Evolutionary Scheduling and Combinatorial Optimisation: Applications, Challenges, and Future Directions

Su Nguyen*†, Yi Mei*, Hui Ma*, Aaron Chen*, Mengjie Zhang*
*Evolutionary Computation Research Group, Victoria University of Wellington, New Zealand
†Hoa Sen University, Ho Chi Minh City, Vietnam
Email: su.nguyenphanbach@hoasen.edu.vn
{yi.mei, hui.ma, aaron.chen, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—Evolutionary scheduling and combinatorial optimisation is an active research area and attracts the attentions of many researchers from computer science and operations research. Many advances have been made in this field and its scope in terms of techniques and applications has been continuously extended. In this position paper, we provide an overall picture of some key challenges in the field, discuss potential future research directions, and give our position in the field. We focus on three major issues that are encountered in practice, namely dynamic changes, multiple interdependent decisions, and multiple objective. Our view is that the researchers should step out of our comfort zone to deal with messy and complicated issues in real-world applications.**

I. Introduction

Scheduling and combinatorial optimisation problems appears in many practical applications in production and service industries and have been the research interest of researchers from operations research and computer science. Solving these optimisation problems can help production managers reduce inventory and leadtime, and increase customer satisfaction and profit. For service industries, solving these problems can improve resource usage and increase revenues while both customers and employees needs are fulfilled. New problems such as scheduling in cloud, scheduling of big-data jobs, scheduling for electric vehicle charging, and drone-assisted parcel delivery, emerge with the growth of new technologies and business models. Researchers in this field have to continuously face with new challenges; as the results, innovative solution methods have been developed.

Evolutionary computation (EC) refers to a range of computational methods that are often inspired by processes that occur in nature. Examples of evolutionary methods are genetic algorithms, genetic programming, ant colony systems, particle swarm optimisation, scatter search and path relinking, memetic algorithms, artificial immune systems, evolutionary strategies, cultural algorithms, etc. Scheduling and combinatorial optimisation problems are the earliest applications of EC for optimisation. Today, evolutionary scheduling and combinatorial optimisation (ESCO) is an important research area at the interface of artificial intelligence (AI) and operations research (OR). In the literature, it is easy to see that almost all new EC methods are tested on scheduling and combinatorial optimisation problems to demonstrate their effectiveness and efficiency. Moreover, challenges in these problems have also motivated EC researchers to develop more advanced techniques. Researchers have established IEEE Task Force on Evolutionary Scheduling and Combinatorial Optimization[1], and various special sessions in major EC conferences such as IEEE CEC, SSCI, ACM GECCO, SEAL conference, and special issue in combinatorial optimisation (Evolutionary Computation Journal). It is safe to say that ESCO is one of the most important research topics in EC. Different from tradition optimisation methods (e.g. branch-and-bound, dynamic programming), EC methods aim at finding near-optimal solutions for the problem of interest in reasonable running time. The strength of EC is its flexibility in solution representations and search mechanisms, which allows it to cope with complex problems and integrate with other algorithms (from both AI and OR). For examples, techniques in graph theories and mathematical programming are usually employed within EC methods to improve their effectiveness and efficiency for many applications (e.g. job shop scheduling, vehicle routing). Currently, EC algorithms have been implemented in most solvers used for scheduling and combinatorial optimisation (e.g. IBM ILOG solver, Frontline solvers).

Research in ESCO is very diverse ranging from simple applications of EC algorithms to the optimisation problems to hybrid approaches which combine the advantages of different solution methods. In most cases, especially when dealing with real-world applications, EC methods can be used if exact optimisation methods failed to solve the problem efficiently, i.e. only optimal solutions for small problem instances are found. Three key components of EC methods are (1) representations of solutions, (2) the fitness function, and (3) search operators. Choosing a suitable representation is crucial for any EC method because representations decide the search space of the algorithms, the search operators (e.g. crossover, mutation in GA), and the efficiency of solution evaluations. Fitness functions help EC methods determine the quality of the obtained solutions and guide their search toward potential region. For evolutionary scheduling, search operators also play an important role because they decide how new solutions are generated. Although the search operators are very different for each EC method, they have to be designed to ensure the feasibility of generated solutions and create a balance between the exploration and exploitation abilities in EC. For

[1]https://escotf.wordpress.com/

scheduling, using pure EC methods usually does not provide satisfactory results. Instead, hybrid methods are preferred in these cases to enhance effectiveness and efficiency of EC. Usually, these methods try to incorporate problem domain knowledge or utilise historical search data via local search, decomposition, coevolution, and machine learning.

