Experimentation on Iterated Local Search Hyper-heuristics for Combinatorial Optimization Problems

Stephen A. Adubi[1], Olufunke O. Oladipupo[2], Oludayo O. Olugbara[3]
Computer and Information Sciences Covenant University, Ota 112104, Ogun State, Nigeria[1, 2]
MICT SETA 4IR Center of Excellence, Durban University of Technology, Durban 4001, South Africa[3]

Abstract*—Designing effective algorithms to solve cross-domain combinatorial optimization problems is an important goal for which manifold search methods have been extensively investigated. However, finding an optimal combination of perturbation operations for solving cross-domain optimization problems is hard because of the different characteristics of each problem and the discrepancies in the strengths of perturbation operations. The algorithm that works effectively for one problem domain may completely falter in the instances of other optimization problems. The objectives of this study are to describe three categories of a hyper-heuristic that combine low-level heuristics with an acceptance mechanism for solving cross-domain optimization problems, compare the three hyper-heuristic categories against the existing benchmark algorithms and experimentally determine the effects of low-level heuristic categorization on the standard optimization problems from the hyper-heuristic flexible framework. The hyper-heuristic categories are based on the methods of Thompson sampling and iterated local search to control the perturbation behavior of the iterated local search. The performances of the perturbation configurations in a hyper-heuristic were experimentally tested against the existing benchmark algorithms on standard optimization problems from the hyper-heuristic flexible framework. Study findings have suggested the most effective hyper-heuristic with improved performance when compared to the existing hyper-heuristics investigated for solving cross-domain optimization problems to be the one with a good balance between "single shaking" and "double shaking" strategies. The findings not only provide a foundation for establishing comparisons with other hyper-heuristics but also demonstrate a flexible alternative to investigate effective hyper-heuristics for solving complex combinatorial optimization problems.

***Keywords*—Combinatorial optimization; heuristic algorithm; heuristic categorization; local search; Thompson sampling**

I. INTRODUCTION

Combinatorial optimization problems (COPs) are practically challenging because of the different characteristics of each problem domain and multiple conflicting constrictions to be adequately resolved. They are intrinsically non-deterministic polynomial-time hard problems with no single method that can generally outperform others across varying problem instances [1]. Due to the intrinsic hiccups of the earlier heuristic, and meta-heuristic methodologies, hyper-heuristic has emerged as a feasible search methodology for solving multifarious COPs occurring in varying practical applications [2]. It has been effectively applied in multiple application domains, including scheduling [3], [4], timetabling [5], routing [6]–[8], software engineering [9], [10], and manufacturing [11]. In general, hyper-heuristics provide two different types of search space, which are low-level heuristics (LLHs) and acceptance mechanisms. However, how to select LLHs and combine them with an acceptance mechanism to realize an effective strategy for solving different COPs is particularly challenging.

In recent times, different methods, including machine learning have been investigated to improve the performance of hyper-heuristic optimization strategies. In this paragraph, we briefly review some related works that have attempted to improve the performance of hyper-heuristics. The Q-learning was utilized to select LLHs for a multi-objective route planning problem [12] and to choose an action in solving the interaction testing problem [9]. Deep Q-network [6] was applied as a heuristic selection mechanism to solve two routing problems from the library of hyper-heuristics flexible (HyFlex) [13]. In addition, Q-learning was applied to solve six problems from the HyFlex library by learning the pair of selection and acceptance mechanisms that are most suitable for an instance of a given problem [14]. Thompson sampling (TS) learning based on the selection of LLHs was recently introduced for solving COPs [15]–[17]. Moreover, TS has been applied to automatically configure the perturbation behavior of iterated local search (ILS) to solve six HyFlex COPs [18]. The evolutionary-based ILS hyper-heuristic was recently designed and tested on the extended HyFlex COPs to provide further evidence of the necessity for more categories of algorithms with improved perturbation strength [19].

A new perturbation strategy for ILS was proposed in [20] to solve the problems of pseudo-Boolean optimization where decision variables are perturbed to improve a local search strategy by maximizing the distance between solutions and maximizing the fitness similarity. An ILS algorithm was proposed in [21] for solving the problem of aircraft landing based on a search methodology that successively invokes a local search procedure to find a local optimum solution. The authors used a perturbation operator to modify the current solution to escape from the local optimum and provide a new solution for the local search procedure. The fair-share iterated local search (FS-ILS) [22] is a simple state-of-the-art selection hyper-heuristic that uses a conservative restart condition to prevent restarts, and only restart when a method is stuck and enough time is available to attain a solution of similar quality. The work [23] presented an efficient algorithm for solving the problem of aircraft landing based on a mechanism that hybridizes the ILS and simulated annealing (SA) algorithms to find a feasible aircraft scheduling solution within the range of target time. The sequence-based selection hyper-heuristic inspired by a hidden Markov model was proposed in [24] as a general purpose hyper-heuristic for solving cross-domain COPs. However, the authors conceded that the performance of a selection hyper-heuristic may vary depending on the choice of the LLHs and that not all the heuristics contribute to the improvement of a candidate solution during the search process unless they are applied in a combination with sequences of heuristics.

Despite myriads of research publications on hyper-heuristics, the published results on cross-domain optimization algorithms still need further improvement. The confines shown by the existing algorithms for solving different COPs have provided a unique opportunity to improve the performance of a hyper-heuristic across different problem domains. The findings from the literature have generally suggested that identifying the right intensity of perturbation operations in ILS is challenging but warrants further investigation [20]. The focus of the present work is to extensively experiment with three categories of the TS-ILS hyper-heuristic [18] imbued with different perturbation behaviors and compare their performances against the existing benchmark algorithms on HyFlex COPs. The testing of a hyper-heuristic algorithm on standard problems with varying characteristics will enable a meticulous comparison of its generalization capability. Since an ILS-based hyper-heuristic is potentially targeted toward solving numerous COPs, an efficient strategy for determining its proper perturbation behavior on a specific problem is paramount to the success of ILS methodology [7], [21].

The overarching objectives of the present work are threefold. To describe three categories of the TS-ILS hyper-heuristic that combine low-level heuristics with an acceptance mechanism for solving standard HyFlex COPs. To experimentally compare the three categories of the TS-ILS hyper-heuristic against the existing benchmark algorithms on the standard HyFlex COPs. To experimentally determine the effects of LLHs categorization on the standard HyFlex COPs. The remaining parts of the paper are fleetingly organized as follows. Section II describes the TS-ILS hyper-heuristic with three categorizations of LLHs. Section III presents the experimental results of comparing the three categories of the TS-ILS hyper-heuristic against the existing benchmark algorithms on the standard HyFlex COPs. Section IV examines the effects of LLHs categorization on the standard HyFlex COPs. Section V discusses the study results and highlights the potential areas for further improvement. The paper is ultimately concluded in Section VI by explicating the category of the TS-ILS hyper-heuristic that recorded a good balance between "single shaking" and "double shaking" configurations.

II. Thompson Sampling Iterated Local Search

Thompson sampling iterated local search (TS-ILS) hyper-heuristic is a probabilistic learning of profitable perturbation operations for a problem instance [19]. The hyper-heuristic augments the functional capabilities of TS and ILS to control the perturbation behavior of ILS. It selects LLHs and accepts or rejects a solution using the FS-ILS [22]. The local search phase of the hyper-heuristic is triggered after a perturbation process to improve the solution obtained from the perturbation stage, while the resultant solution is then considered for acceptance. There are six configurations in the config = {0, 1, 2, 3, 4, 5} set and each one is an integer representation of a perturbation operation. Table I defines each element of a perturbation configuration set of length n, where the value of n was taken to be 6 in the present work.

