1)n_c^2/(n(n-1)) = 2(k+1)n/(b^2(n-1)),$$  \hspace{1cm} (F.13)

since $(\kappa+1)n_c^2$ specimen pairs’ precedence are changed and a probability distribution function is calculated with

$$p(\kappa) = 2(b-\kappa-1)/(b(b-1)),$$  \hspace{1cm} (F.14)

where $\kappa \in \{0,1,...,b-2\}$.

To obtain

$$ED_{INB} = 2n(b+1)/(3b^2(n-1)),$$  \hspace{1cm} (F.15)

$$VarD_{INB} = 2(b+1)(b-2)n^2/(9b^4(n-1)^2).$$  \hspace{1cm} (F.16)

**F.2.2. Elements belonging to the same block are scattered**

Same as the previous definitions, blocks $B_g$ and $B_h$ are two randomly selected blocks, and $B_g$ is before $B_h$. Elements in $B_g$ and $B_h$ are indexed in order as $\{i_{g,1},...,i_{g,q},...,i_{g,n_c}\}$ and $\{i_{h,1},...,i_{h,q},...,i_{h,n_h}\}$, respectively; $i_{g,q}$ is valued as the position of its corresponding element in whole sequence. It leads to an inter-neighbor distance

$$d(g,h) = \sum_{q=1}^{n_c} 2(i_{h,q} - i_{g,q})/(n(n-1)).$$  \hspace{1cm} (F.17)

Probability that both the $g$-th and the $h$-th blocks are selected is

$$p(g,h) = 2/(b(b-1)).$$  \hspace{1cm} (F.18)

Inter-neighbor distance’s expectation is
\[ ED = \sum_{g=0}^{b} \sum_{q=1}^{n_c} \frac{4(i_{h,g} - i_{g,1})}{n(n-1)b(b-1)} . \]  
(F.19)

Its second moment is
\[ ED^2 = \sum_{h=2}^{b} \sum_{g=1}^{h-1} \frac{1}{n(n-1)b(b-1)} \left( \sum_{q=1}^{n_c} i_{h,g} - i_{g,1} \right)^2 . \]  
(F.20)

Its variance is
\[ VarD = ED^2 - E^2D . \]  
(F.21)

Since it is hard to derive explicit expectation and variance, we provide the upper and lower bounds.

**Expectation’s upper bound:** Since \( i_{g,1} \geq g \) and \( \sum_{h=2}^{b} \sum_{g=1}^{n_c} i_{h,g} \leq \sum_{r=1}^{n_c} (n_r + r) \), we obtain
\[ \sum_{1 \leq g \leq h \leq b} \sum_{q=1}^{n_c} i_{h,g} = \sum_{g=1}^{b} \sum_{h=g+1}^{b} \sum_{q=1}^{n_c} i_{h,g} \leq \sum_{g=1}^{b} \sum_{r=1}^{n_c} \sum_{g=1}^{n_c} (n_r + r) . \]  
(F.22)

The upper bound is
\[ ED = \sum_{g=1}^{n_c} \sum_{h=g+1}^{b} \sum_{q=1}^{n_c} 4(i_{h,g} - i_{g,1})/(n(n-1)b(b-1)) \leq \frac{4n_c}{n(n-1)b(b-1)} \sum_{g=1}^{b-1} (g(b-g)n_c + (b-g)(n_c+1)/2 - g(b-g)) \]
\[ = \frac{2n_c}{n(n-1)b(b-1)} \sum_{g=1}^{b-1} (b(bn_c + 1) - (2b+1)g + (n_c + 2)g^2) \]
\[ = \frac{n(4bn_c + n_c - 2b + 1)}{3n(n-1)} \]
\[ = \frac{(4b+1)n + (-2b + 1)b}{3b^2(n-1)} . \]
(F.23)

**Expectation’s lower bound:** Since \( i_{g,1} \leq (g-1)n_c + 1 \), we obtain
\[ \sum_{1 \leq g \leq h \leq b} \sum_{q=1}^{n_c} i_{h,g} = \sum_{g=1}^{b} \sum_{h=g+1}^{b} \sum_{q=1}^{n_c} i_{h,g} \geq \sum_{g=1}^{b-1} \sum_{r=1}^{n_c} (n_r + r) . \]  
(F.24)

The lower bound is
\[ ED = \sum_{g=1}^{n_c} \sum_{h=g+1}^{b} \sum_{q=1}^{n_c} 4(i_{h,g} - i_{g,1})/(n(n-1)b(b-1)) \geq \frac{4n_c}{n(n-1)b(b-1)} \sum_{g=1}^{b-1} (g(b-g)n_c + (b-g)(n_c+1)/2 - (g-1)n_c + 1)(b-g) \]
\[ = \frac{2n_c}{n(n-1)b(b-1)} \sum_{g=1}^{b-1} (b^2n_c + 2bn_c - b) + (-4bn_c - 2n_c + 2b + 1)g + (3n_c - 2)g^2 . \]  
(F.25)

**Variance’s upper bound:** The following inequalities holds for any sequence.

42
\[ i_{g, t} \geq g \]  
(F.26)

\[
\sum_{g=1}^{b-1} \sum_{h=g+1}^{b} a_h^2 \leq \sum_{g=1}^{b-1} \sum_{h=g+1}^{b} \left( \sum_{q=1}^{n_q} (n_q (h-1) + q) \right)^2 
\leq \sum_{g=1}^{b-1} \sum_{h=g+1}^{b} a_h \geq \sum_{g=1}^{b-1} \sum_{i=1}^{n_i} \left( g + \tau \right) 
\]  
(F.27)

where \( a_h = \sum_{q=1}^{n_q} i_{h, q} \).

The upper bound of the second moment is

\[
ED^2 = \frac{8}{n^2(n-1)^2b(b-1)} \sum_{q=1}^{b-1} \sum_{h=g+1}^{b} \left( a_h - n_i i_{g, t} \right)^2
\]

\[
\leq \frac{8}{n^2(n-1)^2b(b-1)} \sum_{q=1}^{b-1} \sum_{h=g+1}^{b} \left( a_h^2 - 2n_q a_h + n_q g^2 \right)
\]

\[
\leq \frac{8}{n^2(n-1)^2b(b-1)} \sum_{q=1}^{b-1} \sum_{h=g+1}^{b} \left( \sum_{q=1}^{n_q} (n_q (h-1) + q) \right)^2 - 2n_q \sum_{q=1}^{n_q} a_h (g + \tau) + n_q g^2 (b - g)
\]

\[
= \frac{2}{3n^2(n-1)^2} \left( 6b^3 + 2b - 2 \right) n^2 - \left( 2b^2 + 6b + 2 \right) bn - \left( 2b^2 + 6b + 1 \right) b^2
\]

Since the distance is normalized to [0, 1], the upper bound of the variance of inter-neighbor distance is

\[
VarD \leq \min \left\{ 1, \frac{\left( 6b^2 + 2b - 2 \right) n^2 - \left( 2b^2 + 6b + 2 \right) bn - \left( 2b^2 + 6b + 1 \right) b^2}{3b^4(n-1)^2} \right\} .
\]

\[
\leq \min \left\{ 1, \frac{6(3b^2 + b - 2) n^2 - 6(b - 2)bn - 2(5b + 2) b^2}{9b^4(n-1)^2} \right\}
\]

**Variance’s lower bound:** The lower bound of the second moment of inter-neighbor distance is

\[
ED^2 = \frac{8}{n^2(n-1)^2b(b-1)} \sum_{q=1}^{b-1} \sum_{h=g+1}^{b} \left( a_h - n_i i_{g, t} \right)^2 
\geq \frac{8}{n^2(n-1)^2b(b-1)} \frac{2}{b(b-1)} \sum_{q=1}^{b-1} \sum_{h=g+1}^{b} \sum_{q=1}^{n_q} \left( h_i - i_{g, t} \right)^2
\]

\[
\geq \frac{16}{n^2(n-1)^2b^2(b-1)^2} \sum_{q=1}^{b-1} \sum_{h=g+1}^{b} \sum_{q=1}^{n_q} \left( h_i - i_{g, t} \right)^2
\]

\[
= \frac{4n_i^2}{n^2(n-1)^2b^2(b-1)} \sum_{q=1}^{b-1} (b + b^2 n_q - (2bn_q + 1)g + n_i g^2)^2
\]

\[
= \frac{2n_i^2 \left( 6b^3 - 9b^2 + b + 1 \right) n^2 + 15(b-1)bn_q + (10b - 5) \right)}{15n^2(n-1)^2b^2(b-1)^2}
\]

Lower bound for the variance of inter-neighbor distance is

\[
= \frac{2 \left( 6b^3 - 9b^2 + b + 1 \right) n^2 + 15(b-1) \left( 2b - 1 \right) b^2}{15(n-1)^2b^2(b-1)}
\]
$$VarD \geq \max \left\{ 0, \min ED^2 - \max E^2D \right\}$$

$$\geq \max \left\{ 0, \frac{2 \left( 6b^3 - 9b^2 + b + 1 \right) n^2 + 15(b - 1)b^2n + 5(2b - 1)b^2}{15(n - 1)^2 b^2(b - 1)} \right\} - \left( \frac{(4b + 1)n + (-2b + 1)b}{3b^2(n - 1)} \right)^2$$

$$= \max \left\{ 0, \frac{-80b^4 - 4b^3 - 89b^2 + b + 6}{45(n - 1)^2 b^3(b - 1)} \right\}$$

(F.32)
**Appendix G: Performances of SSMAs on medium-and large-size instances**

Table G.1
Performance of SSMAs on instances with 100 tasks as the ratio of biochemical tests to immunologic tests is 1:4