The goal of this paper is to describe the current status, challenges, future research directions for ESCO. We will emphasise on three major issues in scheduling and combinatorial problems, namely dynamic changes, multiple decisions, and multiple conflicting objectives. The discussions will reflect these issues in the context of three important applications in the field: (1) production scheduling, (2) routing, and (3) Web/cloud scheduling and innovative solution methods developed in the literature. Our discussions are mainly around scheduling problems but they can be easily related to other general combinatorial optimisation problems. Then, we highlight the corresponding challenges and recommend potential future directions.

II. Dynamic changes

Majority of studies in ESCO focus on static problems where all information is available at the beginning. For examples, number of jobs and their information (e.g. processing times, due dates, weights) are known in advanced for static job shop scheduling problems. Unfortunately, it is not usually the case in practice as jobs may arrive at the production systems randomly over time (e.g. make-to-order companies) and their information may only be available upon their arrivals. Similarly, for cloud scheduling, application schedulers must react promptly to dynamical and unpredictable changes in the cloud. For vehicle routing problems, new costumer requests can be received while the trucks are on their way of delivering the products. It is necessary to respond to such dynamic changes adaptively to improve the quality of service and satisfaction of customers. Dynamic changes are unavoidable in the real-world applications and coping with this issue is an important factor to make EC applicable.

A. Existing studies

Solution methods to cope with dynamic changes can be classified based on their reactiveness, i.e. how fast they can response to the changes. In production scheduling, rescheduling (reoptimisation) is the most popular approach to deal with dynamic changes (can be executed periodically or continuously). Bierwirth and Mattfeld [1] developed a genetic algorithm for scheduling and rescheduling based on rolling time basis where the initial population of GA includes information from good solutions found in the previous run (i.e. the previous rescheduling step). Fang and Xi [2] developed an interesting method that combines GA and a dispatching rule to deal with dynamic flexible manufacturing system. Branke and Mattfeld [3] proposed an evolutionary algorithm to minimize the mean tardiness for job shops with anticipation of changes (penalize early idle times). The experiments showed that the proposed methods are very effective as compared to other meta-heuristics and dispatching rules. Analyses suggested that short-term efficiency losses can improve the overall performance in a dynamic environment. Shen and Yao [4] proposed a new multi-objective evolutionary algorithm (MOEA)-based proactive or reactive method to deal with dynamic flexible job shop scheduling. Both random job arrivals and machine breakdowns are considered in this work. Rescheduling is triggered when critical events occur. The initial population for recheduling is generated by using specialized heuristics to improve the convergence speed of the proposed method.

Another potential approach to deal with dynamic changes is dispatching rules as they are computationally efficient and can react quickly to dynamic changes in the shop floor. Normally a dispatching rule is characterised by a priority function that determines priorities of jobs waiting in the queue (the job with highest priority will be processed next). In recent years, genetic programming (GP) and its variants (such as gene expression programming and grammar-based genetic programming) have been applied for automated design of dispatching rules [5]–[7]. In these studies, variable-length representations (e.g. tree-based) of GP is used to evolve dispatching rules. The experimental results in the literature have shown that evolved dispatching rules are very competitive as compared to other rules and heuristics previously proposed in the literature.

The existing work to solve dynamic vehicle routing problem (VRP) can be divided into two categories: *periodic reoptimisation* and *continuous reoptimisation*. As the name indicates, periodic reoptimisation reoptimizes the problem periodically. The two main approaches are mathematical programming [8], [9] and real-time policy [10], [11]. However, they both have drawbacks. The mathematical programming approaches are usually slow, and not efficient enough for real-time response. The real-time policy, on the other hand, is normally manually designed. It is unknown whether such a design is good or not. The continuous reoptimisation method adopts a framework in which the dispatching office keeps optimising the problem under the latest environment, and communicates with the drivers to give them the next destination whenever they reach a service location. Useful information such as the best routes are stored in an adaptive memory to speed up the reoptimisation when the environment changes. Representative works include [12], [13]. These methods are based on modifying some existing solutions locally to adapt to the environmental changes. Thus, it is challenging to decide the stored solutions which are both of good quality and robust for local modification (quality does not change much). For arc routing problem, which is the arc-counterpart of VRP, the dynamic environment has been ignored so far. The closest model is the stochastic model, in which the actual information (travel time and demand) is uncertain, and can be different from the expectation. Fleury et al. [14] proposed evolutionary algorithms to improve the robustness of solutions under uncertain environment. In [15], [16], we gave a formal definition of the robustness optimisation in the stochastic environment, and provided an instance generator that can produce stochastic benchmark instances from a static one. We also tested the state-of-the-art algorithms