TABLE I. Perturbation Operations of the TS-ILS Hyper-Heuristic

Value	Operation
0	Perturb with Mutation LLH + Mutation LLH (Mut + Mut)
1	Perturb with Mutation LLH + Ruin-Recreate LLH (Mut + RR)
2	Perturb with Ruin-Recreate LLH + Mutation LLH (RR + Mut)
3	Perturb with Ruin-Recreate LLH + Ruin-Recreate LLH (RR + RR)
4	Perturb with Mutation LLH only (Mut)
5	Perturb with Ruin-Recreate LLH only (RR)

The TS-ILS hyper-heuristic algorithm learns promising perturbation operations for the ILS by splitting the mutation (Mut) and ruin-recreate (RR) heuristics into two distinct entities as in the first two lines of Algorithm 1. The two vectors α and β are respectively representing the success and failure tallies for the perturbation configurations in the config set. The pair of elements in the two vectors at the same corresponding positions respectively correspond to the success and failure counts of the options in the config set. The initial solution for the problem instance being solved is generated. The resultant solution is then used to initialize the current solution S_0 and the best solution found so far (S_b). The Thompson sampling procedure generates the utility values for the elements of the config set based on tallies that are stored in the vectors α and β by sampling from a Beta distribution. This phase of the TS-ILS hyper-heuristic algorithm decides which of the perturbation operations that are represented in the config set is to be invoked. These perturbation operations can be seen as the alternative operators to be selected in the bandit problem. The corresponding first element, α_0 and β_0 of the α and β vectors are respectively passed as parameters to the sampling module to generate the utility value for the first element in the config set. The pseudocode for the TS-ILS hyper-heuristic algorithm is given by the following **Algorithm 1**.

Algorithm 1: TS-ILS Hyper-heuristic

```
$M \leftarrow \{m_1, m_2, \dots, m_j\}$
$R \leftarrow \{r_1, r_2, \dots, r_k\}$
config $\leftarrow \{c_1, c_2, \dots, c_n\}$ ⊳ Line 3
$\alpha \leftarrow \{0, \dots, 0\}$
$\beta \leftarrow \{0, \dots, 0\}$
$S_0 \leftarrow$ generateInitialSolution()
$S_b \leftarrow S_0$
while (¬stopping_condition) do
    $\Phi = \{\phi_1, \dots, \phi_l\} \leftarrow$ generateUtilityValues()
    select **i** from config: $\phi_i$ maximizes $\Phi$
    $S' \leftarrow$ perturb($S_0, i$)
    $S'' \leftarrow$ localSearch()
    if ($S''$ is accepted) then
        $S_0 \leftarrow S''$
    end
    if ($S'' < S_b$) then
        $\alpha_i \leftarrow \alpha_i + 1$
        updateLS()
        updateParam()
    else
        $\beta_i \leftarrow \beta_i + 1$
    end
    updateLLH()
end
```

Studying the effects of LLHs categorization on the standard HyFlex COPs is paramount to the present work to determine the most appropriate algorithm for solving cross-domain COPs. Thus, TS-ILS1, TS-ILS2, and TS-ILS3 hyper-heuristics constitute three categories of the TS-ILS hyper-heuristic algorithm. The TS-ILS1 uses the subset {4, 5} and perturbs a solution once before intensification. The TS-ILS2 uses the subset {1, 2, 4, 5} and has been featured in the previous study [18]. The TS-ILS3 uses the entire set {0, 1, 2, 3, 4, 5} and all the options presented in Table I during the perturbation stage. The TS-ILS1 can only perturb a solution once before the intensification (single shaking) phase while the last two categories can perturb a resolution twice before the intensification (double shaking) phase. These differences are enforced by line 3 of Algorithm 1 [18] which has been extended in this study to any set of perturbation configurations. The categorizations have significantly affected the TS-ILS hyper-heuristic for solving diverse COPs. This is because the different perturbation strengths determine the effectiveness of the ILS-based hyper-heuristics [7], [20], [23].

The element of the set config that maximizes the utility values in Φ is selected, and the corresponding perturbation operation is carried out as designated in the next two lines. During the perturbation phase, the ***speedNew*** selection mechanism [14], [22] is employed to choose a given candidate LLH selected by the TS procedure. For example, if the selected configuration is 2, the selection mechanism chooses a perturbative LLH from the ruin-recreate set and applies it to a solution. The resultant solution is further perturbed by selecting and applying a perturbative LLH from the mutation set. The intensity of mutation and the depth of search are two parameter archetypes in the HyFlex framework for parameterizing the use of LLHs [13]. A parameter value is chosen within the {0.0, 0.1, …, 1.0} set of 11 values using a roulette-wheel procedure for the selected LLH to be analogously parameterized as in [24]. The local search module of the TS-ILS hyper-heuristic [18], [19] is triggered on the resultant solution to produce another solution (S''). The next operation decides if S'' is to be accepted to replace the current solution S_0 based on the accept probabilistic worse (APW) acceptance mechanism [14], [22].

The success tally (α_i) of the i element of the config set invoked in the current iteration is incremented only if the solution generated is strictly better than the best solution (S_b) found so far, otherwise, the value of β_i is incremented. This update scheme has enabled the TS-based probabilistic learning algorithm to adjust its preference according to the observed rewards of the alternative actions to be taken at every iteration. The function updateLLH() updates the parameters of the perturbative LLHs applied based on the speedNew selection mechanism. This update scheme does not apply to the local search heuristics and further details of how it is carried out can be found in [22]. The function updateLS() updates the data structures of the local search heuristics employed for the local search phase of the ILS hyper-heuristic. Finally, the function updateParam() updates the utility matrix used by the parameters of parameterized LLHs from the local search, mutation, and ruin-recreate operations. The entry for the parameter value selected by a roulette wheel procedure is updated after the iteration. The selection and update of values for the parameterized LLHs are analogous to the implementation in [24].

III. Experimental Results

Three categories of the TS-ILS hyper-heuristic were implemented on an Intel i5-3340M CPU computer with random access memory of 8 gigabytes and a 2.70 gigahertz clock speed. The TS-ILS1, TS-ILS2, and TS-ILS3 hyper-heuristics were tested on the problem instances of Boolean satisfiability (SAT), Bin packing (BP), Personnel scheduling (PS), Permutation flow-shop (PFS), Travelling salesman problem (TSP), and Vehicle routing problem (VRP) reported in HyFlex v1.0. The testing was also performed on the ten instances of the Knapsack problem (KP), Quadratic assignment problem (QAP), and Maximum cut (MAC) problem reported in HyFlex v2.0. The execution time returned by a benchmark program on the computer machine is 507 seconds, which is the equivalent of 600 seconds on a standard testing machine according to the organizers of the cross-domain heuristic search challenge (CHeSC) in 2011.

The comparison of the different algorithms is based on the metrics of median objective function values (ofvs), formula one, μ-norm, and boxplot visualization as subsequently illustrated. The performances of the three categories of the TS-ILS hyper-heuristic were compared using the ofvs across nine HyFlex COPs. In addition, the performances of TS-ILS1, TS-ILS2, and TS-ILS3 were compared with those of the FS-ILS, NR-FS-ILS, AdapHH, EPH, SR-IE, SR-AM, and SSHH benchmark algorithms [25], [26] across eight HyFlex COPs. The results obtained for the PS problem by the existing algorithms could not be compared with those computed by TS-

ILS1, TS-ILS2, and TS-ILS3 because of the differences in the updated Java library used in the present work. The updated library has fixed a bug in the previous library that was used to produce the results [26]. The overall comparison of the algorithms will be prejudicial if an attempt is made to incorporate the results obtained for the PS problem. The data used for testing the existing algorithms, excluding the SSHH on HyFlex problems, were obtained online (https://github.com/Steven-Adriaensen/hyflext). The ofvs of the results computed by the SSHH algorithm can be found in [25]. In total, 60 instances of HyFlex COPs were tested for each of the three categories of TS-ILS hyper-heuristic based on the median ofvs. Moreover, 55 instances of the HyFlex COPs were tested separately for each of the three categories of the TS-ILS hyper-heuristic. The three categories were also compared with seven benchmark algorithms based on the formula one, μ-norm, and boxplot visualization of median ofvs.

A. *Comparison based on Median ofvs*

Tables II and III highlight the performances of the TS-ILS categories in terms of the median ofvs obtained across the benchmark instances of the HyFlex COPs. The KP, QAP, and MAC problems of HyFlex v2.0 have ten benchmark instances which are five more than the first six problems of SAT, BP, PS, PFS, TSP, and VRP in HyFlex v1.0. The values in bold font denote the median ofv of the hyper-heuristic that reported the best performance for a problem instance.