| Instance   | Performance metrics | SMA-INV | SMA-INS | SMA-SWP | SMA-INB | SMA-ML | CPU TIME |
|------------|---------------------|---------|---------|---------|---------|--------|----------|
| 20_80_1    | BRE                 | 1.33    | 1.21    | 0.00    | 3.16    | 0.92   |          |
|            | ARE                 | 1.59    | 0.98    | 0.00    | 2.17    | 1.02   |          |
|            | WRE                 | 0.00    | 0.35    | 0.49    | 0.13    | 0.00   |          |
|            | CPU TIME            | 23.33   | 23.37   | 19.07   | 18.95   | 19.20  |          |
| 20_80_2    | BRE                 | 0.99    | 0.53    | 0.00    | 1.15    | 0.09   |          |
|            | ARE                 | 0.92    | 0.72    | 0.00    | 0.75    | 0.56   |          |
|            | WRE                 | 0.81    | 0.95    | 0.00    | 0.75    | 0.63   |          |
|            | CPU TIME            | 18.95   | 18.92   | 18.95   | 19.12   | 19.33  |          |
| 20_80_3    | BRE                 | 2.56    | 0.46    | 0.00    | 3.35    | 0.42   |          |
|            | ARE                 | 1.68    | 1.41    | 0.00    | 1.61    | 0.91   |          |
|            | WRE                 | 0.87    | 1.31    | 0.15    | 0.00    | 0.29   |          |
|            | CPU TIME            | 19.31   | 18.89   | 18.82   | 19.04   | 19.01  |          |
| 20_80_4    | BRE                 | 2.21    | 0.56    | 0.15    | 2.31    | 0.00   |          |
|            | ARE                 | 1.55    | 1.10    | 0.00    | 1.18    | 1.10   |          |
|            | WRE                 | 1.21    | 1.24    | 0.00    | 0.51    | 0.83   |          |
|            | CPU TIME            | 19.18   | 18.99   | 19.23   | 19.15   | 18.95  |          |
| 20_80_5    | BRE                 | 1.77    | 1.09    | 0.02    | 2.28    | 0.00   |          |
|            | ARE                 | 1.50    | 1.01    | 0.00    | 1.32    | 1.12   |          |
|            | WRE                 | 0.93    | 1.17    | 0.00    | 0.03    | 0.60   |          |
|            | CPU TIME            | 18.90   | 18.87   | 18.94   | 19.16   | 19.26  |          |
| 20_80_6    | BRE                 | 2.11    | 1.17    | 0.00    | 3.44    | 1.43   |          |
|            | ARE                 | 1.76    | 1.12    | 0.00    | 2.04    | 1.47   |          |
|            | WRE                 | 0.35    | 0.23    | 0.00    | 0.14    | 0.14   |          |
|            | CPU TIME            | 19.00   | 19.04   | 19.11   | 19.06   | 19.26  |          |
| 20_80_7    | BRE                 | 1.40    | 1.03    | 0.00    | 2.79    | 1.18   |          |
|            | ARE                 | 1.74    | 0.91    | 0.00    | 2.25    | 1.12   |          |
|            | WRE                 | 1.18    | 0.48    | 0.00    | 1.34    | 1.00   |          |
|            | CPU TIME            | 18.99   | 18.86   | 19.32   | 19.06   | 19.01  |          |
| 20_80_8    | BRE                 | 2.42    | 1.41    | 0.00    | 3.81    | 0.68   |          |
|            | ARE                 | 1.74    | 1.09    | 0.00    | 2.55    | 1.14   |          |
|            | WRE                 | 1.33    | 1.27    | 0.00    | 1.54    | 1.53   |          |
|            | CPU TIME            | 19.18   | 18.99   | 19.04   | 18.94   | 18.98  |          |
| 20_80_9    | BRE                 | 2.86    | 0.82    | 0.00    | 3.64    | 0.71   |          |
|            | ARE                 | 1.54    | 0.96    | 0.00    | 1.59    | 1.13   |          |
|            | WRE                 | 0.70    | 0.72    | 0.00    | 0.27    | 0.30   |          |
|            | CPU TIME            | 18.80   | 19.01   | 18.84   | 19.03   | 19.10  |          |
| 20_80_10   | BRE                 | 1.98    | 2.44    | 0.00    | 3.19    | 1.73   |          |
|            | ARE                 | 1.41    | 1.24    | 0.00    | 1.28    | 1.12   |          |
|            | WRE                 | 0.53    | 0.44    | 0.00    | 0.24    | 0.33   |          |
|            | CPU TIME            | 18.88   | 19.03   | 18.97   | 19.13   | 19.42  |          |
### Table G.2
Performance of SSMAs on instances with 100 tasks as the ratio of biochemical tests to immunologic tests is 1:1

| Instance | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|---------|
| 50_50_1  | BRE                 | 1.89     | 0.76     | 0.00     | 1.16     | 0.17    |
|          | ARE                 | 1.78     | 0.86     | 0.00     | 0.82     | 0.42    |
|          | WRE                 | 1.60     | 1.23     | 0.00     | 1.17     | 0.99    |
|          | CPU TIME            | 19.87    | 19.89    | 19.64    | 19.90    | 19.94   |
| 50_50_2  | BRE                 | 1.91     | 0.77     | 0.07     | 0.93     | 0.00    |
|          | ARE                 | 2.01     | 0.84     | 0.00     | 1.51     | 0.78    |
|          | WRE                 | 1.68     | 0.81     | 0.00     | 2.14     | 0.87    |
|          | CPU TIME            | 19.66    | 19.94    | 19.66    | 19.70    | 19.88   |
| 50_50_3  | BRE                 | 1.26     | 0.33     | 0.09     | 0.51     | 0.00    |
|          | ARE                 | 1.47     | 0.51     | 0.00     | 0.51     | 0.24    |
|          | WRE                 | 1.33     | 1.11     | 0.00     | 0.38     | 0.63    |
|          | CPU TIME            | 19.72    | 19.93    | 19.84    | 19.87    | 19.94   |
| 50_50_4  | BRE                 | 0.90     | 0.62     | 0.00     | 0.42     | 0.17    |
|          | ARE                 | 1.44     | 0.69     | 0.00     | 0.74     | 0.42    |
|          | WRE                 | 1.30     | 1.15     | 0.00     | 0.69     | 0.54    |
|          | CPU TIME            | 19.76    | 19.85    | 19.65    | 19.72    | 19.82   |
| 50_50_5  | BRE                 | 1.53     | 0.66     | 0.15     | 2.03     | 0.00    |
|          | ARE                 | 1.78     | 0.83     | 0.00     | 1.88     | 0.39    |
|          | WRE                 | 1.73     | 0.96     | 0.00     | 1.53     | 1.40    |
|          | CPU TIME            | 19.79    | 19.69    | 19.73    | 19.87    | 19.78   |
| 50_50_6  | BRE                 | 1.02     | 0.48     | 0.00     | 0.58     | 0.16    |
|          | ARE                 | 1.08     | 0.52     | 0.00     | 0.24     | 0.13    |
|          | WRE                 | 0.70     | 0.65     | 0.00     | 0.26     | 0.18    |
|          | CPU TIME            | 19.83    | 19.93    | 19.66    | 19.89    | 20.15   |
| 50_50_7  | BRE                 | 0.83     | 0.49     | 0.00     | 0.05     | 0.03    |
|          | ARE                 | 0.83     | 0.74     | 0.17     | 0.00     | 0.14    |
|          | WRE                 | 0.84     | 0.84     | 0.46     | 0.00     | 0.81    |
|          | CPU TIME            | 19.88    | 19.99    | 19.71    | 19.84    | 19.96   |
| 50_50_8  | BRE                 | 1.74     | 0.62     | 0.00     | 1.55     | 0.41    |
|          | ARE                 | 1.95     | 0.91     | 0.00     | 1.45     | 0.62    |
|          | WRE                 | 2.07     | 1.69     | 0.00     | 1.45     | 1.40    |
|          | CPU TIME            | 19.64    | 19.83    | 20.10    | 20.44    | 19.96   |
| 50_50_9  | BRE                 | 2.63     | 1.61     | 0.00     | 2.39     | 0.70    |
|          | ARE                 | 2.23     | 1.23     | 0.00     | 1.76     | 0.77    |
|          | WRE                 | 2.30     | 1.84     | 0.00     | 1.31     | 1.28    |
|          | CPU TIME            | 19.74    | 19.81    | 19.81    | 19.91    | 19.80   |
| 50_50_10 | BRE                 | 1.53     | 0.48     | 0.00     | 0.65     | 0.07    |
|          | ARE                 | 1.38     | 0.84     | 0.00     | 0.73     | 0.25    |
|          | WRE                 | 1.17     | 1.06     | 0.00     | 0.39     | 0.92    |
|          | CPU TIME            | 19.76    | 19.78    | 19.93    | 19.80    | 20.04   |
Table G.3

Performance of SSMAs on instances with 100 tasks as the ratio of biochemical tests to immunologic tests is 4:1

| Instance | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|---------|
| 80_20_1  | BRE                 | 0.32     | 0.47     | 0.16     | 0.43     | 0.00    |
|          | ARE                 | 0.61     | 0.43     | 0.37     | 0.00     | 0.14    |
|          | WRE                 | 1.14     | 0.87     | 0.84     | 0.00     | 0.90    |
|          | CPU TIME            | **18.87**| 19.06    | 19.02    | 19.03    | 19.06   |
| 80_20_2  | BRE                 | 0.44     | 0.61     | 0.45     | 0.53     | 0.00    |
|          | ARE                 | 0.37     | 0.19     | 0.30     | 0.32     | 0.00    |
|          | WRE                 | 0.70     | 0.47     | 0.49     | 0.00     | 0.37    |
|          | CPU TIME            | 18.91    | 19.06    | **18.85**| 18.88    | 18.94   |
| 80_20_3  | BRE                 | 0.21     | 0.23     | 0.61     | 0.92     | 0.00    |
|          | ARE                 | 0.24     | **0.00** | 0.03     | 0.51     | 0.09    |
|          | WRE                 | 0.50     | 0.35     | 0.00     | 0.33     | 0.56    |
|          | CPU TIME            | 19.09    | 19.05    | 19.14    | **19.04**| 19.05   |
| 80_20_4  | BRE                 | 0.56     | 0.49     | 0.00     | 0.96     | 0.11    |
|          | ARE                 | 0.58     | 0.22     | 0.00     | 0.38     | 0.02    |
|          | WRE                 | 0.49     | 0.33     | 0.00     | 0.33     | 0.16    |
|          | CPU TIME            | **18.81**| 18.97    | 19.07    | 18.89    | 19.22   |
| 80_20_5  | BRE                 | 0.03     | 0.07     | 0.30     | 0.03     | 0.00    |
|          | ARE                 | 0.71     | 0.79     | 0.73     | 0.00     | 0.32    |
|          | WRE                 | 1.15     | 1.15     | 0.98     | 0.00     | 1.04    |
|          | CPU TIME            | **18.88**| 19.01    | 19.12    | **18.88**| 19.05   |
| 80_20_6  | BRE                 | 0.66     | 0.37     | 0.42     | 0.54     | 0.00    |
|          | ARE                 | 0.26     | 0.30     | 0.09     | 0.03     | 0.00    |
|          | WRE                 | 0.53     | 0.44     | 0.34     | 0.00     | 0.38    |
|          | CPU TIME            | 19.06    | **18.93**| 19.00    | 19.04    | 18.96   |
| 80_20_7  | BRE                 | 0.61     | 0.28     | 0.00     | 1.02     | 0.09    |
|          | ARE                 | 0.50     | 0.08     | 0.00     | 0.50     | 0.13    |
|          | WRE                 | 0.47     | 0.14     | 0.06     | 0.00     | 0.38    |
|          | CPU TIME            | **18.97**| 19.20    | 19.12    | 19.01    | 19.22   |
| 80_20_8  | BRE                 | 0.04     | **0.00** | 0.02     | 0.37     | 0.00    |
|          | ARE                 | 0.21     | 0.16     | **0.00** | 0.04     | 0.02    |
|          | WRE                 | 0.42     | 0.37     | 0.32     | 0.00     | 0.34    |
|          | CPU TIME            | 19.23    | **18.91**| 18.97    | **18.91**| 19.00   |
| 80_20_9  | BRE                 | 0.59     | 0.81     | 0.52     | 0.42     | 0.00    |
|          | ARE                 | 0.66     | 0.72     | 0.50     | 0.00     | 0.26    |
|          | WRE                 | 0.70     | 0.71     | 0.50     | **0.00** | 0.57    |
|          | CPU TIME            | 18.97    | 18.87    | **18.83**| 18.93    | 19.08   |
| 80_20_10 | BRE                 | 0.63     | 0.79     | 0.94     | 0.30     | 0.00    |
|          | ARE                 | 0.91     | 0.86     | 0.76     | **0.00** | 0.35    |
|          | WRE                 | 1.06     | 1.06     | 1.05     | **0.00** | 1.05    |
|          | CPU TIME            | 18.89    | 19.16    | 18.84    | **18.83**| 19.05   |
Table G.4
Performance of SSMAs on instances with 200 tasks as the ratio of biochemical tests to immunologic tests is 1:4