on the stochastic benchmark instances to show its extra challenge. Wang et al. proposed a memetic algorithm [17] and an estimation of distribution algorithm [18] for solving the problem.

B. Research challenges and future directions

Planning and scheduling in dynamic environments are always challenging. They are even more challenging with the existing of stochastic events (e.g. random arrivals, machine breakdowns). Given that most real systems are dynamic, dynamic changes are unavoidable and should be taken into account when designing algorithms or heuristics. In production environment, causes of dynamic changes can be classified into two categories [19]: (1) *resource-related* (e.g. machine breakdown, operator illness, unavailability or tool failures) and (2) *job-related* (e.g. rush jobs, job cancellation, changes in job processing time). Many methods have been proposed in the scheduling literature to deal with these issues (most of them only focus on one type of event). Strategies for dynamic scheduling in production systems can be classified as:

- *Completely reactive scheduling*: decisions are made locally as needed; for example, priority dispatching rules.
- *Predictive-reactive scheduling*: scheduling/rescheduling is triggered by the real-time events where both objectives of interest and stability (measured by the deviation from original schedule) are considered.
- *Robust pro-active scheduling*: focus on building predictable schedules; the key idea is to improve the predictability of the schedules (e.g. by inserting additional time in the predictive schedule) with minimal effects on the schedule performance.

In a loose sense, these categories can also be applied for other scheduling and combinatorial optimisation problems. In dynamic environments, many issues need to be taken into account to ensure that generated solutions are usable. EC methods have been proposed for the three strategies above [4], [20], [21]. However, these studies are still very limited. EC researchers mainly focus on algorithmic aspects and other practical aspects have attracted less attention. In summary, we need to tackle three main challenges related to dynamic changes:

- Improve stability and predictability of schedules
- Handle different sources of disturbances
- Improve the efficiency

For production scheduling, vehicle routing, and cloud scheduling problems, stability is important for planning purposes. Clearly, operators in the shop floor do not want to change the schedules at their station rapidly. Similarly, the truck drivers do not want to change the route continuously when they are on the road. In order to improve the stability of schedules, the deviations from original (previous schedules) must be measured and EC will try to minimise these deviations when searching for new schedules. One advantage of EC is that original schedules can be used to intialise the population of EC methods to reduce the computational time and maintain the consistency between the original and new schedules. However, it is likely that EC will stuck in the local optima. Also, we want to improve the predictability of our solutions (e.g. delivery times are accurate enough so that customers do not have to wait). Unfortunately, there are not many studies on this predictability of schedules generated by EC [21]. Multi-objective optimisation is a promising approach to deal with both stability and predictability. Future studies need to focus on these aspects to make EC solution more appealing for practitioners.

Different sources of disturbances make the optimisation more challenging. Past studies mainly focused on one or two types of disturbances and ignored the others. A study on the disturbance frequencies and their effects on evolutionary scheduling is still missing in the literature. To cope with this problem, the evolutionary scheduling system needs to be flexible enough to accommodate different strategies to deal with different types of disturbances. Without these features the users will have to make manual adjustments. To this point, we expect that evolutionary scheduling systems will play the role of an optimisation system but also a recommendation system which suggests useful schedules for the users.

