OSCAR: Online Selection of Algorithm Portfolios with Case Study on Memetic Algorithms

Mustafa Mısır(✉), Stephanus Daniel Handoko, and Hoong Chuin Lau

School of Information Systems, Singapore Management University, Singapore, Singapore
{mustafamisir,dhandoko,hclau}@smu.edu.sg

Abstract. This paper introduces an automated approach called OSCAR that combines algorithm portfolios and online algorithm selection. The goal of algorithm portfolios is to construct a subset of algorithms with diverse problem solving capabilities. The portfolio is then used to select algorithms from for solving a particular (set of) instance(s). Traditionally, algorithm selection is usually performed in an offline manner and requires the need of domain knowledge about the target problem; while online algorithm selection techniques tend not to pay much attention to a careful construction of algorithm portfolios. By combining algorithm portfolios and online selection, our hope is to design a problem-independent hybrid strategy with diverse problem solving capability. We apply OSCAR to design a portfolio of memetic operator combinations, each including one crossover, one mutation and one local search rather than single operator selection. An empirical analysis is performed on the Quadratic Assignment and Flowshop Scheduling problems to verify the feasibility, efficacy, and robustness of our proposed approach.

1 Introduction

We propose in this paper a framework that combines the ideas of *algorithm portfolio* and *online selection*. We call this framework OSCAR (Online SeleCtion of Algorithm poRtfolio). Algorithm selection [1] essentially learns the mapping between instance features and algorithmic performance, and this is usually performed in an offline fashion, as the process is typically very computationally intensive. The learned mapping can be utilized to choose the best algorithms to solve unseen problem instances based on their features. Algorithm portfolio [2,3] treats the algorithm selection problem in a broader perspective. The goal is to construct a diverse suite of algorithms that altogether are capable of solving a wide variety of problem instances, thus reducing the risk of failure. In terms of online algorithm selection, Adaptive Operator Selection (AOS) [4] deals with a single type of operators at a time, performs on-the-fly selection of evolutionary operators. Selecting from the pool of all possible combinations of crossover, mutation, and local search operators might be beneficial as this would capture the correlation among the different types of operators, but it could be

© Springer International Publishing Switzerland 2015
C. Dhaenens et al. (Eds.): LION 9 2015, LNCS 8994, pp. 59–73, 2015.
DOI: 10.1007/978-3-319-19084-6_6

challenging for the AOS methods. Hyperheuristics [5] can be seen as generic online algorithm selection methods that typically make use of a portfolio of very simple algorithms.

This work is motivated by the objective to provide a rich generic algorithm selection framework for solving diverse problem instances of a given target optimization problem. More specifically, we focus our attention on memetic algorithms (MA) [6] that represent a generic evolutionary search technique for solving complex problems [7]. By interleaving global with local search, MA reaps the benefit of the global convergence of the stochastic global search method as well as the quick and precise convergence of the deterministic local search method thereby avoiding the local optimum trap of deterministic search technique and alleviating the slow, imprecise convergence of the stochastic search technique. Like other evolutionary algorithms, however, the efficacy of MA depends on the correct choice of operators and their parameters. Various evolutionary (i.e. crossover, mutation) operators lead to different solution qualities [8]. For constrained problems, the choice of ranking operator is also important [9]. Reference [10] focused on the frequency of the local search, or in other words, whether local search is needed or can be skipped, since it can be expensive computationally, and may cause difficulty in escaping from local optimality (especially when the population diversity is too low such that all individuals reside in the same basin of attraction). All the above works suggest that there is indeed a correlation between a problem instance and the MA configuration that can render efficacious search.

Rather than relying primarily on the personal expertise or simply employing the widely-used ones, automatic selection of the potentially efficacious operators makes MA not only more likely to yield superior performance, but also easier to use, even by those inexperienced users. In our context, an algorithm refers to one combination of evolutionary operators that need to be successively applied in each MA iteration. Dummy operator is introduced for each operator type to cater for the possibility of not using any operator of that type. As shown in Fig. 1, the algorithm portfolio is constructed offline via a series of operations which encompass *feature extraction*, *feature selection*, *algorithm clustering*, and *portfolio generation*. The resulting portfolio is then sent to an online selection mechanism that performs on-the-fly selection of combination of operators in each MA iteration. The efficacy of the proposed framework is then assessed empirically on quadratic assignment problem (QAP) and flowshop scheduling problem (FSP).

The contributions of the work presented in this paper is three-fold:

1. We propose OSCAR, a novel framework which takes the advantage of both the algorithm portfolio and online selection paradigms. To our knowledge, OSCAR is the first online selection of algorithms in a portfolio.
2. We generate problem-independent features for the construction of portfolio, thereby eliminating the necessity of problem domain expertise.
3. We provide a means of identifying reasonable number of sufficiently diverse combinations of operators for the evolutionary algorithm, such as the MA, allowing AOS to capture the correlation among different types of operators.