TABLE II. MEDIAN OFVS OBTAINED BY CATEGORIES OF TS-ILS HYPER-HEURISTIC ON SIX HYFLEX V1.0 PROBLEMS

Problem	Category	Problem Instance				
		1	*2*	*3*	*4*	*5*
SAT	TS-ILS1	**2.00000000**	**2.00000000**	**1.00000000**	**1.00000000**	9.00000000
	TS-ILS2	**2.00000000**	3.00000000	**1.00000000**	**1.00000000**	**8.00000000**
	TS-ILS3	**2.00000000**	3.00000000	**1.00000000**	**1.00000000**	9.00000000
BP	TS-ILS1	0.02371400	0.00807447	0.00491703	**0.10828062**	0.01260111
	TS-ILS2	0.01876799	**0.00350695**	**0.00052035**	0.10828402	**0.00142866**
	TS-ILS3	**0.01828719**	0.00355599	0.00236484	0.10828455	0.00557662
PS	TS-ILS1	**19.00000000**	**9546.00000000**	3213.00000000	1609.00000000	**330.00000000**
	TS-ILS2	21.00000000	9548.00000000	**3181.00000000**	**1550.00000000**	**330.00000000**
	TS-ILS3	21.00000000	9570.00000000	3193.00000000	1593.00000000	335.00000000
PFS	TS-ILS1	**6223.00000000**	**26755.00000000**	**6323.00000000**	**11327.00000000**	**26585.00000000**
	TS-ILS2	6232.00000000	26785.00000000	6325.00000000	11340.00000000	26601.00000000
	TS-ILS3	6237.00000000	26788.00000000	**6323.00000000**	11354.00000000	26605.00000000
TSP	TS-ILS1	**48194.92010000**	**20701672.20000000**	6809.10000000	66194.70000000	53806.20000000
	TS-ILS2	**48194.92010000**	20779493.20000000	6805.30000000	**66133.00000000**	53762.40000000
	TS-ILS3	**48194.92010000**	20817079.70000000	**6804.70000000**	66150.80000000	**53635.70000000**
VRP	TS-ILS1	65151.40000000	13290.50000000	146927.10000000	20654.10000000	145865.40000000
	TS-ILS2	63709.00000000	13292.80000000	**145401.50000000**	20654.70000000	**145205.40000000**
	TS-ILS3	**62658.50000000**	**13285.50000000**	146801.90000000	**20654.00000000**	145436.20000000

TABLE III. MEDIAN OFV OBTAINED BY CATEGORIES OF TS-ILS HYPER-HEURISTICS ON THREE HYFLEX V2.0 PROBLEMS

Problem	Problem Instance	TS-ILS1	TS-ILS2	TS-ILS3
KP	0	**-104046**	**-104046**	**-104046**
	1	-1257913	-1258367	**-1259059**
	2	**-242324**	-242255	-242179
	3	-431342	-431340	-431336
	4	**-396167**	**-396167**	**-396167**
	5	-4254605	-4254402	-4252958
	6	**-941561**	-940026	-939070
	7	**-1577175**	**-1577175**	**-1577175**
	8	**-1530489**	-1530479	-1530470
	9	**-1467357**	**-1467357**	**-1467357**
QAP	0	**152112**	152132	152156
	1	**153972**	154036	154036
	2	**147894**	147952	147944
	3	149782	**149768**	149778
	4	21276862	**21269484**	21307840

	5	1187188954	1186656060	**1186575184**
	6	**501189032**	501487948	501258178
	7	**44863086**	44865718	44870864
	8	8154812	8155312	**8153852**
	9	**273212**	273228	273266
MAC	0	-41375743	**-41417466**	-41324351
	1	-277192517	-276843715	**-277614604**
	2	-3054	-3054	**-3055**
	3	-3032	**-3034**	**-3034**
	4	-3037	**-3039**	-3038
	5	-13217	-13216	**-13227**
	6	-1354	**-1358**	-1356
	7	-10077	**-10093**	-10084
	8	**-456**	**-456**	**-456**
	9	-2906	**-2914**	-2912

B. Comparison based on Formula One

The formula one scoring system has inspired one of the well-known metrics for evaluating hyper-heuristics [2]. Competing hyper-heuristics are assigned points based on the ofvs of their median best solutions obtained after 31 trials for each problem instance in the given test suite. The scores of 10, 8, 6, 5, 4, 3, 2, and 1 are respectively awarded to the best performing hyper-heuristic down to the eight-best one for a problem instance. Ties are handled by averaging the points that would have been given to one hyper-heuristic if there was no tie and assigning each of the hyper-heuristics the average score. In the rating system, the higher the score, the better the performance of a hyper-heuristic relative to the median results obtained by other hyper-heuristics.

The results of comparing eight hyper-heuristics on HyFlex COPs using formula one scores are presented in Table IV. The overall score in the table is the sum of scores obtained by a given hyper-heuristic across problem instances. The categories of the TS-ILS hyper-heuristic can be observed to emerge as the most dominant hyper-heuristics across the HyFlex COPs considered. The overall performance of the categories of the TS-ILS hyper-heuristic on HyFlex v2.0 COPs was found to be superior to the performances of the other hyper-heuristics. The maximum score for each HyFlex v2.0 problem domain is 100 with the highest score of 10 for each problem instance. It can be inferred that the top three hyper-heuristics on HyFlex v1.0 problem based on the formula one scoring are TS-ILS2 with 162.8 points, followed by TS-ILS3 with 148.8 points, and TS-ILS1 with 146.9 points. The order of performances of the algorithms on HyFlex v2.0 problem instances is TS-ILS2 with 215.8 points, followed by TS-ILS1 with 214.8 points, and TS-ILS3 with 203.3 points. It is noticeable that there is a close race performance among the three categories of the TS-ILS hyper-heuristic on HyFlex v2.0, while TS-ILS2 outperformed the other categories on HyFlex v1.0. problems.

TABLE IV. FORMULA ONE RANKING OF CATEGORIES OF TS-ILS HYPER-HEURISTIC ON EIGHT HYFLEX PROBLEMS, EXCLUDING PS

Problem	**AdapHH**	**EPH**	**FS-ILS**	**NR-FS-ILS**	**SR-AM**	**SR-IE**	**SSHH**	**TS-ILS1**	**TS-ILS2**	**TS-ILS3**
SAT	21.00	10.00	34.85	23.35	0.00	5.00		**35.10**	34.85	30.85
BP	18.00	19.00	11.00	18.00	0.00	18.00		29.00	**44.00**	38.00
PFS	19.50	11.50	25.00	27.50	5.00	0.00		**47.00**	32.50	27.00
TSP	23.00	26.00	25.50	22.00	2.00	3.00		29.50	**33.00**	31.00
VRP	20.00	16.00	26.00	22.00	0.00	8.00		28.00	35.000	**40.00**
Overall	101.50	82.50	122.35	112.85	7.00	34.00		168.60	**179.35**	166.85
KP	59.23	49.33	6.21	11.21	19.85	7.83	48.33	**69.33**	62.33	56.33
QAP	40.00	26.00	31.50	40.50	20.00	0.00	2.00	**84.00**	74.00	72.00
MAC	28.50	5.00	14.50	20.00	36.50	1.00	68.50	61.50	**79.50**	75.00
Overall	127.73	80.33	52.21	71.71	76.36	8.83	118.83	214.83	**215.83**	203.33

C. Comparison based on µ-norm Metric

This section compares the performances of ten hyper-heuristics across eight HyFlex COPs based on the µ-norm scores [25], [26]. The µ-norm is the average normalized evaluation function value. It is a more robust evaluation metric than the formula one scoring because it evaluates the performance of a hyper-heuristic based on the quality of the 31 solutions obtained over 31 trials on a problem instance. The µ-norm metric enables all the obtained ofvs to be normalized within the range [0, 1], where 0 means a hyper-heuristic outperforms other hyper-heuristics on all the tested instances, and a value of 1 connotes the opposite.

Table V provides comparative results of the categories of TS-ILS hyper-heuristics against the existing ones using the µ-norm scores. The data for the existing hyper-heuristics on the problem domains presented in Table V were taken from the paper [26]. The categories of the TS-ILS hyper-heuristic jointly won seven out of the eight HyFlex problems. They outperformed the other hyper-heuristics on the PFS, KP, and QAP. The only problem domain where none of the categories of the TS-ILS hyper-heuristic recorded the best µ-norm value is SAT, where FS-ILS emerged as the best hyper-heuristic. The AdapHH is the closest challenger to the top algorithm (TS-ILS1) on the Knapsack problem. The three categories of the TS-ILS hyper-heuristic dominated all others on the QAP and MAC problems because they all constituted the top three successful algorithms across the problem domains. The EPH and SR-IE algorithms obtained the worst results on the MAC problem. The overall performance in Table V further consolidates the observation that the three categories of the TS-ILS hyper-heuristic are general in their applications to HyFlex v2.0 problems. Overall, the next best algorithms based on the µ-norm score, after the three categories of the TS-ILS hyper-heuristic, are AdapHH and FS-ILS, while the SR-AM algorithm delivered the worst performance.