Efficiency is important for a scheduling system. While EC is a good approach for scheduling, it still has scalability problems. From their experiments with production scheduling, Bierwirth and Mattfeld [1] reported that the performance of GAs deteriorates as the problem size increases, and they have trouble finding near-optimal solutions in reasonable times. For cloud scheduling, EC-based schedulers are relatively slow in speed as compare to policy-based methods and cannot quickly cope with uncertainties. Scheduling of application service composition in clouds should be able to adapt to environmental changes without compromising operational and financial efficiencies. To deal with scalability problem in dynamic environment, obtained schedules from previous runs of EC methods can be used (partly) as the initial solutions in the next run to reduce the computational costs. Decomposition techniques can be employed to improve the efficiency and effectiveness of EC; however, these techniques need to take into account stability, predictability, and sources of disturbances as discussed above. Automated design of dispatching rules (or hyper-heuristics [5]) is also a powerful approach to deal with efficiency challenges. Because the design part is performed offline, it does not influence the efficiency of evolved dispatching rules. One of the drawbacks of dispatching rules is that no schedule is generated, which causes difficulty for planning in advance. A hybrid learning-and-optimising method [22] can be a potential way to take advantages of automated design of dispatching rules and EC for scheduling/rescheduling. The idea is to create an archive of effective rules (e.g. using genetic programming), and use the schedules generated by these rules as the initial solutions for EC in the (re)optimisation stage.

III. Multiple decisions

For about 30 years (since 1960s), operations research mainly focused on idealised problems and lost its empirical

foundations [23]. These problems were initially created for teaching purposes by simplifying real-world problems. They were just partial models of problems that we may encounter in practice because many aspects are not considered if they are not related to the solutions methods or techniques. It seems that evolutionary scheduling may also run into this trap because many studies have been mainly focusing on algorithmic aspects and forgotten the real context that obtained solutions will be applied to. Bonyadi et al. [24] shared this view in their paper and pointed out that there are growing gaps between research and practice in the field of meta-heuristics. They argued that real-world problems usually consist of two or more *sub-problems* that are interdependent (to each other) and the *complexity* of real-world problems is not only limited to the size of the problem but also on its *interdependence*.

> *"An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem." – John Tukey (1915–2000)*

Studies in evolutionary scheduling have mostly emphasised on scheduling or sequencing decisions and tend to ignore or simplify assumptions about other aspects or related decisions. For example, production planning and control involves many decisions such as order acceptance, due date assignment, order release, routing, and output control (e.g. outsourcing, overtime). These decisions can directly or indirectly influence scheduling decisions; however, they are barely considered in evolutionary scheduling studies. Similarly, when dealing with vehicle routing problems, we have to consider other aspects such as packing [25], warehousing (e.g. cross-docking [26]), and crew scheduling [27]. In cloud scheduling, the service allocation problem is interrelated to workflow scheduling problem and we need to handle both of them simultaneously.

A. Existing studies

Research on EC based solution methods to deal with multiple decisions in scheduling problems is still in an early stage. In general, they can be divided into two main categories: (1) *all-in-one* and (2) *divide-and-conquer*.

The first one focuses on generating sophisticated representations that accommodate all related decisions and EC is used (in combination with some local search heuristics) to explore the large search space. For instances, the operation sequence vector and the machine assignment vector are used in flexible job shop scheduling problems (FJSSP) to represent a complete schedule [4], [28] and specialised search operators are developed to deal with each part of the solution. More sophisticated representations such as machine order with bits, operation-based order, and operation order with bits are also investigated by Tay and Wibowo [29]. Unachak and Goodman [30] developed an adaptive representation for FJSSP that consists of two parts: (1) routing policy to govern how an operation will choose a machine and (2) scheduling policy to make scheduling decisions based on system status. In some cases, representations only focus on one decisions and other decisions are determined by some heuristics. For order acceptance and scheduling problems, solutions can be presented by sequence of orders to be processed and acceptance decisions are determined by the construction heuristics [22]. Similarly, for vehicle routing problems (VRP) with crew scheduling [27], traditional representations of VRP can be used and personnel assignment heuristics are applied to construct the complete solutions.

Another natural way to deal with multiple interdependent decisions is divide-and-conquer. The key idea is to reduce the complexity of the original problem by solving each sub-problem and develop strategies to link the obtained partial solutions together. For example, Kim et al. [31] developed a symbiotic (cooperative) evolutionary algorithm for the integration of process planning and job shop scheduling. In their algorithm, each of the sub-problems is treated as a distinct species, and a sub-population is dedicated for each of the species. Therefore, an individual in a sub-population represents a partial solution to the entire problem. This kind of decomposition techniques have been applied in many scheduling applications related to multiple decisions [32]–[34]. For Travelling Thief Problem (TTP), Mei et al. [35] have investigated the collaborative issues in cooperative co-evolution and proposed new approaches that coordinate the optimisation of two sub-components in an efficient way [35], [36]. Ai et al. [37] developed a cooperative coevolution GA to deal with the deadline-constrained resource allocation and scheduling problem for multiple composite Web services. In their paper, problems are decomposed based on the number of composite Web services.