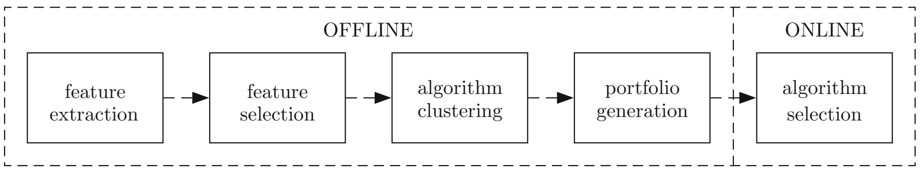

Fig. 1. Workflow of OSCAR

The remainder of the paper is presented as follows. Section 2 reviews related works in the literature. Section 3 introduces OSCAR and explains how it works in detail. Section 4 presents and discusses the experimental results on QAP and FSP. Finally, conclusion and future research directions are given in Sect. 5.

2 Related Works

Algorithm portfolios and (offline) selection have had a long history, and in the following, we review some recent works. SATZilla [11] is a well-known algorithm portfolio selection methodology that is particularly used to solve the SAT problem. It pursues a goal of providing a runtime prediction model for the SAT solvers. A number of problem-specific features for a given SAT instance are used to calculate the expected runtime of each algorithm in the portfolio. Its different versions are consistently ranked among the top portfolio-based solvers in the SAT competitions. 3S [12] utilised the resource constrained set covering problem with column generation to deliver solver schedules. Its superior performance was shown on the SAT domain. A cost-sensitive hierarchical clustering model was proposed in [13]. While the clustering model delivers a selection system, a static solver schedule is generated by 3S. SAT and MaxSAT were used as the application domains. Additionally, a Bayesian model combined with collaborative filtering is introduced to solve the constraint satisfaction and combinatorial auction problems in [14]. Unlike these studies, Hydra [15] addresses algorithm portfolios using parameter tuning. A portfolio is constructed by combining a particular solver with different parameter configurations provided by a version of ParamILS, i.e. FocusedILS [16]. The effectiveness of Hydra was also shown on SAT. Another tool developed for SAT, i.e. SATEnstein [17], targeted the algorithm generation process via tuning. It considers a variety of design elements for stochastic local search algorithms in the form of parameter tuning using ParamILS.

In terms of online algorithm selection, existing studies mostly refer to the terms Adaptive Operator Selection (AOS) [4] and Selection Hyper-heuristics [5]. The main idea is to monitor the search progress while solving a problem instance to immediately make changes on the choice of algorithms. Besides that, the online algorithm selection community deals with the algorithms and problems where solutions can be shared. However, in the case of offline methods, solution sharing can be cumbersome thus usually ignored when multiple algorithms are selected, like CPHydra [18]. Adaptive pursuit [19], multi-armed bandits [4] and reinforcement learning (RL) [20] are some successful examples of online selection.

3 OSCAR

Unlike most existing algorithm portfolio approaches that seek to deliver a portfolio of single solvers, this paper focuses on building a portfolio of algorithm combinations (even though our underlying approach can be used in the context of portfolio of single solvers). Each combination consists of a crossover operator, a mutational heuristic and a local search method. Our goal is to generate a small number of algorithm combinations with diverse performance that can successfully solve a large set of instances from a given problem domain. In order to have such a portfolio, it is initially required to generate a performance database revealing the *behavior* of each combination. Behavior here is denoted as the generic and problem-independent features primarily used in hyper-heuristic studies such as [21]. A class of hyper-heuristics, i.e. selection hyper-heuristics, aims at efficiently managing a given set of heuristics by selecting a heuristic(s) at each decision step. Due to the selection element in hyper-heuristics and their generic nature, we make use of the following features to characterize algorithm combinations for memetic algorithms.

- Number of new best solutions: N_{best}
- Number of improving solutions: N_{imp}
- Number of worsening solutions: N_{wrs}
- Number of equal quality solutions: N_{eql}
- Number of moves: N_{moves}
- Amount of improvement: $\triangle_{imp}$
- Amount of worsening: $\triangle_{wrs}$
- Total spent time: T.

A pseudo-code for OSCAR is presented in Algorithm 1. The process starts by collecting performance data regarding each algorithm combination a_x. The goal here is to perform a *feature extraction* about algorithms. For this purpose, each instance i_y is solved by a memetic algorithm successively using a randomly selected algorithm combination a_x. Algorithm 2 illustrates the basic memetic algorithm implementation. It should be noted that the performance data generation process differs for the cases where offline algorithm selection is applied. In the offline case, each algorithm is separately trained since these algorithms neither interact nor share solutions. Considering that an online selection device is employed and solutions are shared, it is vital to gather the performance data by running all the algorithms while they are selected online and operating on the same solutions.