TABLE V. COMPARATIVE RESULTS USING µ-NORM ON EIGHT HYFLEX PROBLEMS, EXCLUDING PS

Problem	TS-ILS2	TS-ILS3	TS-ILS1	AdapHH	FS-ILS	NR-FS-ILS	EPH	SR-IE	SR-AM
SAT	0.0159	0.0184	0.0181	0.0276	**0.0146**	0.0238	0.0927	0.3787	0.8759
BP	**0.0138**	0.0316	0.0852	0.1828	0.1727	0.1581	0.1478	0.1769	0.9559
PFS	0.1676	0.1817	**0.1263**	0.2224	0.2059	0.1816	0.2671	0.7242	0.6223
TSP	**0.0538**	0.0556	0.0584	0.0677	0.0647	0.0626	0.0658	0.4993	0.5392
VRP	0.0623	**0.0538**	0.0731	0.0841	0.0687	0.0832	0.2186	0.2714	0.9347
KP	0.0315	0.0308	**0.0276**	0.0297	0.1513	0.0554	0.3625	0.3312	0.3970
QAP	0.0728	0.0795	**0.0689**	0.1089	0.1512	0.1396	0.1062	0.6363	0.1097
MAC	0.1018	**0.0987**	0.1112	0.2829	0.2585	0.5222	0.3772	0.7371	0.3946
Overall	**0.0649**	0.0688	0.0711	0.1258	0.1360	0.1533	0.2047	0.4694	0.6037

D. Comparison based on Boxplot Visualization of Median ofvs

Fig. 1 presents the boxplots of the normalized ofvs of ten hyper-heuristics in Table IV. The minimum–maximum normalization scheme was applied to obtain the normalized median ofv of a hyper-heuristic on a particular instance of a problem domain [25]. The performances of the categories of the TS-ILS hyper-heuristic were benchmarked against the existing hyper-heuristics on HyFlex problems [26].

KP, The median score of the TS-ILS2 appears to be closest to the base of the plot in Fig. 1(a) to indicate good performance. A similar phenomenon can be observed for the TS-ILS1 and TS-ILS3 categories. The TS-ILS2 and TS-ILS3 have smaller boxes than the TS-ILS1 category. The three categories performed better than any of the other algorithms on the HyFlex v1.0 problems. The gap in the performance of the categories of the TS-ILS hyper-heuristic and other hyper-heuristics is more glaring for the QAP, and MAC HyFlex v2.0 problems.

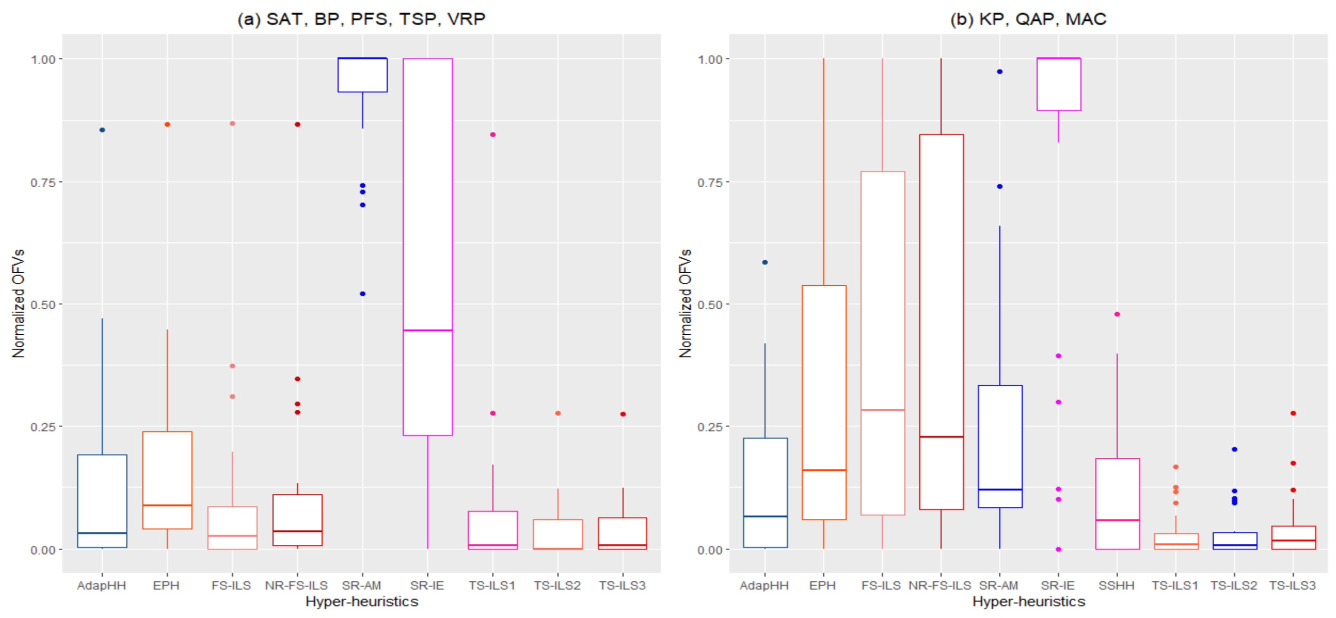

Fig. 1. Boxplots of the overall normalized median ofvs obtained by the hyper-heuristics in Table IV.

The sizes of the boxes in Fig. 1(b) support the dominance of the categories of the TS-ILS hyper-heuristic. The interquartile range of the boxplots of each of the three categories of the TS-ILS hyper-heuristic is a minimal value, denoting low variability in their data. Combining the inter-quartile range with the proximity of the boxes to the minimum score means that the categories have enjoyed dominance over other algorithms on almost all problem instances of Knapsack, Quadratic Assignment, and Maximum-Cut. The categories can be observed to generalize well across eight HyFlex problem domains in sharp contrast to the other standard ILS hyper-heuristics, like FS-ILS and NR-FS-ILS.

IV. EFFECTS OF LOW-LEVEL HEURISTICS CATEGORIZATION ON HYFLEX PROBLEMS

The effects of LLHs on HyFlex problems have been investigated in this work. The first TS-ILS1 hyper-heuristic does not employ the double shaking strategy, but the other two categories that respectively implemented four and six perturbation configurations do. The problem domains of Bin packing, Permutation flow-shop, Knapsack, Maximum cut, Vehicle routing, and Quadratic assignment were chosen to demonstrate the differences among the three categories of the TS-ILS hyper-heuristic. The values in the config set have been defined according to the entries in Table I. The parameter C1 represents the first option that applied two mutation heuristics in succession before the application of the intensification heuristic. The parameter C6 represents the application of only one ruin-recreate heuristic before applying the intensification heuristic. The heuristic calls were recorded throughout the problem-solving process to obtain the boxplots. In addition, the perturbation configurations applied at each iteration were recorded accordingly. If a successful iteration of the best new solution is produced, the selected tally of a configuration was recorded. The log files of the top three runs for each problem domain were congregated for each category of the TS-ILS hyper-heuristic as shown in Fig. 2 to 4.

It is important to explain Fig. 2 to 4 to provide more clarity before interpreting the results provided by the figures. The multiple bar charts provide the success rates of the six different perturbation configurations (C1 to C6) for the three TS-ILS categories of TS-ILS1 (red), TS-ILS2 (blue), and TS-ILS3 (black). TS-ILS1 is a single shaking variant that utilizes the two configurations of mutation only (C5) and ruin-recreate only (C6) and therefore, it explains why only two red bars appear in each of the charts. Similarly, TS-ILS2 utilizes four configurations (C2, C3, C5, and C6) and it explains why only a maximum of four blue bars can be seen on each chart. The most successful configuration for both the TS-ILS2 and TS-ILS3 is C6 using the BP9 instance of Fig. 2 as an example, while the most successful configuration for TS-ILS3 is C4. This means that the application of only the ruin-recreate heuristic (C6) has found more best new solutions than any other configuration during the run of TS-ILS1 and TS-ILS2. However, for TS-ILS3, the application of two ruin-recreate heuristics in succession before the intensification (C4) was found to be the most productive option. Consequently, because TS-ILS2 outperformed TS-ILS3 on the multiple trials on the BP9 instance, it could be said that TS-ILS3 having so many configurations (six) slowed down its performance on the BP9 instance when compared to the performance of TS-IL2. Finally, the presence of the double shaking configuration (C2) in both TS-IL2 and TS-ILS3 has made them superior solvers than TS-ILS1 on the BP9 instance because the configuration has contributed to almost 40% of the best solutions found during the runs of TS-ILS2 and TS-LS3.