B. Research challenges and future directions

Dealing with multiple interdependent decisions is always difficult but also create opportunities to find the *real* global optimal solution. To attain this ultimate goal, a number of challenges need to be overcome. The discussion of managerial issues related to multiple decisions such as handling conflicts of interest from different groups of decision makers is beyond the scope of this paper. Here we only point out two challenges that are most relevant to evolutionary scheduling:

- Handling multiple decisions in dynamic environments
- Efficiency and effectiveness

Most existing studies handle multiple interdependent decisions in a static environment. However, as discussed in the previous section, we have to deal with dynamic changes in practice. This is particularly challenging because of a number of reasons. First, reoptimisation may be triggered more frequently when dynamic changes are expected to influence any decision under consideration, especially when multiple decisions need to be made at the same time. In this case, maintaining the stability and predictability of solutions are much more difficult. When multiple decisions are made at different points in time (e.g. due date assignment and sequencing), applying EC for optimisation is not straightforward. Using periodic reoptimisation is a possible solution but it may also reduce the reactiveness of the scheduling system. One way to handle these issues is to apply automated heuristics design (e.g. via genetic programming) to evolve reactive heuristics

for each decision. For example, Nguyen et al. [32] developed a cooperative coevolution genetic programming method to evolve both due date assignment rules and dispatching rules. Similarly, Park et al. [38] developed different representations for GP to simultaneously handle both order acceptance and scheduling decision in a single machine environment. With the same spirit, Mei et al. [36] used genetic programming to evolve a gain function and a picking function to enhance the performance of the two-stage memetic algorithm for TTP. However, as discussed previously, no concrete plan/solution is constructed when using evolved rules as decisions are made at the latest moments. In the future studies, it would be interesting to investigate more advanced EC methods that take into account the stability of obtained solutions and how they can be combined with reactive rules (possibly evolved by genetic programming), which can be used to generate contingency plans.

Multiple interdependent decisions influence the complexity of the problem; therefore, it can directly influence the effectiveness and efficiency of EC methods. For example, the two sub-problems in TTP, i.e. knapsack and traveling salesman, are NP-hard on their own right; thus solving both of them simultaneously is obviously very challenging. Using pure EC methods in these situations is inefficient because of high computational costs (mostly for fitness evaluations). Commonly, efficient solution methods can be used to handle one or more decisions and other decisions will be handled by EC. For example, Lin-Kernighan [39] algorithms can be used to find good tours for TTP and EC can be used to deal with the knapsack sub-problem. Cooperative coveolution such as the ones proposed by Kim et al. [31] and Mei et al. [35] are also promising approaches to improve the efficiency of EC methods. Hybrids methods between EC, OR, AI, and machine learning techniques will be the key to handle complex scheduling problems with multiple interdependent decisions.

IV. Multiple objectives

In practice, there are usually multiple conflicting objectives, which require a set of *non-dominated* trade-off solutions instead of a single best solution. For example, in production scheduling, the total flowtime and maximal flowtime are conflicting. In vehicle routing, the total cost and number of vehicles are conflicting. In cloud computing, the cost, response time and throughput are often conflicting. As compared to traditional multi-objective optimisation problems, multi-objective evolutionary scheduling has some special characteristics. First, the calculation of most multi-objective performance measures (e.g. IGD, the ϵ indicator) rely on the true Pareto front. However, in real-world scheduling problems, the true Pareto front is unknown. Thus, the accuracy of these performance measures are affected, and it is hard to have an accurate evaluation of an algorithm or a fair comparison between algorithms. Second, due to the discrete and combinatorial solution space, the Pareto front of scheduling problems can be very irregular. This makes it very difficult to design an intelligent search process to put more efforts on search towards the areas that are harder to reach (e.g. with fewer Pareto solutions). Third, in scheduling problems, the objectives can have quite different magnitudes. It is hard to normalise the objectives properly, since the upper and lower bounds are usually unknown. Improper normalisation will lead to poor search process which can only reach a small (biased) part of the Pareto front.

A. Existing studies

There are mainly three categories of existing methods to deal with multi-objective scheduling problems: (1) *aggregation* methods, (2) *dominance-based* method and (3) *decomposition-based* methods.