The corresponding crossover (c_x), mutation (m_x) and local search (l_x) operators of a_x are applied in a relay fashion. The performance data generation process ends after each instance is solved within a given time limit (t_{limit}). The resulting performance data is used to generate features for each algorithm, $F(a_x)$.

Algorithm 1. OSCAR($\mathcal{A}$, $\mathcal{I}_{train}$, $\mathcal{I}_{test}$, $\mathcal{FS}$, $\mathcal{C}$, $\mathcal{OAS}$, $\mathcal{BC}$)

```
  Input : $\mathcal{A}$: an algorithm with multiple operators to choose from, $\mathcal{I}_{train}$: a set
          of training instances, $\mathcal{I}_{test}$: a set of test instances, $\mathcal{FS}$: a feature
          selection method, $\mathcal{C}$: a clustering algorithm, $\mathcal{OAS}$: an online algorithm
          selector, $\mathcal{BC}$: criterion for algorithm comparison
  Operator combination $a_x = c_x + m_x + l_x$ where $c_x$, $m_x$ and $l_x$ refer to crossover,
  mutation and local search operators respectively
  Performance vector for the algorithm combination $a_x$ on the instance $i_y$:
  $P(a_x, i_y) = \{p_1(a_x, i_y), \ldots, p_k(a_x, i_y)\}$
  Feature vector for the algorithm combination $a_x$:
  $F(a_x) = \{p_1(a_x, i_1), \ldots, p_k(a_x, i_m)\}$
  **Feature extraction**
1 $F \leftarrow P = A(.)$ algorithm $A$ is iteratively applied using randomly selected
  operator combinations $a_x$ to gather performance data $P$ for generating features
  $F$
  **Feature selection**
2 $F \leftarrow \mathcal{FS}(F)$
  **Algorithm clustering**
3 Cluster algorithm combinations: $\mathcal{C}(A, F)$
  **Portfolio generation**
4 Build portfolio using best algorithm combination from each cluster of $C$:
  $AP = \{cl_1 \rightarrow a, \ldots, cl_t \rightarrow a\}$ w.r.t. $\mathcal{BC}$
  **Online selection**
5 $S_{best} \leftarrow \mathcal{A}(AP, \mathcal{OAS}, \mathcal{I}_{test})$
```

Algorithm 2. MA(c, m, l)

```
  $n$: population size, $k$: number of newly produced individuals / solutions at each
  generation
1 Initialisation: Generate a population of solutions: $P(S_i)$ for $1 \leq i \leq n$
2 while $!stoppingCriteria()$ do
      k = 1
3     while $c \leq n_c$ do
4         Apply a crossover: $S_{n+k} = c(S_a, S_b)$
5         Apply a mutation method: $S_{n+k} = m(S_{n+c})$
6         Apply a local search operator: $S_{n+k} = l(S_{n+c})$
7         $k++$
      end
8     $updatePopulation(P)$
  end
```

Each feature vector is composed of the normalised versions of the following 7 features for each instance: $f_1 = N_{best}/T$, $f_2 = N_{imp}/T$, $f_3 = N_{wrs}/T$, $f_4 = N_{eql}/T$, $f_5 = \triangle_{imp}/T$, $f_6 = \triangle_{wrs}/T$ and $f_7 = T/N_{moves}$ As a result, each algorithm combination has $\#instances \times 7$ features.

After completing the feature extraction process, a *feature selection* or elimination [22] method is applied. Gini Importance[1] [23] and Gain Ratio[2] [24] were used for feature selection purpose. Gini Importance is mostly used with Random Forests to detect the effective features w.r.t. the given class information. Gain Ratio is a information theoretic measure used to detect the effect of each feature by checking the variations on the values of each feature.

Next, *algorithm clustering* is performed. k-means clustering is applied as the clustering method $\mathcal{C}$ to identify the (dis-)similarity of the algorithm combinations. The best performing algorithm combinations, one from each selected cluster compose the portfolio during the *portfolio generation* process. During this process, the clusters with operator combinations which couldn't find any new best solution are ignored. Of significant importance is that when a cluster manage to find some new best solution, that cluster must be part of the portfolio, no matter how small the cluster may be. Such small cluster may in fact be the special combination that works well only on some very specific problem instances. The best combination for each cluster are then determined w.r.t. $\mathcal{BC}$ which is the number of new best solutions found. The overall procedure is finalised by applying the corresponding memetic algorithm with a given online selection approach $\mathcal{OAS}$ to the test instances $\mathcal{I}_{test}$ during the *online selection* phase. For the experiments, uniform random selection is used as the $\mathcal{OAS}$ option.