| Instance   | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|------------|---------------------|----------|----------|----------|----------|---------|
| 40_160_1   | BRE                 | 1.67     | 1.37     | 0.38     | 0.00     | 0.02    |
|            | ARE                 | 2.24     | 2.51     | 1.73     | 0.00     | 1.11    |
|            | WRE                 | 1.44     | 1.62     | 1.54     | 0.00     | 1.17    |
|            | CPU TIME            | 56.14    | 56.64    | 56.28    | 56.20    | 52.87   |
| 40_160_2   | BRE                 | 1.74     | 1.72     | 0.00     | 1.15     | 1.13    |
|            | ARE                 | 1.21     | 1.31     | 0.51     | 0.00     | 0.47    |
|            | WRE                 | 1.55     | 2.22     | 2.70     | 0.00     | 1.40    |
|            | CPU TIME            | 56.09    | 56.67    | 55.90    | 56.78    | 52.60   |
| 40_160_3   | BRE                 | 1.58     | 1.66     | 0.00     | 1.69     | 1.63    |
|            | ARE                 | 1.09     | 1.50     | 0.53     | 0.00     | 0.55    |
|            | WRE                 | 1.34     | 1.46     | 1.36     | 0.00     | 1.37    |
|            | CPU TIME            | 56.59    | 55.87    | 56.38    | 56.55    | 52.43   |
| 40_160_4   | BRE                 | 1.35     | 1.15     | 0.00     | 0.64     | 0.73    |
|            | ARE                 | 1.20     | 1.20     | 0.40     | 0.00     | 0.50    |
|            | WRE                 | 1.72     | 2.39     | 1.76     | 0.00     | 1.83    |
|            | CPU TIME            | 55.80    | 56.26    | 56.62    | 56.36    | 53.31   |
| 40_160_5   | BRE                 | 1.48     | 1.37     | 0.00     | 0.19     | 0.92    |
|            | ARE                 | 1.45     | 1.86     | 0.85     | 0.00     | 0.97    |
|            | WRE                 | 1.62     | 2.38     | 1.43     | 0.00     | 1.54    |
|            | CPU TIME            | 56.61    | 56.45    | 55.87    | 55.93    | 52.74   |
| 40_160_6   | BRE                 | 2.92     | 1.87     | 0.00     | 0.42     | 0.83    |
|            | ARE                 | 2.77     | 3.06     | 1.84     | 0.00     | 1.15    |
|            | WRE                 | 3.36     | 3.57     | 2.78     | 0.00     | 2.62    |
|            | CPU TIME            | 56.55    | 56.41    | 56.67    | 56.43    | 52.51   |
| 40_160_7   | BRE                 | 2.03     | 1.78     | 0.65     | 0.12     | 0.00    |
|            | ARE                 | 2.78     | 2.44     | 1.83     | 0.00     | 1.09    |
|            | WRE                 | 2.90     | 2.86     | 2.74     | 0.00     | 2.50    |
|            | CPU TIME            | 56.50    | 55.95    | 56.22    | 56.51    | 53.52   |
| 40_160_8   | BRE                 | 0.77     | 1.09     | 0.77     | 0.00     | 0.02    |
|            | ARE                 | 2.23     | 2.50     | 1.66     | 0.00     | 0.79    |
|            | WRE                 | 2.60     | 3.27     | 2.41     | 0.00     | 2.43    |
|            | CPU TIME            | 56.36    | 56.25    | 56.10    | 56.62    | 52.79   |
| 40_160_9   | BRE                 | 1.41     | 1.57     | 0.00     | 0.03     | 0.65    |
|            | ARE                 | 2.27     | 2.37     | 1.14     | 0.00     | 0.89    |
|            | WRE                 | 2.65     | 3.33     | 2.61     | 0.00     | 2.23    |
|            | CPU TIME            | 55.80    | 56.20    | 55.99    | 56.36    | 52.89   |
| 40_160_10  | BRE                 | 1.39     | 0.00     | 0.09     | 0.11     | 0.40    |
|            | ARE                 | 1.33     | 1.26     | 0.58     | 0.00     | 0.56    |
|            | WRE                 | 1.88     | 2.15     | 1.70     | 0.00     | 1.42    |
|            | CPU TIME            | 56.14    | 56.77    | 56.16    | 56.41    | 52.56   |
| Instance  | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|-----------|---------------------|----------|----------|----------|----------|----------|
| 100_100_1 | BRE                 | 2.27     | 2.38     | 1.67     | 0.00     | 0.19     |
|           | ARE                 | 2.00     | 2.04     | 1.84     | 0.00     | 0.90     |
|           | WRE                 | 1.78     | 1.90     | 1.79     | 0.00     | 1.83     |
|           | CPU TIME            | 56.12    | 56.23    | 56.27    | 56.17    | 52.28    |
| 100_100_2 | BRE                 | 0.47     | 0.47     | 0.47     | 0.06     | 0.00     |
|           | ARE                 | 0.34     | 0.23     | 0.28     | 0.00     | 0.15     |
|           | WRE                 | 0.83     | 0.83     | 0.83     | 0.00     | 0.68     |
|           | CPU TIME            | 56.36    | 56.19    | 56.02    | 56.44    | 52.34    |
| 100_100_3 | BRE                 | 2.44     | 2.31     | 1.70     | 0.00     | 0.23     |
|           | ARE                 | 2.23     | 2.55     | 1.74     | 0.00     | 0.83     |
|           | WRE                 | 2.08     | 2.25     | 1.65     | 0.00     | 1.75     |
|           | CPU TIME            | 56.79    | 56.39    | 56.42    | 56.42    | 52.71    |
| 100_100_4 | BRE                 | 1.56     | 1.89     | 1.09     | 0.00     | 0.25     |
|           | ARE                 | 1.46     | 1.48     | 1.32     | 0.00     | 0.78     |
|           | WRE                 | 1.29     | 1.29     | 1.26     | 0.00     | 1.26     |
|           | CPU TIME            | 56.08    | 56.37    | 56.05    | 56.74    | 52.60    |
| 100_100_5 | BRE                 | 1.55     | 1.54     | 1.22     | 0.03     | 0.00     |
|           | ARE                 | 1.17     | 1.19     | 1.13     | 0.00     | 0.63     |
|           | WRE                 | 1.45     | 1.55     | 1.27     | 0.00     | 1.49     |
|           | CPU TIME            | 56.31    | 56.18    | 56.24    | 56.23    | 52.76    |
| 100_100_6 | BRE                 | 1.86     | 2.02     | 1.91     | 0.00     | 0.39     |
|           | ARE                 | 1.47     | 1.49     | 1.49     | 0.00     | 0.91     |
|           | WRE                 | 1.03     | 1.03     | 1.03     | 0.00     | 1.03     |
|           | CPU TIME            | 55.99    | 56.54    | 55.87    | 56.66    | 52.37    |
| 100_100_7 | BRE                 | 1.36     | 1.30     | 1.07     | 0.00     | 0.34     |
|           | ARE                 | 0.85     | 0.87     | 0.85     | 0.00     | 0.53     |
|           | WRE                 | 0.48     | 0.63     | 0.63     | 0.00     | 0.05     |
|           | CPU TIME            | 56.69    | 56.39    | 56.16    | 56.90    | 52.65    |
| 100_100_8 | BRE                 | 0.69     | 1.70     | 0.55     | 0.00     | 0.09     |
|           | ARE                 | 1.34     | 1.73     | 1.10     | 0.00     | 0.94     |
|           | WRE                 | 1.33     | 1.33     | 0.60     | 0.00     | 1.11     |
|           | CPU TIME            | 56.29    | 56.31    | 55.93    | 56.61    | 53.01    |
| 100_100_9 | BRE                 | 1.61     | 1.58     | 1.44     | 0.00     | 0.30     |
|           | ARE                 | 1.40     | 1.39     | 1.36     | 0.00     | 0.77     |
|           | WRE                 | 1.22     | 1.17     | 1.21     | 0.00     | 1.16     |
|           | CPU TIME            | 56.75    | 55.98    | 56.20    | 56.20    | 52.57    |
| 100_100_10| BRE                | 1.75     | 1.87     | 1.36     | 0.11     | 0.00     |
|           | ARE                 | 1.71     | 1.73     | 1.48     | 0.00     | 0.81     |
|           | WRE                 | 1.81     | 1.59     | 1.33     | 0.00     | 1.46     |
|           | CPU TIME            | 57.14    | 56.41    | 56.52    | 56.19    | 53.09    |
## Table G.6
Performance of SSMAs on instances with 200 tasks as the ratio of biochemical tests to immunologic tests is 4:1