The aggregation methods follow the most intuitive way that transforms multiple objectives into a single one, e.g. by the weighted sum approach. For example, for production scheduling, Rajendran and Ziegler [40] proposed a multi-objective ant colony system for minimising makespan and total flowtime. The algorithm adopted the weighted sum approach, and different trade-off solutions were obtained by running the algorithm with different weight vectors. For vehicle routing, Ombuki et al. [41] proposed a multi-objective evolutionary algorithm consisting of two approaches, one is weighted sum and the other is the dominance-based approach. For Web service composition, Da Silva et al. [42] used an aggregation method to combine four objectives into a single one.

The dominance-based approaches are based on the Pareto dominance relation between objective vectors. Wang et al. [43] proposed a multi-objective genetic algorithm for flexible job-shop scheduling problem, which adopts the fitness evaluation of SPEA2 [44]. For vehicle routing, Tan el al. [45] proposed a multi-objective evolutionary algorithm hybridized with local search. Tan et al. [46] proposed a multi-objective particle swarm optimization for Web service location allocation.

The decomposition-based approaches [47] are somehow similar to the aggregation methods, in the sense of combining the objectives into a single one, and transforming the original multi-objective optimisation into a single-objective optimisation. However, they create multiple such single-objective optimisation problems simultaneously, each by a subset of the population. For arc routing, Mei et al. [48] proposed a decomposition-based memetic algorithm, which uses the dominance-based and aggregation-based approaches simultaneously. When updating the individuals in the population, the dominance-based evaluation is used. During the local search, on the other hand, multiple objectives are combined into a single one by weighted sum. The weight vector is defined according to the location of the individual in the objective space.

B. Research challenges and future directions

The challenges of multi-objective scheduling include both the challenges of multi-objective evolutionary optimisation and the challenges in scheduling. Here, four main challenges are discussed: (1) fitness assignment during local search, (2)

normalisation, (3) performance assessment and (4) many-objective optimisation.

First, it has been demonstrated that the combination of global search and local search (e.g. memetic algorithm) is quite effective to deal with the combinatorial solution space in scheduling problems [48]–[50]. However, unlike in population-based algorithms, which can maintain a set of non-dominated solutions, one can only move to one neighbor during the local search. How to select one solution out of a set of non-dominated solutions during the local search becomes an important issue. Some work has been done by aggregating the objectives with weighted sum [48], [51], [52]. However, how to properly set the weight vector is still an open issue.

Second, it is known that normalisation is important both in the optimisation process and the final performance assessment. It has been used in all existing methods[2]. Normalisation usually depends on the upper and lower bounds of the objectives. However, there are two issues in normalisation in multi-objective scheduling. First, the bounds of the objectives are often unknown. The magnitudes of different objectives can be quite different, and the information to estimate the bounds can vary from one objective to another. For example, in vehicle routing problem, estimating the bounds of the number of vehicles is easier than estimating the bounds of the total cost. Therefore, the tightness of the estimated bound is different cross the objectives, which leads to unexpected bias to the objectives with tighter estimated bounds during the search process. Second, the distribution over all the possible solutions is not uniform within the estimated bounds. For example, when estimating the number of vehicles, the upper bound is estimated by assuming that each vehicle serves only one customer. Then the upper bound equals to the number of customers (nodes). However, such an upper bound is rarely reached, and most of the individuals have much less number of vehicles than the estimated upper bound. Similar phenomenon has been found in the multi-objective Web service location allocation, in which the distribution of solutions is very biased in the total cost objective space. In this situation, the search will also bias to the objectives with more uniform distribution, which can be considered as having tighter estimated bounds.

Third, due to the unknown true Pareto front and bounds of the objectives, it is hard to have a proper performance assessment of algorithms. For example, we cannot properly use the performance metrics that rely on the true Pareto front such as IGD and the ϵ indicator. When using hyper-volume, the nadir point is important and determines the accuracy of the assessment. Assuming in minimisation, it is normally set to the upper bounds of the objectives. However, such a setting relies on the accuracy of the bound estimation. Similar to the second issue, the hyper-volume metric will bias to the solutions which perform better in the objectives with tighter estimated bounds. In addition, the true Pareto front of the problem may not be uniformly distributed. Therefore, even the true Pareto front may be worse than some other non-dominated sets in terms of the uniformity performance metrics. In other words, it my not be proper to use the uniformity performance metrics (e.g. spacing).