4 Computational Results

For the memetic algorithm, the population size is set to 40. As many as 20 new individuals are generated during each generation. 4 crossovers, 1 mutation operator and 3 local search heuristics are the available memetic operators. Since the mutation operator needs a mutation rate to be set, 6 different values are considered: 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. Setting the mutation rate to zero actually means that the mutation operator is not used. In order to have the same effect for the other two operator types, we added one dummy crossover operator and one dummy local search heuristic. In total, 119 (5 crossovers $\times$ 6 mutations $\times$ 4 local search - 1[3]) operator combinations are generated. The details of these memetic operators are given as follows:

- Crossover:
  - *CYCLE* crossover: iteratively construct individuals by taking values from one parent and appointing the location of a next value from the second parent.
  - *DISTANCE_PRESERVING* crossover: outputs an individual where the distance referring to the number of genes assigned to different locations should be the same for the both parents.

[1] Using Scikit http://scikit-learn.org.
[2] Using Java-ML http://java-ml.sourceforge.net/.
[3] No crossover + no mutation + no local search case is ignored.

  - *ORDER* crossover: a subgroup of genes are taken from one parent and the remaining genes come from the second parent respecting their order.
  - *PARTIALLY_MAPPED* crossover: two randomly gene segments swap and partial maps denoting the elements located at common loci are used to change the conflicting genes with the swapped segment.
- Mutation: perturbs a given individual based on a mutation rate
- Local search:
  - *BEST_2_OPT* local search: attempts pairwise swap between 2 loci and applies the one producing best improvement in an iterative manner.
  - *FIRST_2_OPT* local search: attempts pairwise swap between 2 loci in a systematic fashion and applies the first one that produces improvement in an iterative manner.
  - *RANDOM_2_OPT* local search: attempts pairwise swap between 2 loci in a random order and applies the first one that produces improvement in an iterative manner.

For the training phase, t_{limit} is set to 300 s. The testing is performed with the per-instance execution time limit of 30 min for 5 trials. Java on an Intel Core I5 2300 CPU @ 2.80 GHz PC is used for the experiments.

4.1 Quadratic Assignment Problem

The QAP [25] requires the assignment of n facilities to n locations. Equation 1 shows the objective to minimise for the QAP. $f_{\pi_i \pi_j}$ is the flow between the facilities π_i and π_j. π refers to a solution where each element is a facility and the locus of each facility shows its location. d_{ij} is the distance between the location i and j. The objective is to minimise the total distance weighted by the flow values.

$$min \sum_{i}^{n} \sum_{j}^{n} f_{\pi_i \pi_j} d_{ij} \tag{1}$$

60 QAP instances from QAPLIB [26] were used. 31 instances are selected for training such that we can have enough performance data for each algorithm combination within the aforementioned time limit.

Portfolio Generation. The feature generation process resulted in 217 (31 instances $\times$ 7 per instance features) features. The features calculated for each operator combination on each instance is discarded if the number of moves performed is less than 10. After eliminating such features, 182 (26 instances $\times$ 7 per instance features) are left for each operator combination. Next, k-means was called with $k = 5$ to detect clusters of operator combinations. The features with this cluster information was considered as a classification problem in order to understand the nature of clusters. For this purpose, a random forests based feature importance evaluation method, i.e. Gini importance [23], is applied.

It revealed that 27 out of 182 features are the ones actually shaping these clusters. In addition, the features $f_1 = N_{best}/T$ and $f_2 = N_{imp}/T$ are from these 27 features for most of the QAP instances.

Besides using these 27 features, the same number of features are taken from the most critical features determined by other feature importance metrics. Table 1 lists the algorithm combination portfolios found using different feature sets provided by the metrics. The general view of these portfolios suggest that it is not always a good idea to keep applying all the three types of memetic operators together. Thus, in certain operator combinations, one or two operator types are missing. DISTANCE_PRESERVING and PARTIALLY_MAPPED crossovers are not included any of the operator combinations of the derived portfolios. Mutation is either ignored or applied with a small rate, i.e. 0.2 and 0.4. Among the local search heuristic, FIRST_2_OPT is detected as the most popular local search method while BEST_2_OPT is never picked. Besides, the portfolio sizes vary between 3 and 4. Considering that $k = 5$, 1 or 2 clusters have no operator combination yielded new best solutions during the training phase. In order to show whether using multiple operator combinations in an online setting is useful, the single best combination is also detected. The single best for the QAP uses CYCLE crossover and FIRST_2_OPT without mutation.

Table 1. Operator combination portfolios determined by OSCAR for the QAP

Feature selection	Algorithm portfolios		
	Crossover	Mutation	Local search
No selection	CYCLE	–	FIRST_2_OPT
	CYCLE	–	RANDOM_2_OPT
	ORDER	0.4	FIRST_2_OPT
	CYCLE	0.2	FIRST_2_OPT
Gini importance	CYCLE	–	FIRST_2_OPT
	CYCLE	–	RANDOM_2_OPT
	–	–	FIRST_2_OPT
Gain ratio	CYCLE	–	FIRST_2_OPT
	CYCLE	–	RANDOM_2_OPT
	–	–	FIRST_2_OPT
	CYCLE	0.2	FIRST_2_OPT

Figure 2 visualises the operator combinations for each operator type to determine what actually shapes these clusters via multidimensional scaling (MDS) [27] with Euclidean distance. These graphs indicate that the operator combinations are grouped particularly in reference to the local search operators. Figure 3 shows the effect of individual performance measures on clustering. The amount of improvement and worsening w.r.t. the total time spent by each operator combination is utilised as the most critical performance measures. The operator combinations' speed, the number of new best solutions and equal quality solutions detected w.r.t. the total time spent by each operator combination are determined as the measures affecting clusters least.