Most successful iterations were achieved with the application of ruin-recreate heuristics for the TS-ILS1 on the BP problem domain. This can be seen in the C6 column of the TS-ILS1 which has the highest bar for all instances. The TS-ILS2 and TS-ILS3 are better algorithms for solving the BP problem. They utilize more pairing of heuristics, especially with the pairing of Mut and RR (C2). Though, an exception is observed in the BP10 instance, where a single application of

RR is the most rewarding strategy. This phenomenon explains why the TS-ILS1 category, which is the weakest algorithm on the BP problem, outperformed its counterparts on the BP10 instance. The reason is that it could easily focus on the RR among only two options with the second option of Mut (C5) being a bad choice. The overall comparison of the performances of the hyper-heuristics on the BP problem domain has shown in Table II that the TS-ILS2 and TS-ILS3 are better than the TS-ILS1 across four of the five problem instances. This can be traced to their heavy reliance on C2, which cannot be observed for the TS-ILS1 category.

The single application of perturbation heuristics from the Mut is the best strategy for solving the instances of the PFS problem as evidenced in the plots of PFS3, PFS8, PFS10, and PFS11. It is not surprising that the TS-ILS1 outperformed the TS-ILS2 and TS-ILS3 categories. The six options available to the TS-ILS3 have appeared to be noisy. The reason is that it would take a considerable number of epochs for the TS procedure to converge using only the Mut strategy (C5) while solving the instances of the PFS problem. The double shaking options available to the TS-ILS2 and TS-ILS3 somewhat mired them from performing at a higher level for most instances of KP2, KP5, and KP6 of the knapsack problem. Interestingly, the four instances of KP1, KP2, KP5, and KP6 perfectly present the TS-ILS2 with four options as the most balanced based on the number of perturbation configurations. Observing the behavior of the TS-ILS3 on the MAC problem instances, none of the double shaking options of Mut + RR, RR + Mut, RR + RR, and Mut + Mut have a lower bar than the single perturbation options. This observation implies the importance of pairing heuristics for solving these instances, and it explains why the TS-ILS2 and TS-ILS3 outperformed the TS-ILS1 on the problem. More interestingly, the performance of the TS-ILS2 on the MAC problem diverges from the performance of the other categories. Comparing their median ofvs across the ten problem instances, the TS-ILS2 obtained a lower (better) value across five problem instances while the TS-ILS3 managed to achieve the same feat in only three problem instances. This means that the two double-shaking options available to it are sufficient to make it excel in solving the MAC problem instances.

The mutation heuristics are more appropriate for the VRP problem because the column for the application of mutation heuristics is way longer than the application of ruin-recreate heuristics for the three categories of the TS-ILS hyper-heuristic. It is innocuous to generally conclude that the TS-ILS3 is a better solver of the VRP problem instances. This assertion can be justified by comparing the figures for the TS-ILS1, TS-ILS2, and TS-ILS3 on the VRP problem with their median ofvs in Table II. The top two perturbation configurations of the TS-ILS hyper-heuristic are the application of Mut and Mut + Mut (Fig. 4). The latter may be why the TS-ILS3 has performed better than the TS-ILS1 and TS-ILS2 on the VRP problem according to the values presented in Table V. This is because only the TS-ILS3 possesses the Mut + Mut option. The TS-ILS1 effectively leveraged the effectiveness of mutation heuristics on the QAP, while the plot for the TS-ILS2 and TS-ILS3 have demonstrated their partial reliance on the Mut option only.

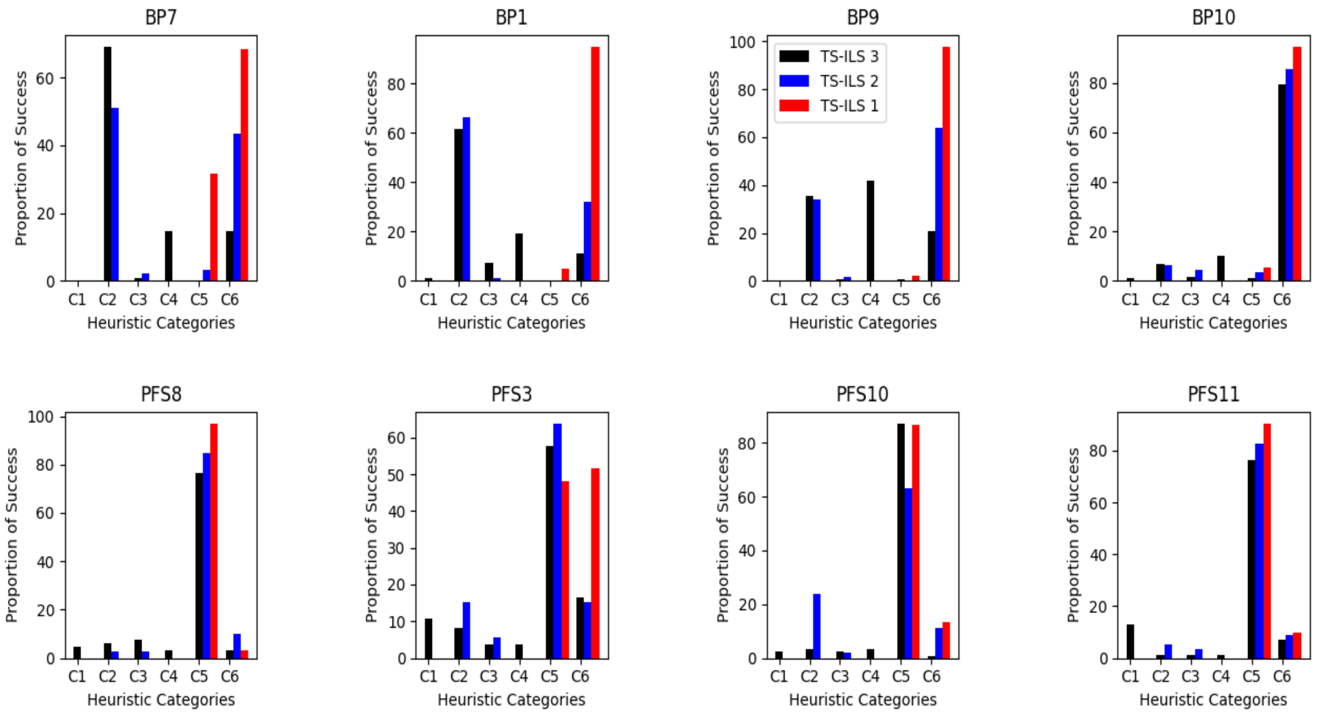

Fig. 2. The analysis of the perturbation configurations on the BP and PFS problems.