| Instance | BRE | ARE | WRE | CPU TIME | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|-----|-----|-----|----------|----------|----------|----------|----------|---------|
| 160_40_1 | 1.84 | 1.75 | 1.47 | 56.18 | 1.52 | 1.73 | 1.83 | 0.25 | 0.63 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.92 |
| 160_40_2 | 0.95 | 0.50 | 0.65 | 56.28 | 0.95 | 0.57 | 0.69 | 0.25 | 0.70 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.75 |
| 160_40_3 | 0.74 | 0.75 | 0.72 | 56.30 | 0.32 | 0.74 | 0.84 | 0.00 | 0.03 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.23 |
| 160_40_4 | 0.89 | 0.62 | 0.41 | 56.45 | 1.32 | 0.62 | 0.63 | 0.26 | 0.39 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.84 |
| 160_40_5 | 1.03 | 0.54 | 0.61 | 56.54 | 1.03 | 0.64 | 0.86 | 1.08 | 0.14 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.86 |
| 160_40_6 | 0.69 | 0.32 | 0.62 | 56.02 | 0.50 | 0.32 | 0.63 | 0.26 | 0.39 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.71 |
| 160_40_7 | 0.39 | 0.15 | 0.62 | 56.32 | 0.39 | 0.11 | 0.63 | 0.15 | 0.39 |
|          | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 52.28 |
| 160_40_8 | 1.08 | 1.15 | 0.74 | 56.62 | 1.39 | 1.16 | 0.73 | 0.23 | 0.71 |
|          | 0.00 | 0.00 | 0.00 | 57.19 | 0.00 | 0.00 | 0.00 | 52.39 |
| 160_40_9 | 1.19 | 0.83 | 0.83 | 55.93 | 1.19 | 0.83 | 0.83 | 1.19 | 0.87 |
|          | 0.00 | 0.00 | 0.00 | 56.07 | 0.00 | 0.00 | 0.00 | 52.49 |
| Instance  | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|---------|
| 60_240_1 | BRE                 | 1.31     | 2.02     | 0.69     | 0.03     | 0.00    |
|          | ARE                 | 1.76     | 2.12     | 1.81     | 0.00     | 0.89    |
|          | WRE                 | 0.94     | 1.13     | 1.12     | 0.00     | 0.84    |
|          | CPU TIME            | 122.48   | 121.66   | 122.74   | 122.78   | 122.99  |
| 60_240_2 | BRE                 | 2.60     | 2.68     | 2.94     | 0.00     | 1.19    |
|          | ARE                 | 2.50     | 2.82     | 2.47     | 0.00     | 1.56    |
|          | WRE                 | 2.25     | 2.73     | 2.64     | 0.00     | 1.80    |
|          | CPU TIME            | 122.12   | 122.81   | 122.72   | 122.52   | 123.05  |
| 60_240_3 | BRE                 | 1.33     | 2.25     | 1.30     | 0.29     | 0.00    |
|          | ARE                 | 1.68     | 2.24     | 1.79     | 0.00     | 0.58    |
|          | WRE                 | 1.56     | 1.79     | 1.87     | 0.00     | 1.41    |
|          | CPU TIME            | 122.82   | 122.90   | 121.97   | 124.14   | 122.30  |
| 60_240_4 | BRE                 | 2.06     | 2.20     | 1.31     | 0.44     | 0.00    |
|          | ARE                 | 1.54     | 1.82     | 1.42     | 0.00     | 0.55    |
|          | WRE                 | 1.17     | 1.34     | 1.27     | 0.00     | 1.30    |
|          | CPU TIME            | 121.86   | 123.56   | 122.03   | 122.29   | 122.53  |
| 60_240_5 | BRE                 | 2.17     | 2.01     | 2.41     | 0.30     | 0.00    |
|          | ARE                 | 2.04     | 2.53     | 2.25     | 0.00     | 1.10    |
|          | WRE                 | 1.05     | 1.35     | 1.34     | 0.00     | 1.13    |
|          | CPU TIME            | 121.97   | 122.41   | 122.30   | 121.90   | 123.65  |
| 60_240_6 | BRE                 | 0.31     | 0.36     | 0.48     | 0.32     | 0.00    |
|          | ARE                 | 1.14     | 1.15     | 1.14     | 0.00     | 0.66    |
|          | WRE                 | 1.22     | 1.01     | 1.58     | 0.00     | 1.35    |
|          | CPU TIME            | 122.16   | 122.21   | 122.23   | 123.39   | 122.46  |
| 60_240_7 | BRE                 | 2.47     | 0.90     | 1.89     | 0.00     | 0.55    |
|          | ARE                 | 1.96     | 2.42     | 1.90     | 0.00     | 0.94    |
|          | WRE                 | 1.65     | 2.07     | 1.45     | 0.00     | 1.63    |
|          | CPU TIME            | 123.73   | 121.78   | 123.80   | 122.62   | 121.63  |
| 60_240_8 | BRE                 | 1.52     | 1.39     | 1.43     | 0.00     | 0.64    |
|          | ARE                 | 1.39     | 1.91     | 1.61     | 0.00     | 0.66    |
|          | WRE                 | 0.91     | 1.21     | 1.08     | 0.00     | 0.80    |
|          | CPU TIME            | 122.77   | 121.73   | 121.35   | 121.35   | 123.05  |
| 60_240_9 | BRE                 | 2.88     | 3.37     | 3.04     | 0.89     | 0.00    |
|          | ARE                 | 1.51     | 1.77     | 1.42     | 0.00     | 0.51    |
|          | WRE                 | 1.57     | 1.66     | 1.18     | 0.00     | 1.35    |
|          | CPU TIME            | 122.20   | 123.36   | 122.74   | 121.99   | 122.22  |
| 60_240_10| BRE                | 2.12     | 2.67     | 1.78     | 0.26     | 0.00    |
|          | ARE                 | 1.91     | 2.24     | 2.26     | 0.00     | 1.11    |
|          | WRE                 | 1.73     | 2.07     | 2.22     | 0.00     | 1.37    |
|          | CPU TIME            | 122.97   | 123.66   | 123.48   | 122.98   | 123.09  |
Table G.8
Performance of SSMAs on instances with 300 tasks as the ratio of biochemical tests to immunologic tests is 1:1

| Instance | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|----------|
| 150_150_1| BRE                 | 1.12     | 1.55     | 1.77     | 0.00     | 0.24     |
|          | ARE                 | 1.90     | 2.20     | 2.18     | 0.00     | 0.83     |
|          | WRE                 | 2.74     | 1.85     | 2.92     | 0.00     | 2.12     |
|          | CPU TIME            | **124.23**| 125.22   | 125.56   | 125.77   | 125.36   |
| 150_150_2| BRE                 | 2.19     | 2.19     | 2.25     | 0.17     | **0.00** |
|          | ARE                 | 1.96     | 2.06     | 2.03     | **0.00** | 0.68     |
|          | WRE                 | 1.47     | 1.41     | 2.14     | **0.00** | 1.15     |
|          | CPU TIME            | 126.14   | 125.23   | 125.09   | **124.67**| 126.16   |
| 150_150_3| BRE                 | 1.89     | 2.32     | 1.49     | 0.00     | 0.02     |
|          | ARE                 | 1.84     | 1.97     | 1.90     | **0.00** | 0.80     |
|          | WRE                 | 1.25     | 1.48     | 1.60     | **0.00** | 0.92     |
|          | CPU TIME            | 125.24   | **124.39**| 124.50   | 125.55   | 126.20   |
| 150_150_4| BRE                 | 2.33     | 2.41     | 2.67     | 0.48     | **0.00** |
|          | ARE                 | 1.80     | 1.98     | 2.00     | **0.00** | 0.51     |
|          | WRE                 | 1.92     | 2.45     | 2.08     | **0.00** | 1.37     |
|          | CPU TIME            | 125.59   | **124.56**| 125.68   | 125.31   | 125.52   |
| 150_150_5| BRE                 | 2.14     | 3.19     | 3.70     | 0.00     | 0.22     |
|          | ARE                 | 2.54     | 2.70     | 2.81     | **0.00** | 1.10     |
|          | WRE                 | 2.39     | 2.77     | 2.77     | **0.00** | 2.56     |
|          | CPU TIME            | 126.09   | **124.32**| 125.57   | 124.69   | 125.16   |
| 150_150_6| BRE                 | 1.08     | 1.14     | 1.21     | 0.00     | 0.01     |
|          | ARE                 | 1.25     | 1.37     | 1.26     | **0.00** | 0.44     |
|          | WRE                 | 1.43     | 1.52     | 1.47     | **0.00** | 0.84     |
|          | CPU TIME            | 126.09   | 124.91   | **124.75**| 125.20   | 126.01   |
| 150_150_7| BRE                 | 2.01     | 2.01     | 1.32     | 0.00     | 0.68     |
|          | ARE                 | 1.57     | 1.76     | 1.62     | **0.00** | 0.82     |
|          | WRE                 | 2.14     | 2.35     | 1.89     | **0.00** | 1.54     |
|          | CPU TIME            | 126.17   | **125.68**| 125.77   | 125.88   | 126.07   |
| 150_150_8| BRE                 | 1.56     | 0.79     | 0.51     | 0.00     | 0.22     |
|          | ARE                 | 2.11     | 2.16     | 2.04     | **0.00** | 1.02     |
|          | WRE                 | 1.79     | 2.34     | 1.87     | **0.00** | 1.82     |
|          | CPU TIME            | 125.44   | **124.49**| 125.73   | 126.55   | 125.18   |
| 150_150_9| BRE                 | 1.34     | 1.55     | 1.95     | 0.01     | **0.00** |
|          | ARE                 | 1.77     | 2.16     | 2.03     | **0.00** | 1.06     |
|          | WRE                 | 0.41     | 0.95     | 1.01     | **0.00** | 1.05     |
|          | CPU TIME            | 125.42   | **125.31**| 126.42   | 126.18   | 126.80   |
| 150_150_10| BRE                | 2.40     | 2.15     | 2.46     | 0.29     | **0.00** |
|          | ARE                 | 2.42     | 2.87     | 2.39     | **0.00** | 0.67     |
|          | WRE                 | 2.15     | 3.37     | 2.17     | **0.00** | 1.91     |
|          | CPU TIME            | 125.10   | 125.10   | **124.87**| 124.99   | 126.54   |
### Table G.9
Performance of SSMAs on instances with 300 tasks as the ratio of biochemical tests to immunologic tests is 4:1