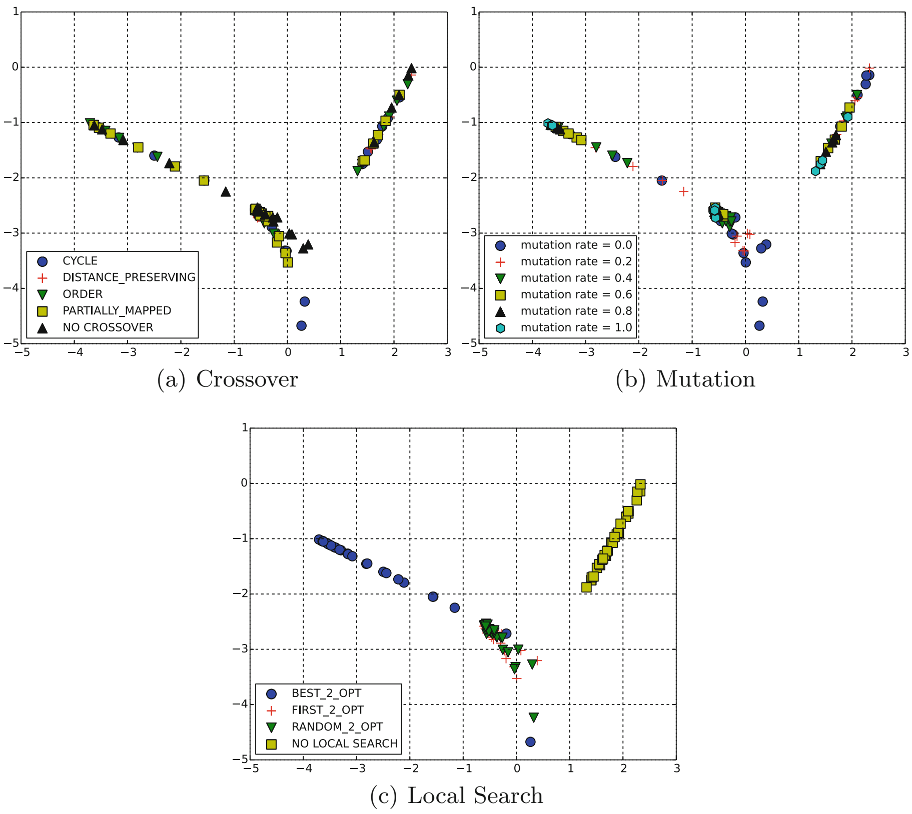

Fig. 2. MDS of operator combinations w.r.t. each operator type for the QAP

Online Algorithm Selection. Figure 4(a) shows the performance of three portfolios together with the Single Best combination when Random is used as online selector, in terms of the success rate (i.e. how many times the best known or optimum solutions are found, expressed in percentage). The results indicate that the single best is able to deliver around 23 % of the best known QAP solutions while OSCAR with different portfolios can find between 36 % and 37 % of the best known solutions. Although Gini and Gain Ratio based portfolios perform slightly better than the case without feature selection, there seems to be of only slight difference. However, when we look at the results closely by considering the solution quality, the performance difference becomes clearer. Figure 4(b) presents box plots indicating the ranks of each tested method. Besides the superior performance of OSCAR against the Single Best in ranking, the portfolio constructed using Gini delivers the best results among the three portfolios.

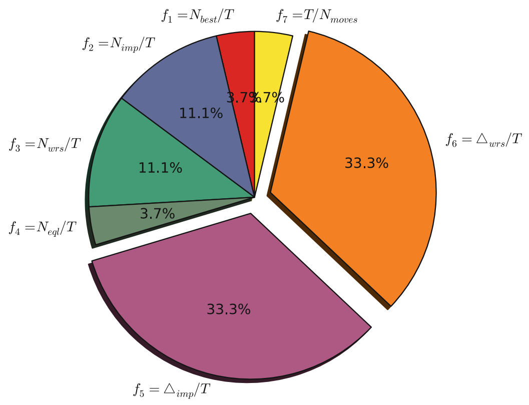

Fig. 3. Contribution of the 7 problem-independent performance measures to the top QAP features, determined by Gini