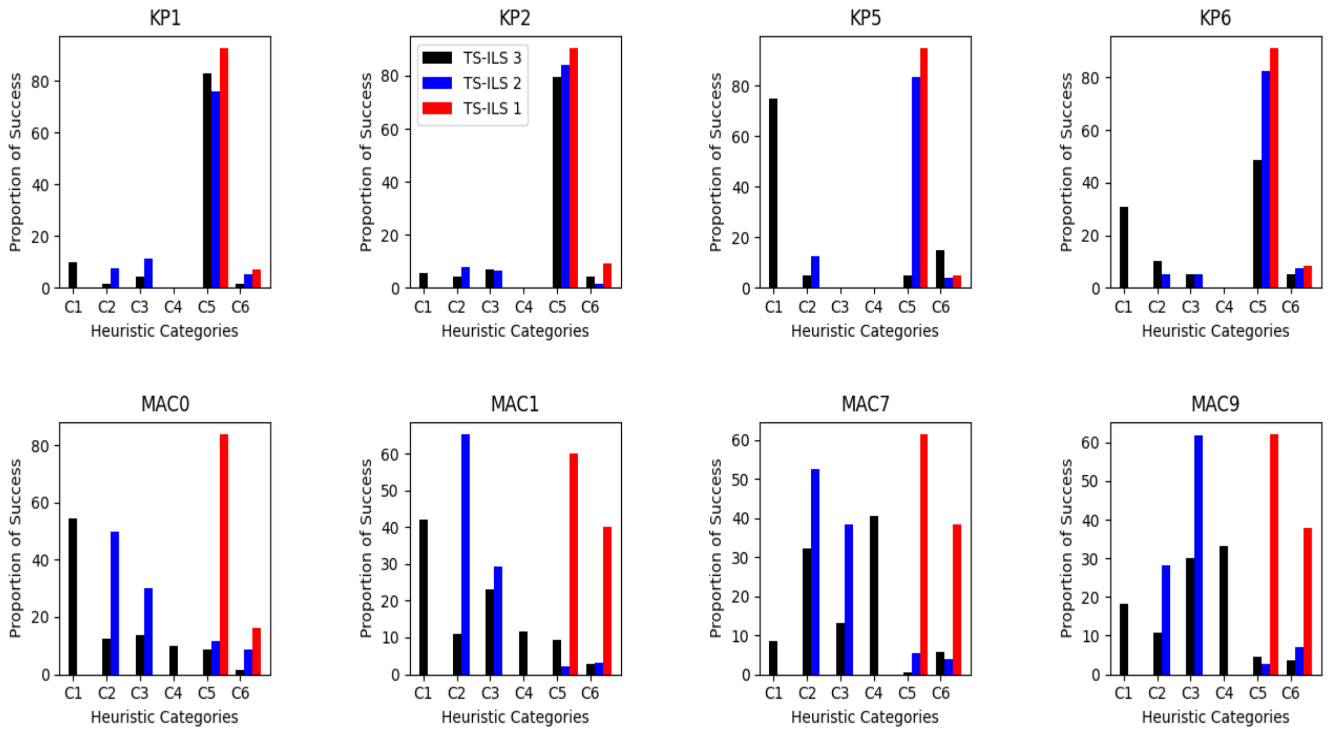

Fig. 3. The analysis of the perturbation configurations on the KP and MAC problems.

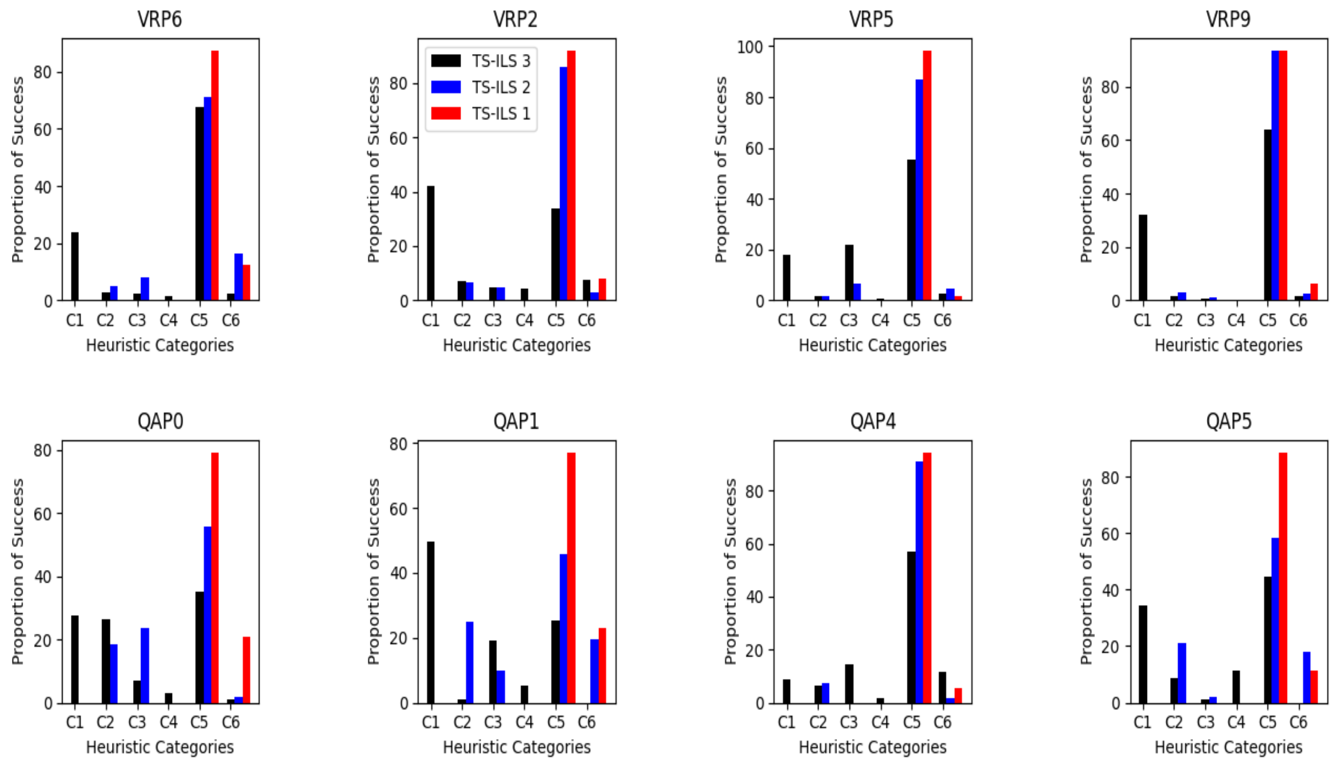

Fig. 4. The analysis of the perturbation configurations on the VRP and QAP problems.

V. DISCUSSION OF RESULTS

In this section, an extensive discussion of the results computed by the three categories of the TS-ILS hyper-heuristic is provided. The study results generally indicate why the TS-ILS algorithm is effective using a component-based analysis of the hyper-heuristic. In addition, a few areas where further improvement of the TS-ILS hyper-heuristic is required are highlighted in this section. The TS-ILS hyper-heuristics share similarities with existing ILS-based hyper-heuristics like FS-ILS [22]. The FS-ILS selects a single LLH heuristic during the perturbation stage by engaging a heuristic selection metric based on the ratio of the number of improvements made on an incumbent solution by an LLH to the total amount of its execution time when invoked. The TS-ILS hyper-heuristics use the same procedure to select LLHs, but the perturbation actions are governed by a Thompson sampling layer procedure that is featured in Algorithm 1.

Experimentation was set up to study the perturbation behavior of FS-ILS, TS-ILS1, TS-ILS2, and TS-ILS3 to reinforce the claim made in Section I about the need to control the depth of a perturbation. The experimentation profiles how these four hyper-heuristics solve cross-domain optimization problems, with results showing the weakness of some algorithms like FS-ILS, and why the successful categories of TS-ILS hyper-heuristics have overcome this weakness by varying perturbation control mechanisms. In the experimentation, three instances BP9, KP1, and MAC6 from three different problem domains were selected. The selected instances were taken from the problem domains where FS-ILS struggled to obtain good results to study if the perturbation control mechanisms of TS-ILS algorithms were responsible for their success. The four algorithms were run wherein each trial lasted for 279 secs, and the starting solution of each hyper-heuristics was initialized using the same seed. Tables VI, VII, and VIII present the results of the experiments for BP9, KP1, and MAC6 respectively.

There are seven trials for each algorithm, and the entries for the second trial of BP9 can be seen in the row "BP9-2". This convention is used in all the tables for easy understanding. The average values obtained for the four algorithms in the order of appearance in the tables on the BP9 instance are captured in the following set {0.01247419, 0.00565258, **0.00137980**, 0.00400226}. Likewise, the average values for the KP1 instance are {-1235575.0, **-1256610.9**, -1249428.4, -1250803.0} and finally, for MAC6, it is {-1330.3, -1348.3, **-1354.9**, -1350.6}. Each entry in the last six columns ("Mut + Mut" to "RR") represents the number of times a configuration was used during the run of a hyper-heuristic. For example, FS-ILS invoked 8,834 ruin-recreate heuristics and 7,324 mutation heuristics during its first trial as in row BP9-1 of Table VI.