| Instance | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|---------|
| 240_60_1 | BRE                 | 1.12     | 1.12     | 1.12     | 0.02     | 0.00    |
|          | ARE                 | 0.50     | 0.49     | 0.49     | 0.00     | 0.26    |
|          | WRE                 | 0.12     | 0.18     | 0.27     | 0.12     | 0.00    |
|          | CPU TIME            | **121.62** | 122.10   | 122.86   | 122.69   | 124.00  |
| 240_60_2 | BRE                 | 1.28     | 1.17     | 0.49     | 0.12     | 0.00    |
|          | ARE                 | 0.80     | 0.79     | 0.64     | 0.00     | 0.33    |
|          | WRE                 | 0.40     | 0.39     | 0.47     | 0.00     | 0.24    |
|          | CPU TIME            | **122.11** | 123.17   | 123.71   | 122.78   | 123.15  |
| 240_60_3 | BRE                 | 0.83     | 0.83     | 0.64     | 0.03     | 0.00    |
|          | ARE                 | 0.32     | 0.37     | 0.34     | 0.00     | 0.23    |
|          | WRE                 | 0.19     | 0.27     | 0.14     | 0.00     | 0.13    |
|          | CPU TIME            | **121.63** | 121.73   | 123.72   | 122.45   | 122.74  |
| 240_60_4 | BRE                 | 0.89     | 0.89     | 0.69     | 0.00     | 0.09    |
|          | ARE                 | 0.43     | 0.39     | 0.44     | 0.00     | 0.19    |
|          | WRE                 | 0.26     | 0.30     | 0.10     | 0.00     | 0.10    |
|          | CPU TIME            | **121.97** | 122.84   | 122.77   | 122.99   | 123.03  |
| 240_60_5 | BRE                 | 0.90     | 0.90     | 0.29     | 0.00     | 0.24    |
|          | ARE                 | 0.26     | 0.30     | 0.20     | 0.00     | 0.15    |
|          | WRE                 | 0.16     | 0.31     | 0.31     | 0.00     | 0.31    |
|          | CPU TIME            | **123.51** | 122.43   | **122.24** | 122.38   | 123.59  |
| 240_60_6 | BRE                 | 1.93     | 1.64     | 1.68     | 0.00     | 0.25    |
|          | ARE                 | 0.99     | 0.97     | 0.93     | 0.00     | 0.33    |
|          | WRE                 | 0.89     | 0.81     | 0.69     | 0.00     | 0.69    |
|          | CPU TIME            | **121.54** | 122.78   | 121.56   | 122.55   | 123.20  |
| 240_60_7 | BRE                 | 1.09     | 1.09     | 0.71     | 0.03     | 0.00    |
|          | ARE                 | 0.48     | 0.48     | 0.38     | 0.00     | 0.29    |
|          | WRE                 | 0.56     | 0.59     | 0.44     | 0.00     | 0.59    |
|          | CPU TIME            | **121.70** | 122.71   | 122.47   | 121.78   | 123.39  |
| 240_60_8 | BRE                 | 1.01     | 1.01     | 0.86     | 0.09     | 0.00    |
|          | ARE                 | 0.50     | 0.49     | 0.59     | 0.00     | 0.31    |
|          | WRE                 | 0.44     | 0.80     | 0.63     | 0.00     | 0.44    |
|          | CPU TIME            | 122.01   | **121.48** | 122.25   | 122.03   | 123.61  |
| 240_60_9 | BRE                 | 1.37     | 1.37     | 1.19     | 0.04     | 0.00    |
|          | ARE                 | 0.63     | 0.61     | 0.59     | 0.00     | 0.16    |
|          | WRE                 | 0.22     | 0.30     | 0.24     | 0.00     | 0.21    |
|          | CPU TIME            | 122.61   | 122.41   | 121.98   | **121.92** | 122.37  |
| 240_60_10| BRE                 | 0.83     | 0.83     | 0.83     | 0.00     | 0.32    |
|          | ARE                 | 0.52     | 0.50     | 0.46     | 0.00     | 0.38    |
|          | WRE                 | 0.22     | 0.22     | 0.22     | 0.00     | 0.21    |
|          | CPU TIME            | 122.83   | 122.49   | **122.26** | 122.78   | 122.45  |
| Instance   | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|------------|---------------------|----------|----------|----------|----------|---------|
| 80_320_1   | BRE                 | 2.47     | 3.55     | 2.85     | 0.20     | 0.00    |
|            | ARE                 | 2.43     | 3.98     | 2.63     | 0.00     | 1.17    |
|            | WRE                 | 2.69     | 3.78     | 2.54     | 0.00     | 1.96    |
|            | CPU TIME            | 242.24   | 242.19   | 241.43   | 240.63   | 240.87  |
| 80_320_2   | BRE                 | 2.06     | 3.65     | 2.80     | 0.00     | 0.62    |
|            | ARE                 | 3.58     | 4.57     | 3.37     | 0.00     | 1.63    |
|            | WRE                 | 4.47     | 4.59     | 3.52     | 0.00     | 3.10    |
|            | CPU TIME            | 240.91   | 240.76   | 242.66   | 240.81   | 242.63  |
| 80_320_3   | BRE                 | 3.99     | 5.40     | 4.23     | 0.00     | 0.37    |
|            | ARE                 | 4.56     | 5.58     | 3.91     | 0.00     | 1.50    |
|            | WRE                 | 3.99     | 4.82     | 3.25     | 0.00     | 3.48    |
|            | CPU TIME            | 242.12   | 242.56   | 242.30   | 243.68   | 244.06  |
| 80_320_4   | BRE                 | 4.05     | 5.30     | 4.28     | 0.00     | 1.14    |
|            | ARE                 | 3.31     | 5.07     | 2.97     | 0.00     | 1.11    |
|            | WRE                 | 3.20     | 5.24     | 2.63     | 0.00     | 3.92    |
|            | CPU TIME            | 241.57   | 239.85   | 242.11   | 241.70   | 243.97  |
| 80_320_5   | BRE                 | 2.67     | 2.95     | 3.00     | 0.00     | 0.10    |
|            | ARE                 | 3.04     | 4.46     | 2.80     | 0.00     | 0.94    |
|            | WRE                 | 2.70     | 4.25     | 2.56     | 0.00     | 2.95    |
|            | CPU TIME            | 240.36   | 241.79   | 244.04   | 241.72   | 241.28  |
| 80_320_6   | BRE                 | 3.60     | 4.77     | 2.73     | 0.00     | 0.41    |
|            | ARE                 | 2.62     | 4.17     | 2.65     | 0.00     | 0.98    |
|            | WRE                 | 2.39     | 3.67     | 2.49     | 0.00     | 2.40    |
|            | CPU TIME            | 242.56   | 242.83   | 242.39   | 242.46   | 242.40  |
| 80_320_7   | BRE                 | 4.31     | 5.60     | 2.37     | 1.19     | 0.00    |
|            | ARE                 | 3.26     | 4.74     | 2.33     | 0.00     | 1.17    |
|            | WRE                 | 3.48     | 4.86     | 2.47     | 0.00     | 2.48    |
|            | CPU TIME            | 242.21   | 240.73   | 242.20   | 242.08   | 243.47  |
| 80_320_8   | BRE                 | 2.27     | 4.20     | 2.33     | 0.00     | 0.64    |
|            | ARE                 | 3.23     | 4.51     | 2.53     | 0.00     | 0.94    |
|            | WRE                 | 3.96     | 4.33     | 2.58     | 0.00     | 1.68    |
|            | CPU TIME            | 241.54   | 241.26   | 242.40   | 241.37   | 241.59  |
| 80_320_9   | BRE                 | 3.59     | 4.24     | 2.76     | 0.01     | 0.00    |
|            | ARE                 | 3.23     | 4.41     | 2.39     | 0.00     | 1.19    |
|            | WRE                 | 3.20     | 4.00     | 1.70     | 0.00     | 2.46    |
|            | CPU TIME            | 240.59   | 242.01   | 241.06   | 241.25   | 244.08  |
| 80_320_10  | BRE                 | 3.54     | 4.76     | 3.55     | 0.00     | 0.23    |
|            | ARE                 | 3.24     | 4.91     | 3.20     | 0.00     | 1.18    |
|            | WRE                 | 3.12     | 4.53     | 2.41     | 0.00     | 2.31    |
|            | CPU TIME            | 241.34   | 242.66   | 244.04   | 240.57   | 242.30  |
| Instance | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|----------|
| 200_200_1 | BRE                 | 2.29     | 3.35     | 2.50     | 0.00     | 1.24     |
|          | ARE                 | 1.69     | 2.39     | 2.02     | 0.00     | 1.04     |
|          | WRE                 | 1.67     | 1.74     | 1.74     | 0.00     | 1.74     |
|          | CPU TIME            | 241.23   | 240.90   | 242.95   | 240.41   | 242.01   |
| 200_200_2 | BRE                 | 2.40     | 2.40     | 1.86     | 0.62     | 0.00     |
|          | ARE                 | 1.17     | 1.17     | 1.13     | 0.00     | 0.31     |
|          | WRE                 | 0.53     | 0.53     | 0.53     | 0.00     | 0.53     |
|          | CPU TIME            | 239.98   | 242.43   | 241.92   | 241.85   | 243.37   |
| 200_200_3 | BRE                 | 1.94     | 2.91     | 2.82     | 0.71     | 0.00     |
|          | ARE                 | 1.43     | 2.02     | 1.84     | 0.00     | 0.42     |
|          | WRE                 | 1.44     | 1.47     | 1.50     | 0.00     | 1.07     |
|          | CPU TIME            | 241.85   | 240.99   | 241.26   | 241.76   | 241.25   |
| 200_200_4 | BRE                 | 2.13     | 3.21     | 2.43     | 0.00     | 0.30     |
|          | ARE                 | 1.62     | 1.98     | 1.92     | 0.00     | 0.84     |
|          | WRE                 | 1.27     | 1.28     | 1.28     | 0.00     | 1.27     |
|          | CPU TIME            | 242.69   | 240.97   | 240.74   | 241.57   | 244.68   |
| 200_200_5 | BRE                 | 2.01     | 3.12     | 2.56     | 0.00     | 0.29     |
|          | ARE                 | 1.72     | 2.04     | 1.97     | 0.00     | 1.00     |
|          | WRE                 | 1.34     | 1.34     | 1.34     | 0.00     | 1.34     |
|          | CPU TIME            | 243.12   | 241.85   | 241.01   | 240.58   | 242.11   |
| 200_200_6 | BRE                 | 2.10     | 3.17     | 2.37     | 0.00     | 1.03     |
|          | ARE                 | 1.57     | 2.02     | 1.77     | 0.00     | 0.70     |
|          | WRE                 | 1.63     | 1.44     | 1.46     | 0.00     | 1.41     |
|          | CPU TIME            | 240.39   | 242.10   | 241.67   | 242.37   | 243.31   |
| 200_200_7 | BRE                 | 1.80     | 2.08     | 2.08     | 0.00     | 1.14     |
|          | ARE                 | 0.60     | 0.61     | 0.61     | 0.00     | 0.38     |
|          | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|          | CPU TIME            | 241.71   | 241.82   | 240.94   | 241.85   | 242.64   |
| 200_200_8 | BRE                 | 1.70     | 2.43     | 1.65     | 0.26     | 0.00     |
|          | ARE                 | 1.44     | 1.83     | 1.80     | 0.00     | 0.33     |
|          | WRE                 | 1.13     | 1.47     | 1.79     | 0.00     | 1.14     |
|          | CPU TIME            | 241.59   | 241.87   | 241.71   | 242.26   | 242.85   |
| 200_200_9 | BRE                 | 1.78     | 1.89     | 1.81     | 0.00     | 0.16     |
|          | ARE                 | 1.73     | 2.23     | 2.14     | 0.00     | 1.24     |
|          | WRE                 | 1.39     | 1.59     | 1.59     | 0.00     | 1.26     |
|          | CPU TIME            | 241.80   | 241.07   | 240.29   | 241.63   | 242.25   |
| 200_200_10| BRE                | 1.43     | 3.51     | 2.07     | 0.31     | 0.00     |
|           | ARE                 | 1.74     | 2.46     | 2.26     | 0.00     | 0.66     |
|           | WRE                 | 0.87     | 0.87     | 0.87     | 0.00     | 0.87     |
|           | CPU TIME            | 240.05   | 240.12   | 241.56   | 241.35   | 242.58   |
| Instance | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|---------|
| 320_80_1 | BRE                 | 1.48     | 1.60     | 1.47     | 0.64     | 0.00    |
|          | ARE                 | 0.42     | 0.42     | 0.42     | 0.00     | 0.19    |
|          | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00    |
|          | CPU TIME            | 242.47   | 242.79   | **241.52** | 243.01   | 244.12  |
| 320_80_2 | BRE                 | 1.98     | 1.98     | 1.73     | 0.00     | 0.16    |
|          | ARE                 | 0.71     | 0.71     | 0.68     | 0.00     | 0.23    |
|          | WRE                 | 0.07     | 0.07     | 0.07     | 0.00     | 0.07    |
|          | CPU TIME            | **241.50** | 241.83   | 241.80   | 241.55   | 243.02  |
| 320_80_3 | BRE                 | 1.48     | 1.48     | 1.48     | 0.00     | 0.41    |
|          | ARE                 | 0.86     | 0.89     | 0.88     | 0.00     | 0.57    |
|          | WRE                 | 0.48     | 0.48     | 0.48     | 0.00     | 0.48    |
|          | CPU TIME            | **240.66** | 240.81   | 242.29   | 243.19   | 240.74  |
| 320_80_4 | BRE                 | 1.54     | 1.54     | 1.54     | 0.00     | 0.41    |
|          | ARE                 | 0.62     | 0.62     | 0.62     | 0.00     | 0.54    |
|          | WRE                 | 0.00     | **0.00** | 0.00     | 0.00     | 0.00    |
|          | CPU TIME            | **240.12** | 240.88   | 240.96   | 242.03   | 243.69  |
| 320_80_5 | BRE                 | 1.74     | 1.51     | 1.62     | 0.00     | 0.30    |
|          | ARE                 | 0.87     | 0.86     | 0.85     | 0.00     | 0.48    |
|          | WRE                 | 0.07     | 0.07     | 0.07     | 0.00     | 0.07    |
|          | CPU TIME            | 246.24   | 242.84   | **241.10** | 243.26   | 241.83  |
| 320_80_6 | BRE                 | 1.44     | 1.36     | 1.00     | 0.01     | 0.00    |
|          | ARE                 | 0.82     | 0.82     | 0.77     | 0.00     | 0.47    |
|          | WRE                 | 0.00     | **0.00** | 0.00     | 0.00     | 0.00    |
|          | CPU TIME            | **241.75** | 242.17   | 243.05   | **241.54** | 243.63  |
| 320_80_7 | BRE                 | 1.84     | 0.88     | 0.00     | 0.38     | 0.66    |
|          | ARE                 | 0.81     | 0.68     | 0.51     | 0.00     | 0.14    |
|          | WRE                 | 0.17     | 0.17     | 0.17     | 0.00     | 0.17    |
|          | CPU TIME            | 241.03   | 242.92   | **240.92** | 240.99   | 243.47  |
| 320_80_8 | BRE                 | 1.06     | 1.11     | 0.63     | 0.02     | 0.00    |
|          | ARE                 | 0.33     | 0.34     | 0.15     | 0.00     | 0.14    |
|          | WRE                 | 0.00     | **0.00** | 0.00     | 0.00     | 0.00    |
|          | CPU TIME            | **241.63** | 241.65   | 242.02   | 241.66   | 242.39  |
| 320_80_9 | BRE                 | 0.47     | 0.47     | 0.33     | 0.00     | 0.16    |
|          | ARE                 | 0.08     | 0.08     | 0.07     | 0.00     | 0.06    |
|          | WRE                 | 0.00     | **0.00** | 0.00     | 0.00     | 0.00    |
|          | CPU TIME            | 241.41   | **240.37** | 240.91   | 241.84   | 243.89  |
| 320_80_10| BRE                | 2.41     | 2.53     | 1.44     | 0.00     | 0.30    |
|           | ARE                | 1.67     | 1.68     | 1.61     | 0.00     | 0.97    |
|           | WRE                | 0.55     | 0.55     | 0.55     | 0.00     | 0.55    |
|           | CPU TIME           | 241.10   | **240.88** | 240.93   | 241.37   | 242.70  |
| Instance  | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|----------|---------------------|----------|----------|----------|----------|---------|
| 100_400_1 | BRE                 | 2.03     | 3.34     | 2.08     | 0.00     | 0.12    |
|          | ARE                 | 1.72     | 3.03     | 2.39     | 0.00     | 0.88    |
|          | WRE                 | 1.56     | 2.11     | 2.09     | 0.00     | 1.99    |
|          | CPU TIME            | 525.14   | 523.20   | 522.19   | 521.67   | 523.53  |
| 100_400_2 | BRE                 | 3.30     | 4.53     | 3.48     | 0.00     | 0.31    |
|          | ARE                 | 3.05     | 4.11     | 3.24     | 0.00     | 0.86    |
|          | WRE                 | 2.10     | 2.14     | 2.14     | 0.00     | 1.17    |
|          | CPU TIME            | 522.62   | 519.71   | 521.35   | 522.17   | 523.04  |
| 100_400_3 | BRE                 | 2.32     | 3.92     | 2.99     | 0.00     | 0.96    |
|          | ARE                 | 3.01     | 4.18     | 3.46     | 0.00     | 1.39    |
|          | WRE                 | 2.52     | 3.00     | 2.90     | 0.00     | 2.18    |
|          | CPU TIME            | 524.49   | 519.44   | 521.60   | 521.73   | 523.69  |
| 100_400_4 | BRE                 | 2.99     | 4.15     | 3.26     | 0.00     | 0.38    |
|          | ARE                 | 2.04     | 2.98     | 2.64     | 0.00     | 0.86    |
|          | WRE                 | 1.38     | 1.62     | 1.62     | 0.00     | 1.51    |
|          | CPU TIME            | 520.67   | 523.14   | 521.86   | 522.03   | 524.27  |
| 100_400_5 | BRE                 | 3.41     | 4.67     | 3.09     | 0.00     | 1.00    |
|          | ARE                 | 2.76     | 3.60     | 3.04     | 0.00     | 1.16    |
|          | WRE                 | 2.61     | 2.77     | 2.76     | 0.00     | 2.24    |
|          | CPU TIME            | 524.96   | 521.14   | 521.21   | 524.51   | 524.07  |
| 100_400_6 | BRE                 | 2.62     | 4.42     | 3.47     | 0.19     | 0.00    |
|          | ARE                 | 2.42     | 3.14     | 2.85     | 0.00     | 0.90    |
|          | WRE                 | 1.84     | 1.84     | 1.84     | 0.00     | 1.84    |
|          | CPU TIME            | 524.18   | 522.34   | 521.42   | 522.30   | 524.07  |
| 100_400_7 | BRE                 | 2.64     | 4.11     | 3.12     | 0.25     | 0.00    |
|          | ARE                 | 2.48     | 3.09     | 2.84     | 0.00     | 0.89    |
|          | WRE                 | 2.11     | 2.11     | 2.11     | 0.00     | 2.11    |
|          | CPU TIME            | 522.22   | 521.23   | 521.85   | 522.45   | 526.10  |
| 100_400_8 | BRE                 | 2.53     | 4.40     | 2.27     | 0.00     | 0.23    |
|          | ARE                 | 2.41     | 3.25     | 2.73     | 0.00     | 0.76    |
|          | WRE                 | 1.53     | 1.80     | 1.80     | 0.00     | 1.76    |
|          | CPU TIME            | 521.99   | 521.32   | 521.73   | 521.58   | 526.60  |
| 100_400_9 | BRE                 | 2.71     | 4.80     | 2.71     | 0.00     | 0.11    |
|          | ARE                 | 2.43     | 3.42     | 2.92     | 0.00     | 0.74    |
|          | WRE                 | 1.99     | 2.00     | 2.00     | 0.00     | 1.68    |
|          | CPU TIME            | 521.65   | 520.35   | 522.59   | 522.69   | 523.20  |
| 100_400_10 | BRE                | 3.06     | 4.03     | 3.27     | 1.29     | 0.00    |
|           | ARE                 | 2.05     | 2.79     | 2.67     | 0.00     | 0.42    |
|           | WRE                 | 2.03     | 2.18     | 2.18     | 0.00     | 1.43    |
|           | CPU TIME            | 522.56   | 519.81   | 519.24   | 521.01   | 524.96  |
## Table G.14