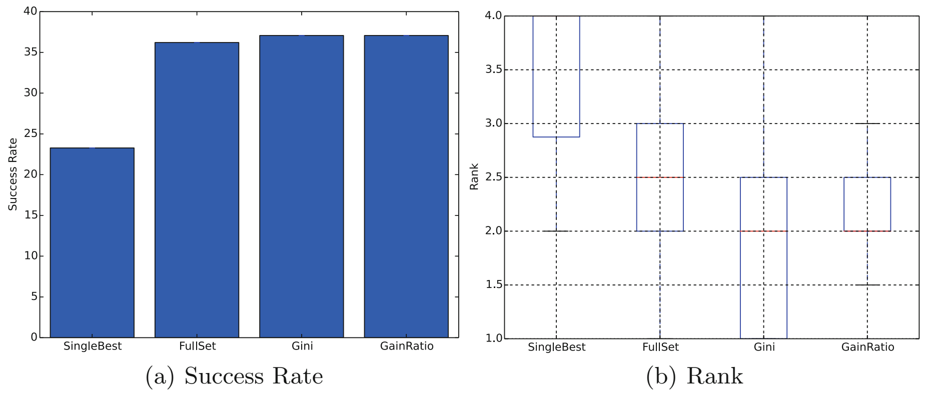

Fig. 4. Success rates and ranks of operator combination portfolios on the QAP

4.2 Flowshop Scheduling Problem

The Flowshop Scheduling Problem (FSP) is related to the assignment of n jobs to m machines aiming at minimizing the completion time of the last job, i,e. the makespan. The 68 FSP instances from the Taillard FSP benchmarks[4] [28] are used. 41 of these instances are taken as the training instances while the remaining 27 instances are considered as the test set.

Portfolio Generation. The feature generation process provided 287 features (41 instances $\times$ 7 per instance features) for each instance. After performing

[4] http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html.

k-means clustering with $k = 5$, the Gini importance metric calculated via applying Random Forests indicated that only 29 of these 287 features contributed to the clustering process. Thus, we use 29 as the number of top features to check. This is achieved using the aforementioned importance metrics as we did for the QAP case. Table 2 lists the portfolios of operator combinations derived using each of these importance metrics. Unlike the QAP case, DISTANCE_PRESERVING and PARTIALLY_MAPPED crossovers are also used in the FSP portfolios. For Mutation, higher rates are preferred, i.e. 0.6 and 0.8, or no mutation is applied. RANDOM_2_OPT, here, is as frequently picked as FIRST_2_OPT and BEST_2_OPT is used in one operator combination where DISTANCE_PRESERVING is included. Similar to the QAP portfolios, here each portfolio has either 3 or 4 operator combinations. The single best combination for the FSP applies PARTIALLY_MAPPED crossover, mutation with rate of 0.6 and RANDOM_2_OPT.

Table 2. Operator combination portfolios determined by OSCAR for the FSP

Feature selection	Algorithm portfolios		
	Crossover	Mutation	Local search
No selection	CYCLE	–	FIRST_2_OPT
	CYCLE	–	RANDOM_2_OPT
	DISTANCE_PRESERVING	0.6	BEST_2_OPT
	PARTIALLY_MAPPED	0.6	RANDOM_2_OPT
Gini importance	CYCLE	-	FIRST_2_OPT
	CYCLE	–	RANDOM_2_OPT
	PARTIALLY_MAPPED	0.6	RANDOM_2_OPT
	ORDER	–	FIRST_2_OPT
Gain ratio	PARTIALLY_MAPPED	0.6	RANDOM_2_OPT
	–	0.8	FIRST_2_OPT
	ORDER	–	FIRST_2_OPT

Figure 5 presents the operator combinations w.r.t. their problem-independent features in 2D via MDS. As with the QAP, the local search operators mainly characterise the operator combinations' groups. Figure 6 shows the which individual performance measure is used while clustering. Operator combinations' speed is detected as the major factor. Additionally, the number of new best solutions, worsening solutions and equal quality solutions w.r.t. the total time spent by each operator combination are also highly effective on the clusters. The amount of worsening w.r.t. the total time spent by each operator combination is utilised as the least important performance measure.

Online Algorithm Selection. Figure 7(a) details the performance of 3 portfolios and the single best combination in terms of success rate (i.e. how many