The evaluation of the results presented in Table VI shows a huge discrepancy in the perturbation behavior of the FS-ILS and TS-ILS algorithms. It can be observed that while the TS-ILS algorithms choose to use more ruin-recreate heuristics, FS-ILS did not discriminate amongst the two categories of perturbative heuristics, although it did slightly favor the ruin-recreate heuristics. This explains why FS-ILS did not perform well on Bin packing problems. The poor performance of FS-ILS for the trials on the instance of KP1 can be blamed on the issue that plagued it. Its relatively lower number of perturbation-intensification cycles (sum of the invocations of the number of perturbation and ruin-recreate heuristics) when compared with the TS-ILS algorithms. For the first three trials on KP1, FS-ILS completed an average of 173.3 cycles as opposed to 1,019.7 cycles for TS-ILS, 1,074 cycles for TS-ILS2, and 1,150 cycles for TS-ILS3. This means that FS-ILS did not have enough opportunity to search the heuristic space (and ultimately the solution space) unlike the TS-ILS variants. The main reason for this problem of FS-ILS is its excessive invocations of local search heuristics during the intensification phase, as reported in a previous study [19]. Finally, on the MAC6 instance, although the number of cycles completed by FS-ILS is not too far from that of TS-ILS2 and TS-ILS3, it still fell short in relative performance, as shown by its reported ofv per trial in Table VIII. The other algorithms concentrated their best perturbation efforts on invoking two perturbative heuristics in succession before entering the intensification phase. For example, the following phenomenon can be observed in the behavior of TS-ILS2 during the last three trials of Table VIII. The invocation of double shaking (Mut + RR or RR + Mut) was more favored (71.8% of the time) than the single shaking strategy.

The experiments performed in this work have shed more light on why the strategies of TS-ILS algorithms are effective. Moreover, it has provided a deeper understanding of the shortcomings of the previous ILS-based algorithms that do not affect the TS-ILS algorithms. The ability of TS-ILS to automate its perturbation behavior and utilize a local search module that is mindful of excessive invocations of local search heuristics elevated its performance and offered its better generalization ability across the nine problem domains from the extended HyFlex library. In future works, TS-ILS2 should be used in conjunction with tabu search, hidden Markov, and other adaptive perturbation strategies for performance improvement. In addition, investigating the variations of TS-ILS2 such as TS-ILS3 for different applications is an attractive venture. In particular, it should be exciting to apply The S-ILS2 algorithm in the field of evolutionary dynamic optimization, where perturbation strategy can assume an influential role.

TABLE VI. EXPERIMENTAL RESULTS OF PERTURBATION PROFILE ON BP9

Instance	Algorithm	Objective Function Value	Mut + Mut	Mut + RR	RR + Mut	RR + RR	Mut	RR
BP9-1	FS-ILS	0.01243285					7324	8384
	TS-ILS1	0.00701407					452	17674
	TS-ILS2	**0.00256314**		2628	70		521	12159
	TS-ILS3	0.00501827	66	2899	537	7230	71	542
BP9-2	FS-ILS	0.01334600					7612	8311
	TS-ILS1	0.00389196					1175	16989
	TS-ILS2	**0.00148164**		7376	622		60	5550
	TS-ILS3	0.00498399	134	2059	489	8304	134	1502
BP9-3	FS-ILS	0.01010801					7702	8139
	TS-ILS1	0.00500537					780	15476
	TS-ILS2	**0.00270674**		2776	1441		488	9560

	TS-ILS3	0.00279360	108	3551	522	6030	109	2286
BP9-4	FS-ILS	0.01136454					7035	7977
	TS-ILS1	0.00463139					176	17709
	TS-ILS2	**0.00052194**		8191	180		374	4714
	TS-ILS3	0.00138283	344	5168	102	5044	272	2625
BP9-5	FS-ILS	0.01116060					7699	8526
	TS-ILS1	0.00484655					81	15532
	TS-ILS2	**0.00053781**		4068	614		64	12111
	TS-ILS3	0.00495967	454	1188	477	7388	309	2850
BP9-6	FS-ILS	0.01224321					7247	7933
	TS-ILS1	0.00481612					172	16357
	TS-ILS2	**0.00131507**		7965	76		133	8121
	TS-ILS3	0.00499338	121	1222	1018	8292	323	2174
BP9-7	FS-ILS	0.01666414					6847	7709
	TS-ILS1	0.00936258					480	14971
	TS-ILS2	**0.00053226**		6796	346		378	7622
	TS-ILS3	0.00388404	70	2530	108	6627	111	3881

TABLE VII. EXPERIMENTAL RESULTS OF PERTURBATION PROFILE ON KP1

Instance	Algorithm	Objective Function Value	Mut + Mut	Mut + RR	RR + Mut	RR + RR	Mut	RR
KP1-1	FS-ILS	-1245175.0					117	63
	TS-ILS1	**-1262316.0**					659	109
	TS-ILS2	-1254037.0		115	391		348	376
	TS-ILS3	-1259804.0	99	100	181	109	586	57
KP1-2	FS-ILS	-1225378.0					95	41
	TS-ILS1	**-1262078.0**					899	211
	TS-ILS2	-1261056.0		169	100		563	94
	TS-ILS3	-1247433.0	464	104	57	116	56	59
KP1-3	FS-ILS	**-1260047.0**					150	54
	TS-ILS1	-1245069.0					863	318
	TS-ILS2	-1256584.0		116	68		758	124
	TS-ILS3	-1259408.0	97	206	117	61	922	59
KP1-4	FS-ILS	-1231437.0					113	56
	TS-ILS1	-1258008.0					735	142
	TS-ILS2	-1258850.0		310	129		1034	144
	TS-ILS3	**-1259233.0**	49	111	50	59	546	102
KP1-5	FS-ILS	-1242785.0					117	62
	TS-ILS1	**-1258381.0**					744	505
	TS-ILS2	-1229977.0		186	551		86	92
	TS-ILS3	-1250668.0	454	1188	477	7388	309	2850
KP1-6	FS-ILS	-1207909.0					113	35
	TS-ILS1	-1251513.0					840	183
	TS-ILS2	**-1256018.0**		150	80		978	77
	TS-ILS3	-1250344.0	286	66	144	68	393	224
KP1-7	FS-ILS	-1236294.0					90	39
	TS-ILS1	**-1258911.0**					729	126
	TS-ILS2	-1229477.0		338	246		79	274
	TS-ILS3	-1228731.0	252	80	79	252	80	83

TABLE VIII. EXPERIMENTAL RESULTS OF PERTURBATION PROFILE ON MAC6

Instance	Algorithm	Objective Function Value	Mut + Mut	Mut + RR	RR + Mut	RR + RR	Mut	RR
MAC6-1	FS-ILS	-1328.0					6343	10412
	TS-ILS1	**-1350.0**					11410	11894
	TS-ILS2	**-1350.0**		7648	6852		589	2371
	TS-ILS3	-1344.0	4796	3065	4220	2059	1489	291
MAC6-2	FS-ILS	-1320.0					6734	9871
	TS-ILS1	-1340.0					4744	19056
	TS-ILS2	-1348.0		4408	7677		3209	1889
	TS-ILS3	**-1350.0**	1407	2491	5584	2966	2026	3415
MAC6-3	FS-ILS	-1322.0					6286	10955
	TS-ILS1	-1348.0					14548	9450
	TS-ILS2	-1350.0		11251	5128		2348	405
	TS-ILS3	**-1358.0**	1635	6021	4021	5437	344	2943
MAC6-4	FS-ILS	-1346.0					6728	10178
	TS-ILS1	**-1362.0**					25825	8968
	TS-ILS2	-1354.0		10312	6789		356	1135
	TS-ILS3	-1354.0	5292	6626	344	1055	2740	3639

MAC6-5	FS-ILS	-1330.0					6251	10932
	TS-ILS1	-1346.0					8292	25385
	TS-ILS2	**-1360.0**		9655	3989		2625	4659
	TS-ILS3	-1350.0	3148	320	4529	8014	1003	2698
MAC6-6	FS-ILS	-1336.0					6208	11061
	TS-ILS1	-1346.0					18116	11993
	TS-ILS2	**-1358.0**		2309	8734		3611	1875
	TS-ILS3	-1348.0	4371	318	1912	7161	1320	3532
MAC6-7	FS-ILS	-1330.0					6223	10677
	TS-ILS1	-1346.0					8633	22889
	TS-ILS2	**-1364.0**		14213	2685		1525	2069
	TS-ILS3	.1350.0	2553	3022	3167	4293	483	1574

VI. Conclusion

The objectives of the present work have been achieved through the description of three categories of the TS-ILS hyper-heuristic, experimentally comparing the three categories against the existing benchmark algorithms and the determination of the effects of LLHs categorization on HyFlex COPs. The TS-ILS2 with $\{1, 2, 4, 5\}$ subset configuration has edged out the other categories across eight HyFlex problems. It can be observed with a further granularity that the TS-ILS1 has struggled to effectively solve the instances of the Bin packing problem according to the comparison based on the μ-norm scores. This observation can be ascribed to the lack of double shaking or a stronger perturbation feature in its composition because it uses only $\{4, 5\}$ configuration subset. Although the TS-ILS1 outshone the other categories on the problems of Permutation flow-shop, Knapsack, and Quadratic assignment, its weakness was badly exposed when it was applied to solve the Bin packing problem. This eventually had a strong effect on its overall performance when compared to the other categories. The overall performances of the algorithms on the HyFlex v2.0 problems suggest a close race among the three categories after TS-ILS2 emerged as the overall best algorithm based on formula one and μ-norm scores.