Performance of SSMAs on instances with 500 tasks as the ratio of biochemical tests to immunologic tests is 1:1

| Instance     | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|--------------|---------------------|----------|----------|----------|----------|----------|
| 250_250_1    | BRE                 | 0.03     | 0.00     | 0.00     | 0.03     | 0.02     |
|              | ARE                 | 0.17     | 0.00     | 0.02     | 0.00     | 0.11     |
|              | WRE                 | 0.38     | 0.75     | **0.00** | 0.03     | 1.01     |
|              | CPU TIME            | 530.35   | 530.76   | 531.36   | **529.62** | 532.28   |
| 250_250_2    | BRE                 | 0.46     | 0.38     | 0.47     | **0.00** | 0.20     |
|              | ARE                 | 0.45     | 0.24     | 0.34     | **0.00** | 0.41     |
|              | WRE                 | 0.48     | 0.44     | 0.43     | **0.00** | 0.23     |
|              | CPU TIME            | **529.51** | 529.67   | 530.54   | 531.46   | 531.01   |
| 250_250_3    | BRE                 | **0.00** | 0.13     | 0.09     | 0.07     | 0.08     |
|              | ARE                 | 0.45     | **0.00** | 0.31     | 0.09     | 0.18     |
|              | WRE                 | 0.20     | 0.58     | 0.45     | 0.16     | **0.00** |
|              | CPU TIME            | 530.82   | 529.89   | 529.63   | 530.39   | 531.42   |
| 250_250_4    | BRE                 | 0.06     | 0.02     | 0.13     | **0.00** | **0.00** |
|              | ARE                 | 0.15     | 0.05     | 0.17     | **0.00** | 0.05     |
|              | WRE                 | 0.53     | 0.68     | 0.55     | **0.00** | 0.64     |
|              | CPU TIME            | 532.91   | **531.14** | 531.85   | 531.79   | 531.28   |
| 250_250_5    | BRE                 | 0.18     | 0.34     | 0.11     | 0.55     | **0.00** |
|              | ARE                 | 0.27     | 0.35     | 0.42     | **0.00** | 0.10     |
|              | WRE                 | 0.48     | 0.60     | 0.51     | **0.00** | 0.38     |
|              | CPU TIME            | 531.50   | 530.89   | 530.39   | **530.18** | 532.79   |
| 250_250_6    | BRE                 | 0.19     | **0.00** | 0.01     | 0.28     | 0.16     |
|              | ARE                 | 0.06     | **0.00** | 0.08     | 0.03     | 0.12     |
|              | WRE                 | 0.59     | 0.72     | 0.25     | **0.00** | 0.28     |
|              | CPU TIME            | **528.83** | 530.15   | 530.83   | 533.57   | 532.87   |
| 250_250_7    | BRE                 | 0.37     | 0.96     | 1.03     | **0.00** | 0.95     |
|              | ARE                 | 0.05     | 0.14     | 0.19     | **0.00** | 0.20     |
|              | WRE                 | 0.17     | **0.00** | 0.22     | 0.22     | 0.20     |
|              | CPU TIME            | 534.65   | 528.47   | 531.21   | **527.41** | 533.37   |
| 250_250_8    | BRE                 | 0.46     | 0.39     | 0.39     | **0.00** | 0.32     |
|              | ARE                 | 0.16     | 0.08     | 0.15     | 0.03     | **0.00** |
|              | WRE                 | 0.34     | 0.12     | 0.47     | **0.00** | 0.19     |
|              | CPU TIME            | 530.28   | 534.83   | **530.01** | 532.50   | 534.06   |
| 250_250_9    | BRE                 | 0.21     | **0.00** | 0.02     | 0.37     | 0.23     |
|              | ARE                 | 0.10     | 0.04     | 0.17     | **0.00** | 0.05     |
|              | WRE                 | 0.24     | 0.62     | 0.80     | **0.00** | 0.36     |
|              | CPU TIME            | 530.68   | **527.95** | 531.01   | 529.88   | 533.42   |
| 250_250_10   | BRE                 | 0.03     | 0.16     | **0.00** | 0.07     | 0.11     |
|              | ARE                 | 0.03     | **0.00** | 0.01     | 0.24     | 0.07     |
|              | WRE                 | 0.02     | **0.00** | 0.11     | 0.21     | 0.37     |
|              | CPU TIME            | 530.27   | 529.36   | 530.02   | **527.67** | 533.51   |