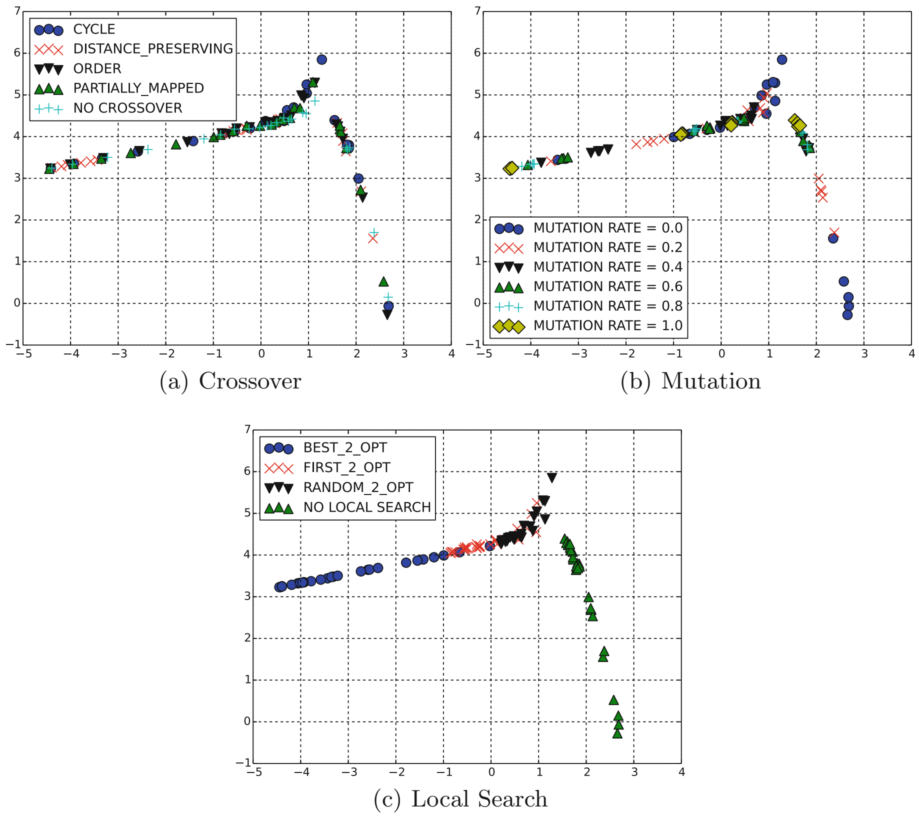

Fig. 5. MDS of operator combinations w.r.t. each operator type for the FSP

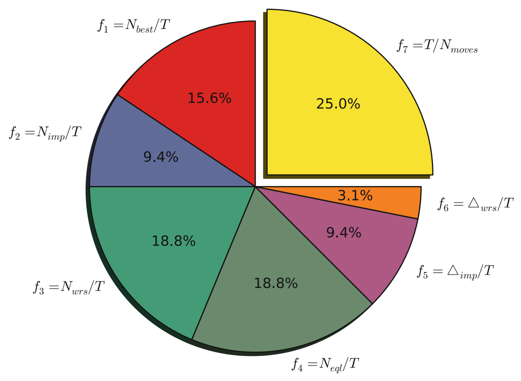

Fig. 6. Contribution of the 7 problem-independent performance measures to the top FSP features, determined by Gini

times the best known or optimal FSP solutions are found, expressed in percentage). The portfolios generated using full feature set and Gain Ratio show similar performance to the single best combination by reaching between 47 % and 49 % of the best known or optimum solutions. However, the portfolio with Gini found around 56 % of the best known solutions as the best tested method. Figure 7(b) presents these results in terms of ranks w.r.t. the solution quality where OSCAR's superior performance can be clearly seen. Among the reported portfolios, the Gini based portfolio reveals the statistically significant best results.

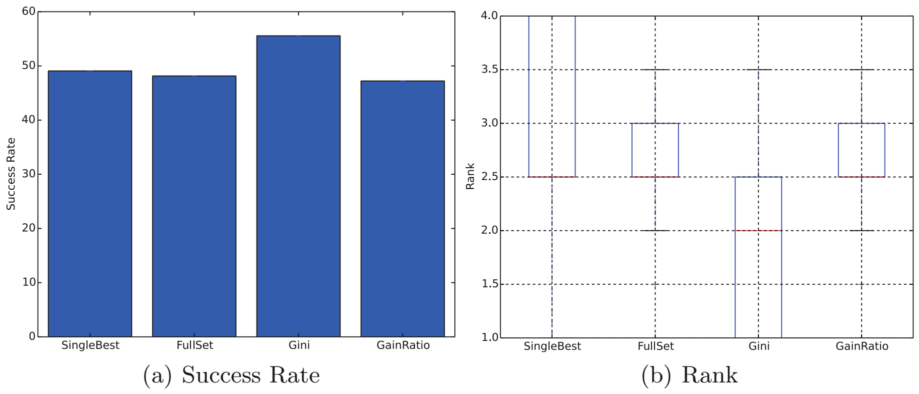

Fig. 7. Success rates and ranks of operator combination portfolios on the FSP

Overall, the results on both the QAP and the FSP indicate that using multiple operator combinations is profitable when they are selected online. This shows that OSCAR is able to combine the strengths of both offline algorithm portfolios and online algorithm selection in a problem-independent manner. Of particular significance is that the Gini-based portfolio always perform the best.