The comparative results show that the TS-ILS3 recorded the best performance on three problem domains while the TS-ILS2 gave the best performance on the remaining problems. The uncanny fact is that the TS-ILS2 outperformed the TS-ILS3 on all but one of the problem domains. The TS-ILS2 recorded a better performance than the TS-ILS3 on the KP and MAC problems according to the formula one score, while the TS-ILS3 fared better than TS-ILS2 on the same problem domains based on the μ-norm scores. The reasonable explanation for this scenario is that the TS-ILS3 with more strategy options can sometimes obtain poor runs because of the convergence of the Thompson sampling module on the sub-optimal selection. Hence, this will affect its median ofv on the problem domains, but it has failed to perform better than TS-ILS2 in certain situations. In conclusion, the TS-ILS2 has a good balance between "single shaking" and "double shaking" configurations and emerged as the best hyper-heuristic algorithm for solving COPs.

Acknowledgment

The authors thank the Directorate of Research of Covenant University (CUCRID) for providing financial support for the publication of this article.

References

[1] S. P. Adam, S.-A. N. Alexandropoulos, P. M. Pardalos, and M. N. Vrahatis, "No Free Lunch Theorem: A Review," in Approximation and Optimization, I. C. Demetriou and P. M. Pardalos, Eds. Cham: Springer, 2019, pp. 57–82.

[2] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," Eur. J. Oper. Res., vol. 285, no. 2, pp. 405–428, 2020.

[3] A. Kheiri, A. Gretsista, E. Keedwell, G. Lulli, M. G. Epitropakis, and E. K. Burke, "A hyper-heuristic approach based upon a hidden Markov model for the multi-stage nurse rostering problem," Comput. Oper. Res., vol. 130, p. 105221, 2021, doi: 10.1016/j.cor.2021.105221.

[4] H. B. Song and J. Lin, "A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times," Swarm Evol. Comput., vol. 60, p. 100807, 2021, doi: 10.1016/j.swevo.2020.100807.

[5] G. Mweshi and N. Pillay, "An improved grammatical evolution approach for generating perturbative heuristics to solve combinatorial optimization problems," Expert Syst. Appl., vol. 165, p. 113853, 2021, doi: 10.1016/j.eswa.2020.113853.

[6] A. Dantas, A. F. do Rego, and A. Pozo, "Using deep Q-network for selection hyper-heuristics," in Proceedings of the Genetic and Evolutionary Computation Conference, 2021, pp. 1488–1492. doi: 10.1145/3449726.3463187.

[7] N. R. Sabar, S. L. Goh, A. Turky, and G. Kendall, "Population-based iterated local search approach for dynamic vehicle routing problems," IEEE Trans. Autom. Sci. Eng., pp. 1–11, 2021, doi: 10.1109/TASE.2021.3097778.

[8] Y. Zhang, R. Bai, R. Qu, C. Tu, and J. Jin, "A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties," Eur. J. Oper. Res., vol. 300, no. 2, pp. 418–427, 2022.

[9] B. S. Ahmed, E. Enoiu, W. Afzal, and K. Z. Zamli, "An evaluation of Monte Carlo-based hyper-heuristic for interaction testing of industrial embedded software applications," Soft Comput., vol. 24, no. 18, pp. 13929–13954, 2020, doi: 10.1007/s00500-020-04769-z.

[10] G. Guizzo, F. Sarro, J. Krinke, and S. R. Vergilio, "Sentinel: A Hyper-Heuristic for the Generation of Mutant Reduction Strategies," IEEE Trans. Softw. Eng., pp. 1–16, 2020, doi: 10.1109/TSE.2020.3002496.

[11] Y. Zhou, J. J. Yang, and L. Y. Zheng, "Multi-Agent Based Hyper-Heuristics for Multi-Objective Flexible Job Shop Scheduling: A Case Study in an Aero-Engine Blade Manufacturing Plant," IEEE Access, vol. 7, pp. 21147–21176, 2019, doi: 10.1109/ACCESS.2019.2897603.

[12] Y. Yao, Z. Peng, and B. Xiao, "Parallel Hyper-Heuristic Algorithm for Multi-Objective Route Planning in a Smart City," IEEE Trans. Veh. Technol., vol. 67, no. 11, pp. 10307–10318, 2018, doi: 10.1109/TVT.2018.2868942.

[13] G. Ochoa et al., "HyFlex: A benchmark framework for cross-domain heuristic search," in European Conference on Evolutionary Computation in Combinatorial Optimization, 2012, pp. 136–147.

[14] S. S. Choong, L. P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," Inf. Sci. (Ny)., vol. 436, pp. 89–107, 2018.

[15] A. Aslan, I. Bakir, and I. F. Vis, "A dynamic Thompson sampling hyper-heuristic framework for learning activity planning in personalized learning," Eur. J. Oper. Res., vol. 286, no. 2, pp. 673–688, 2020, doi: 10.1016/j.ejor.2020.03.038.

[16] M. Lassouaoui, D. Boughaci, and B. Benhamou, “A multilevel synergy Thompson sampling hyper-heuristic for solving Max-SAT,” Intell. Decis. Technol., vol. 13, no. 2, pp. 193–210, 2019, doi: 10.3233/IDT-180036.

[17] M. Scoczynski et al., “A selection hyperheuristic guided by Thompson sampling for numerical optimization,” in GECCO 2021 Companion - Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion, 2021, pp. 1394–1402. doi: 10.1145/3449726.3463140.

[18] S. A. Adubi, O. O. Oladipupo, and O. O. Olugbara, “Configuring the Perturbation Operations of an Iterated Local Search Algorithm for Cross-domain Search: A Probabilistic Learning Approach,” in 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 1372–1379.

[19] S. A. Adubi, O. O. Oladipupo, and O. O. Olugbara, “Evolutionary Algorithm-Based Iterated Local Search Hyper-Heuristic for Combinatorial Optimization Problems,” Algorithms, vol. 15, no. 11, p. 405, 2022, doi: 10.3390/a15110405.

[20] R. Tinos, M. W. Przewozniczek, and D. Whitley, “Iterated local search with perturbation based on variables interaction for pseudo-boolean optimization,” in Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 296–304.

[21] N. R. Sabar and G. Kendall, “An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem,” Omega, vol. 56, pp. 88–98, 2015.

[22] S. Adriaensen, T. Brys, and A. Nowé, “Fair-share ILS: A simple state of the art iterated local search hyperheuristic,” in Proceedings of the 2014 annual conference on genetic and evolutionary computation, 2014, pp. 1303–1310.

[23] A. I. Hammouri, M. S. Braik, M. A. Al-Betar, and M. A. Awadallah, “ISA: a hybridization between iterated local search and simulated annealing for multiple-runway aircraft landing problem,” Neural Comput. Appl., vol. 32, no. 15, pp. 11745–11765, 2020, doi: 10.1007/s00521-019-04659-y.

[24] A. Kheiri and E. Keedwell, “A sequence-based selection hyper-heuristic utilising a hidden Markov model,” in GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference, 2015, pp. 417–424.

[25] A. Almutairi, E. Özcan, A. Kheiri, and W. G. Jackson, “Performance of selection hyper-heuristics on the extended hyFlex domains,” in International Symposium on Computer and Information Sciences, 2016, pp. 154–162.

[26] S. Adriaensen, G. Ochoa, and A. Nowé, “A benchmark set extension and comparative study for the HyFlex framework,” in 2015 IEEE Congress on Evolutionary Computation, CEC 2015, 2015, pp. 784–791.