| Instance  | Performance metrics | SSMA-INV | SSMA-INS | SSMA-SWP | SSMA-INB | SSMA-ML |
|-----------|---------------------|----------|----------|----------|----------|----------|
| 400_100_1 | BRE                 | 1.79     | 1.88     | 1.78     | 0.41     | 0.00     |
|           | ARE                 | 0.58     | 0.58     | 0.57     | 0.00     | 0.06     |
|           | WRE                 | 0.05     | 0.05     | 0.05     | 0.00     | 0.05     |
|           | CPU TIME            | 519.54   | 519.86   | 523.41   | 320.30   | 317.17   |
| 400_100_2 | BRE                 | 1.00     | 1.00     | 1.00     | 0.00     | 0.28     |
|           | ARE                 | 0.22     | 0.22     | 0.22     | 0.00     | 0.11     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 521.49   | 516.88   | 520.53   | 519.16   | 516.64   |
| 400_100_3 | BRE                 | 0.71     | 0.71     | 0.71     | 0.18     | 0.00     |
|           | ARE                 | 0.13     | 0.13     | 0.12     | 0.09     | 0.00     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 517.21   | 518.79   | 519.97   | 520.36   | 525.18   |
| 400_100_4 | BRE                 | 1.81     | 1.81     | 1.81     | 0.00     | 1.31     |
|           | ARE                 | 0.32     | 0.32     | 0.32     | 0.00     | 0.30     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 519.30   | 522.41   | 519.84   | 521.47   | 523.36   |
| 400_100_5 | BRE                 | 2.66     | 2.66     | 2.66     | 0.37     | 0.00     |
|           | ARE                 | 0.55     | 0.55     | 0.55     | 0.00     | 0.16     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 519.53   | 521.42   | 522.89   | 523.51   | 525.34   |
| 400_100_6 | BRE                 | 1.36     | 1.36     | 1.36     | 0.00     | 0.66     |
|           | ARE                 | 0.57     | 0.56     | 0.57     | 0.00     | 0.35     |
|           | WRE                 | 0.13     | 0.13     | 0.13     | 0.00     | 0.13     |
|           | CPU TIME            | 522.48   | 523.77   | 521.17   | 520.58   | 522.62   |
| 400_100_7 | BRE                 | 1.23     | 1.24     | 1.24     | 0.16     | 0.00     |
|           | ARE                 | 0.42     | 0.42     | 0.42     | 0.00     | 0.12     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 518.64   | 521.90   | 520.72   | 520.96   | 524.49   |
| 400_100_8 | BRE                 | 1.41     | 1.40     | 1.40     | 0.00     | 0.60     |
|           | ARE                 | 0.36     | 0.36     | 0.36     | 0.00     | 0.29     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 523.12   | 525.70   | 521.08   | 519.76   | 522.85   |
| 400_100_9 | BRE                 | 0.97     | 0.97     | 0.97     | 0.00     | 0.06     |
|           | ARE                 | 0.24     | 0.24     | 0.24     | 0.00     | 0.00     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 520.57   | 520.25   | 521.44   | 522.16   | 521.87   |
| 400_100_10| BRE                 | 0.68     | 0.68     | 0.68     | 0.00     | 0.68     |
|           | ARE                 | 0.10     | 0.10     | 0.10     | 0.00     | 0.10     |
|           | WRE                 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     |
|           | CPU TIME            | 524.34   | 521.34   | 524.74   | 521.59   | 523.77   |
Appendix H: Aggregated performances of SSMAs on medium- and large-size instances

Table H.1
SSMA performance on 30 medium-size instances and 120 large-size instances

| Instance_20_80_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|------------------------|-----------|------|------|------|----------|
| SSMA-INV               | 1.95      | 1.54 | 0.73 | 19.45|
| SSMA-INS               | 1.05      | 1.05 | 0.75 | 19.39|
| SSMA-SWP               | 0.00      | 0.00 | 0.00 | 19.03|
| SSMA-INB               | 2.89      | 1.67 | 0.43 | 19.06|
| SSMA-ML                | 0.70      | 1.07 | 0.50 | 19.15|

| Instance_50_50_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|------------------------|-----------|------|------|------|----------|
| SSMA-INV               | 1.49      | 1.58 | 1.42 | 19.76|
| SSMA-INS               | 0.65      | 0.78 | 1.09 | 19.87|
| SSMA-SWP               | 0.00      | 0.00 | 0.00 | 19.77|
| SSMA-INB               | 0.99      | 0.94 | 0.88 | 19.89|
| SSMA-ML                | 0.14      | 0.40 | 0.86 | 19.93|

| Instance_80_20_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|------------------------|-----------|------|------|------|----------|
| SSMA-INV               | 0.39      | 0.37 | 0.65 | 18.97|
| SSMA-INS               | 0.39      | 0.24 | 0.52 | 19.02|
| SSMA-SWP               | 0.32      | 0.14 | 0.39 | 19.32|
| SSMA-INB               | 0.53      | 0.04 | 0.00 | 18.94|
| SSMA-ML                | 0.00      | 0.00 | 0.00 | 19.06|

| Instance_40_160_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|-------------------------|-----------|------|------|------|----------|
| SSMA-INV                | 1.44      | 1.85 | 2.10 | 56.26|
| SSMA-INS                | 1.17      | 2.00 | 2.52 | 56.35|
| SSMA-SWP                | 0.00      | 1.10 | 2.10 | 56.22|
| SSMA-INB                | 0.25      | 0.00 | 0.00 | 56.42|
| SSMA-ML                 | 0.45      | 0.81 | 1.85 | 52.82|

| Instance_100_100_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|--------------------------|-----------|------|------|------|----------|
| SSMA-INV                 | 1.53      | 1.40 | 1.33 | 56.45|
| SSMA-INS                 | 1.68      | 1.47 | 1.36 | 56.30|
| SSMA-SWP                 | 1.23      | 1.26 | 1.16 | 56.17|
| SSMA-INB                 | 0.00      | 0.00 | 0.00 | 56.46|
| SSMA-ML                  | 0.16      | 0.73 | 1.18 | 52.64|

| Instance_160_40_{1-10}  | Algorithm | BRE  | ARE  | WRE  | CPU time |
|--------------------------|-----------|------|------|------|----------|
| SSMA-INV                 | 0.96      | 0.72 | 0.71 | 56.23|
| SSMA-INS                 | 0.86      | 0.73 | 0.79 | 56.16|
| SSMA-SWP                 | 0.57      | 0.54 | 0.67 | 56.39|
| SSMA-INB                 | 0.00      | 0.00 | 0.00 | 56.35|
| SSMA-ML                  | 0.11      | 0.40 | 0.62 | 52.61|

| Instance_200_200_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|--------------------------|-----------|------|------|------|----------|
| SSMA-INV                 | 1.09      | 0.54 | 0.33 | 122.15|
| SSMA-INS                 | 1.05      | 0.54 | 0.41 | 122.41|
| SSMA-SWP                 | 0.82      | 0.51 | 0.34 | 122.58|
| SSMA-INB                 | 0.00      | 0.00 | 0.00 | 122.43|
| SSMA-ML                  | 0.06      | 0.26 | 0.28 | 123.15|

| Instance_300_300_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|--------------------------|-----------|------|------|------|----------|
| SSMA-INV                 | 3.11      | 3.25 | 3.32 | 241.54|
| SSMA-INS                 | 4.29      | 4.64 | 4.41 | 241.66|
| SSMA-SWP                 | 2.94      | 2.88 | 2.61 | 242.46|
| SSMA-INB                 | 0.00      | 0.00 | 0.00 | 241.63|
| SSMA-ML                  | 0.21      | 1.18 | 2.67 | 242.66|

| Instance_400_400_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|--------------------------|-----------|------|------|------|----------|
| SSMA-INV                 | 1.76      | 1.47 | 1.13 | 241.44|
| SSMA-INS                 | 2.61      | 1.88 | 1.17 | 241.41|
| SSMA-SWP                 | 2.02      | 1.75 | 1.21 | 241.40|
| SSMA-INB                 | 0.00      | 0.00 | 0.00 | 241.56|
| SSMA-ML                  | 0.22      | 0.69 | 1.06 | 242.70|

| Instance_500_500_{1-10} | Algorithm | BRE  | ARE  | WRE  | CPU time |
|--------------------------|-----------|------|------|------|----------|
| SSMA-INV                 | 1.44      | 0.72 | 0.14 | 241.33|
| SSMA-INS                 | 1.34      | 0.71 | 0.14 | 241.81|
| SSMA-SWP                 | 1.02      | 0.66 | 0.14 | 241.55|
| SSMA-INB                 | 0.00      | 0.00 | 0.00 | 241.86|
| SSMA-ML                  | 0.13      | 0.38 | 0.14 | 242.95|
| INSTANCE_100_400_{1-10} | SSMA-INV | 2.58 | 2.44 | 1.97 | 523.05 |
|-------------------------|----------|------|------|------|--------|
|                         | SSMA-INS | 4.06 | 3.36 | 2.16 | 521.17 |
|                         | SSMA-SWP | 2.79 | 2.88 | 2.14 | 521.79 |
|                         | SSMA-INB | 0.00 | 0.00 | 0.00 | 522.21 |
|                         | SSMA-ML  | 0.14 | 0.88 | 1.79 | 524.35 |
| INSTANCE_250_250_{1-10} | SSMA-INV | 0.06 | 0.15 | 0.28 | 530.98 |
|                         | SSMA-INS | 0.10 | 0.05 | 0.39 | 530.31 |
|                         | SSMA-SWP | 0.09 | 0.15 | 0.32 | 530.69 |
|                         | SSMA-INB | 0.00 | 0.00 | 0.00 | 530.45 |
|                         | SSMA-ML  | 0.07 | 0.09 | 0.31 | 532.60 |
| INSTANCE_400_100_{1-10} | SSMA-INV | 1.25 | 0.34 | 0.02 | 521.12 |
|                         | SSMA-INS | 1.26 | 0.34 | 0.02 | 521.43 |
|                         | SSMA-SWP | 1.25 | 0.34 | 0.02 | 521.58 |
|                         | SSMA-INB | 0.00 | 0.00 | 0.00 | 520.99 |
|                         | SSMA-ML  | 0.24 | 0.15 | 0.02 | 522.33 |
References

[1] L. Esposito, Hospital Labs: Behind the Scenes. <https://health.usnews.com/health-news/patient-advice/articles/2015/01/30/hospital-labs-behind-the-scenes> [accessed 03/09/2018], (2015).

[2] L.L. Holland, L.L. Smith, K.E. Blick, Total laboratory automation can help eliminate the laboratory as a factor in emergency department length of stay, American Journal of Clinical Pathology, 125(5) (2006) 765-770.

[3] A.J. Singer, J. Ardise, J. Gulla, J. Cangro, Point-of-care testing reduces length of stay in emergency department chest pain patients, Ann Emerg Med, 45(6) (2005) 587-591.

[4] P. Keskinocak, N. Savva, A Review of the Healthcare-Management (Modeling) Literature Published in Manufacturing & Service Operations Management, Manufacturing & Service Operations Management, 22(1) (2020) 59-72.

[5] R.J. Aarts, J.S. Lindsey, L.A. Corkan, S.F. Smith, Flexible Protocols Improve Parallel Experimentation Throughput, Clinical Chemistry, 41(7) (1995) 1004-1010.

[6] B.M. Elena, P. Beraldi, D. Conforti, Improving the efficiency of a clinical laboratory: a mathematical approach, IFAC Proceedings Volumes, 39(3) (2006) 659-664.

[7] T. Jones, S. Forrest, Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms, in: Proceedings Proceedings of the Sixth International Conference on Genetic Algorithms, 1995, pp. 184–192.