5 Conclusions

In this paper, we have introduced OSCAR as a framework that performs Online SeleCtion of Algorithm poRtfolio. The algorithm portfolio is constructed offline to determine which combinations of the memetic operators are efficacious for solving certain problem domains. Those combinations in the portfolio are then fetched to some online selection mechanism. This hybridization allows an online selection method to capture the correlation among different types of the memetic operators. This paper presents the first study of such hybridization. Additionally, OSCAR does not require any problem-specific features to generate the portfolio, thereby eliminating the necessity of problem domain expertise.

Empirical assessments on QAP and FSP have demonstrated the efficacy of OSCAR. OSCAR is able to deliver superior performance compared to the single best operator combinations for both problems. This shows that the problem-independent features introduced are practical to differentiate one available operator combination from the others, which eventually lead to an efficient portfolio. Furthermore, the improving performance delivered after feature selection, particularly when Gini importance index is employed, indicates the usefulness of the feature selection part of OSCAR.

Moving forward, the explanatory landscape analysis [29] will be incorporated to extend the algorithm feature space. The multi-objective performance measures shall be studied to build portfolios for multi-objective evolutionary algorithms. An in-depth analysis will be performed to evaluate the performance of different clustering techniques and online selection methods.

References

1. Rice, J.: The algorithm selection problem. Adv. comput. **15**, 65–118 (1976)
2. Gomes, C., Selman, B.: Algorithm portfolio design: theory vs. practice. In: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI 1997), Providence/Rhode Island, USA, pp. 190–197 (1997)
3. Huberman, B., Lukose, R., Hogg, T.: An economics approach to hard computational problems. Science **275**, 51 (1997)
4. Da Costa, L., Fialho, A., Schoenauer, M., Sebag, M., et al.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2008), Atlanta, Georgia, USA, pp. 913–920 (2008)
5. Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**, 1695–1724 (2013)
6. Moscato, P., Cotta, C., Mendes, A.: Memetic algorithms. In: Moscato, P., Cotta, C., Mendes, A. (eds.) New Optimization Techniques in Engineering, pp. 53–85. Springer, Heidelberg (2004)
7. Krasnogor, N., Smith, J.: A memetic algorithm with self-adaptive local search: TSP as a case study. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2000), Las Vegas/Nevada, USA, pp. 987–994 (2000)
8. Yuan, Z., Handoko, S.D., Nguyen, D.T., Lau, H.C.: An empirical study of off-line configuration and on-line adaptation in operator selection. In: Pardalos, P.M., Resende, M.G.C., Vogiatzis, C., Walteros, J.L. (eds.) LION 2014. LNCS, vol. 8426, pp. 62–76. Springer International Publishing, Switzerland (2014)
9. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **4**, 284–294 (2000)
10. Handoko, S.D., Kwoh, C.K., Ong, Y.S.: Feasibility structure modeling: an effective chaperone for constrained memetic algorithms. IEEE Trans. Evol. Comput. **14**, 740–758 (2010)
11. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: SATzilla: portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. **32**, 565–606 (2008)
12. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 454–469. Springer, Heidelberg (2011)

13. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm portfolios based on cost-sensitive hierarchical clustering. In: Proceedings of the 23rd International Joint Conference on Artifical Intelligence (IJCAI 2013), pp. 608–614 (2013)
14. Stern, D., Herbrich, R., Graepel, T., Samulowitz, H., Pulina, L., Tacchella, A.: Collaborative expert portfolio management. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), Atlanta/Georgia, USA, pp. 179–184 (2010)
15. Xu, L., Hoos, H., Leyton-Brown, K.: Hydra: automatically configuring algorithms for portfolio-based selection. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 210–216 (2010)
16. Hutter, F., Hoos, H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. J. Artif. Intell. Res. **36**, 267–306 (2009)
17. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: Satenstein: automatically building local search sat solvers from components. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), vol. 9, pp. 517–524 (2009)
18. O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., O'Sullivan, B.: Using case-based reasoning in an algorithm portfolio for constraint solving. In: Irish Conference on Artificial Intelligence and Cognitive Science (2008)
19. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO 2005), pp. 1539–1546. ACM (2005)
20. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: Resende, M., de Sousa, J. (eds.) Metaheuristics: Computer Decision-Making, pp. 523–544. Kluwer Academic Publishers, Dordrecht (2003)
21. Mısır, M.: Intelligent hyper-heuristics: a tool for solving generic optimisation problems. Ph.D. thesis, Department of Computer Science, KU Leuven (2012)
22. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
23. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
24. Quinlan, J.R.: C4.5. Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
25. Lawler, E.: The quadratic assignment problem. Manag. Sci. **9**, 586–599 (1963)
26. Burkard, R.E., Karisch, S.E., Rendl, F.: Qaplib-a quadratic assignment problem library. J. Global Optim. **10**, 391–403 (1997)
27. Borg, I., Groenen, P.J.: Modern Multidimensional Scaling: Theory and Applications. Springer, New York (2005)
28. Taillard, E.: Benchmarks for basic scheduling problems. Eur. J. Oper. Res. **64**, 278–285 (1993)
29. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 829–836. ACM (2011)