[8] E. Weinberger, Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference, Biological Cybernetics, 63(5) (1990) 325-336.

[9] G. Ochoa, N. Veerapen, Mapping the global structure of TSP fitness landscapes, Journal of Heuristics, 24(3) (2018) 265-294.

[10] Y.S. Ong, A.J. Keane, Meta-Lamarckian learning in memetic algorithms, IEEE Transactions on Evolutionary Computation, 8(2) (2004) 99-110.

[11] J. Davis, Scheduling in the Clinical Laboratory, Clinical Chemistry, 41(7) (1995) 961-962.

[12] J.W. Kim, B.J. Choi, K.H. Noh, H.R. Choi, J.C. Koo, S.M. Ryew, J.H. Kim, W.H. Son, Automatic scheduling algorithm for personalized clinical test, in: Proceedings 2006 SICE-ICASE International Joint Conference Busan, South Korea, 2006, pp. 5823-5827.

[13] G.G. Van Merode, M. Oosten, O.J. Vrieze, J. Derks, A. Hasman, Optimisation of the structure of the clinical laboratory, European Journal of Operational Research, 105(2) (1998) 308-316.

[14] J.C. Boyd, J. Savory, Genetic algorithm for scheduling of laboratory personnel, Clinical Chemistry, 47(1) (2001) 118-123.

[15] A. Toptal, I. Sabuncuooglu, Distributed scheduling: a review of concepts and applications, International Journal of Production Research, 48(18) (2010) 5235-5262.

[16] J. Behnamian, S.M.T.F. Ghomi, A survey of multi-factory scheduling, Journal of Intelligent Manufacturing, 27(1) (2016) 231-249.

[17] S. Hatami, R. Ruiz, C. Andres-Romano, The Distributed Assembly Permutation Flowshop Scheduling Problem, International Journal of Production Research, 51(17) (2013) 5292-5308.

[18] V. Fernandez-Viagas, J.M. Framinan, A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem, International Journal of Production Research, 53(4) (2015) 1111-1123.
[19] J.J. Wang, L. Wang, A Knowledge-Based Cooperative Algorithm for Energy-Efficient Scheduling of Distributed Flow-Shop, IEEE Transactions on Systems Man Cybernetics-Systems, 50(5) (2020) 1805-1819.

[20] J. Lohmer, R. Lasch, Production planning and scheduling in multi-factory production networks: a systematic literature review, International Journal of Production Research, (2020) 1-27.

[21] J.N. Hooker, Planning and scheduling by logic-based benders decomposition, Operations Research, 55(3) (2007) 588-602.

[22] M. Ziaee, A heuristic algorithm for the distributed and flexible job-shop scheduling problem, J. Supercomput., 67(1) (2014) 69-83.

[23] S.H. Chung, F.T.S. Chan, H.K. Chan, A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling, Engineering Applications of Artificial Intelligence, 22(7) (2009) 1005-1014.

[24] L. De Giovanni, F. Pezzella, An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem, European Journal of Operational Research, 200(2) (2010) 395-408.

[25] V. Roshanaei, C. Luong, D.M. Aleman, D.R. Urbach, Collaborative Operating Room Planning and Scheduling, INFORMS Journal on Computing, 29(3) (2017) 558-580.

[26] J. Gratch, S. Chien, Adaptive problem-solving for large-scale scheduling problems: A case study, Journal of Artificial Intelligence Research, 4 (1996) 365-396.

[27] D. Frost, R. Dechter, In Search of the Best Constraint Satisfaction Search, Proceedings of the Twelfth National Conference on Artificial Intelligence, Vols 1 and 2, (1994) 301-306.

[28] S. Kambhampati, C.A. Knoblock, Q. Yang, Planning as Refinement Search - a Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning, Artificial Intelligence, 76(1-2) (1995) 167-238.

[29] A. Barrett, D.S. Weld, Partial-order planning: Evaluating possible efficiency gains, Artificial Intelligence, 67(1) (1994) 71-112.

[30] S. Minton, Learning search control knowledge: an explanation-based approach, Kluwer Academic, Boston, 1988.

[31] G. Schmidt, Case-based reasoning for production scheduling, International Journal of Production Economics, 56-57 (1998) 537-546.

[32] Q.H. Nguyen, Y.S. Ong, M.H. Lim, A Probabilistic Memetic Framework, IEEE Transactions on Evolutionary Computation, 13(3) (2009) 604-623.

[33] Y. Bengio, A. Lodi, A. Prouvost, Machine learning for combinatorial optimization: A methodological tour d’horizon, European Journal of Operational Research, 290(2) (2021) 405-421.

[34] W. Kool, H. van Hoof, M. Welling, Attention, Learn to Solve Routing Problems!, in: Proceedings ICLR, 2019.

[35] J. Gratch, G. DeJong, A statistical approach to adaptive problem solving, Artificial Intelligence, 88(1) (1996) 101-142.

[36] J.H. Drake, A. Kheiri, E. Özcan, E.K. Burke, Recent advances in selection hyper-heuristics, European Journal of Operational Research, 285(2) (2020) 405-428.

[37] K.M. Malan, A.P. Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward, Information Sciences, 241 (2013) 148-163.
[38] C.R. Reeves, Landscapes, operators and heuristic search, Annals of Operations Research, 86 (1999) 473-490.

[39] C.M. Reidys, P.F. Stadler, Combinatorial landscapes, SIAM Review, 44(1) (2002) 3-54.

[40] F. Daolio, A. Liefooghe, S. Verel, H. Aguirre, K. Tanaka, Problem Features versus Algorithm Performance on Rugged Multiobjective Combinatorial Fitness Landscapes, Evolutionary Computation, 25(4) (2017) 555-585.

[41] F. Hutter, L. Xu, H.H. Hoos, K. Leyton-Brown, Algorithm runtime prediction: Methods & evaluation, Artificial Intelligence, 206 (2014) 79-111.

[42] J.P. Watson, L.D. Whitley, A.E. Howe, Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search, Journal of Artificial Intelligence Research, 24(1) (2005) 221-261.

[43] M.J. Streeter, S.F. Smith, How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines, Journal of Artificial Intelligence Research, 26 (2006) 247-287.

[44] P. Merz, K. Katayama, Memetic algorithms for the unconstrained binary quadratic programming problem, Biosystems, 78(1-3) (2004) 99-118.

[45] W. Zhang, Configuration landscape analysis and backbone guided local search.: Part I: Satisfiability and maximum satisfiability, Artificial Intelligence, 158(1) (2004) 1-26.

[46] L. Barbulescu, A.E. Howe, L.D. Whitley, M. Roberts, Understanding algorithm performance on an oversubscribed scheduling application, Journal of Artificial Intelligence Research, 27 (2006) 577-615.

[47] C.A. Mendez, J. Cerda, I.E. Grossmann, I. Harjunkoski, M. Fahl, State-of-the-art review of optimization methods for short-term scheduling of batch processes, Computers & Chemical Engineering, 30(6-7) (2006) 913-946.

[48] M.L. Pinedo, Scheduling: theory, algorithms, and systems, 4 ed., Springer, Boston, MA, 2012.

[49] F. Pezzella, G. Morganti, G. Ciaschetti, A genetic algorithm for the Flexible Job-shop Scheduling Problem, Computers & Operations Research, 35(10) (2008) 3202-3212.

[50] F. Glover, A template for scatter search and path relinking, in: J. Hao, E. Lutton, E. Ronald, M. Schoenauer, S. D. (Eds.) Artificial Evolution, Springer, Berlin, Heidelberg, 1998, pp. 1-51.

[51] E. Nowicki, C. Smutnicki, Some aspects of scatter search in the flow-shop problem, European Journal of Operational Research, 169(2) (2006) 654-666.

[52] B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop scheduling problem, European Journal of Operational Research, 239(2) (2014) 323-334.

[53] Y. Yang, P. Li, S. Wang, B. Liu, Y. Luo, Scatter search for distributed assembly flowshop scheduling to minimize total tardiness, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 861-868.

[54] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, 1(1) (1997) 67-82.

[55] W.A. Dembski, R.J. Marks, Conservation of Information in Search: Measuring the Cost of Success, IEEE Transactions on Systems Man and Cybernetics Part A-Systems and Humans, 39(5) (2009) 1051-1061.

[56] B. Liu, L. Wang, Y.H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 37(1) (2007) 18-27.
[57] M. Nawaz, E.E. Enscore, I. Ham, A Heuristic Algorithm for the M-Machine, N-Job Flowshop Sequencing Problem, Omega-International Journal of Management Science, 11(1) (1983) 91-95.

[58] B. Liu, L. Wang, Y. Liu, S.Y. Wang, A unified framework for population-based metaheuristics, Annals of Operations Research, 186(1) (2011) 231-262.

[59] I. Ribas, R. Company, X. Tort-Martorell, Comparing three-step heuristics for the permutation flow shop problem, Computers & Operations Research, 37(12) (2010) 2062-2070.

[60] B. Liu, J. Xu, B. Qian, J. Wang, Y. Chu, Probabilistic memetic algorithm for flow shop scheduling, in: 2013 IEEE Workshop on Memetic Computing (MC), 2013, pp. 60-64.

[61] F. Glover, Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms), Discrete Applied Mathematics, 49(1-3) (1994) 231-255.

[62] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated Annealing, Science, 220(4598) (1983) 671-680.

[63] R.C. Ball, J. Branke, S. Meisel, Optimal Sampling for Simulated Annealing Under Noise, INFORMS Journal on Computing, 30(1) (2018) 200-215.

[64] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of State Calculations by Fast Computing Machines, The Journal of Chemical Physics, 21(6) (1953) 1087-1092.

[65] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., 1989.

[66] J.P. Watson, J.C. Beck, A.E. Howe, L.D. Whitley, Problem difficulty for tabu search in job-shop scheduling, Artificial Intelligence, 143(2) (2003) 189-217.

[67] J.P. Watson, L. Barbulescu, L.D. Whitley, A.E. Howe, Contrasting structured and random permutation flow-shop scheduling problems: Search-space topology and algorithm performance, Informs Journal on Computing, 14(2) (2002) 98-123.

[68] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local Optima Networks of the Permutation Flow-Shop Problem, in: P. Legrand, M.-M. Corsini, J.-K. Hao, N. Monmarché, E. Lutton, M. Schoenauer (Eds.) Artificial Evolution: 11th International Conference, Evolution Artificielle, EA 2013, Bordeaux, France, October 21-23, 2013. Revised Selected Papers, Springer International Publishing, Cham, 2014, pp. 41-52.

[69] K. Wang, B. Liu, Benchmark for real-practice distributed scheduling in heterogeneous, flexible job shop. Technical Report, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. DOI: 10.13140/RG.2.2.18275.48164, (2021).

[70] Gurobi optimization. <http://www.gurobi.com> [accessed 12/20/2020].

[71] J. Lofberg, A toolbox for modeling and optimization in MATLAB, in: IEEE International Symposium on Computer Aided Control Systems Design, 2005.

[72] B. Naderi, A. Azab, Modeling and heuristics for scheduling of distributed job shops, Expert Systems with Applications, 41(17) (2014) 7754-7763.

[73] A.M. Mood, F.A. Graybill, D.C. Boes, Introduction to the theory of statistics, McGraw-Hill, New York, 1973.