Fourth, there can be many objectives in scheduling (e.g. makespan, total flowtime and tardiness, maximal flowtime and tardiness, proportion of tardy jobs in job shop scheduling). Scheduling in cloud must address many different concerns and quality-of-service (QoS) requirements that are usually conflicting in nature. Consequently, it is natural to consider cloud scheduling as a many-objective optimisation problem. Developing effective algorithms for many-objective scheduling in the cloud remains to be a challenging task. Many-objective optimisation itself is very challenging in the EMO field. Thus, in scheduling problems, one also needs to deal with the challenges of many-objective optimisation such as the exponentially increasing non-dominated solutions, difficulty to maintain useful building blocks, huge population size, visualisation, etc.

V. Others

Previous sections shows three major issues in evolutionary scheduling and combinatorial optimisation. There are many other interesting issues, in terms of theory and practice, that demand more search and we would like to discuss in this section.

A. Modeling and formulation

EC has been studied intensively for decades but its applications are very modest. One of the main reasons is that defining/formulating the problems (especially scheduling and combinatorial optimisation problems) to be solved with EC is tricky for ones with little or no programming skills. Mathematical programming such as linear programming has been widely used because there are many available modeling languages (e.g. AMPL, GAMS) for these solution methods. The key idea is that ease of modeling makes them more popular. In our knowledge, there is no well-established modeling language dedicated to EC. In the new version of the Excel solver, evolution algorithms are available as an optimisation method. Although we can define constraints and decisions variables in the Excel solver, it is quite cumbersome and not suitable for complex problems such as scheduling and combinatorial optimisation. Evolutionary computing modeling language (ECML) [53] is an innovative approach but it is mainly used to specify genotype structures and implementation of EC methods. ILOG solver is a good example of combining constraint programming (CP) and EC. In this case, EC can be used to solve the problem modeled in CP. However, this is restricted to basic implementation of genetic algorithm and advanced EC techniques are not considered.

Similar to modeling languages used for mathematical programming, we also want to have a modeling language that is (1) easy to learn, (2) flexible to deal with complex problems, and (3) able to connect with variety of EC and other solvers. Another important feature that is essential for EC based

[2]In the dominance-based methods, normalisation may not be necessary in ranking the individuals based on dominance relation. However, it is important for diversity preservation, e.g. calculating the crowding distance in NSGA-II.

modeling language is visualisation, especially when dealing with scheduling and combinatorial optimisation problems. This allows us to identify special structures of the problem to improve the efficiency and effectiveness of EC.

B. Adaptability, scalability and reusability

Advances in memetic algorithms and automated design of heuristics in the last decade have helped us cope with many challenging problems in scheduling and combinatorial optimisation. They also help to point out some important issues regarding the design of EC algorithms such as adaptability, scalability and reusability. These issues are not new but they still remain a big challenge in evolutionary scheduling and combinatorial optimisation. For adaptability, we try to make EC methods self-adapted, which is necessary to cope with a wide range of optimisation problems (in the same or different classes). This has been investigated intensively in many existing studies on memetic algorithms [54] and hyper-heuristics [55] and will be an active research topic in future studies. For scalability, researchers have created advanced EC techniques using ideas from different fields so that the developed algorithms can deal with large-scale instances. Parallel EC-based methods would be an interesting area to study for scheduling and combinatorial optimisation. Regarding reusability, we would like to emphasise on the ability of EC methods to reuse the knowledge obtained in its search for solving other instances and problems. Automated design of heuristics [5], [56] is currently applying the ideas of reusability to generate heuristics that can cope with different situations. However, the research is still at an early stage and its potentials have not yet revealed.

C. Performance assessment

Our discussions so far have shown that there are many aspects that we need to consider when designing EC methods for scheduling and combinatorial optimisation. But, *how can we determine which algorithms are most suitable for our applications?* For most existing studies on evolutionary scheduling and combinatorial optimisation, comparisons are easy as we only look at the quality of obtained solution in terms of objective values and running times. If we take into account other aspects such as stability, predictability, adaptability, scalability, comparisons will become very complicated. We believe that it is important to develop an agreeable framework for such comparisons, possibly not to identify a single best method but analyse their strength and weakness systematically.

VI. Conclusions

This paper discussed the current status of evolutionary scheduling and combinatorial optimisation and revealed some key challenges that need to be addressed in future studies. Our perspective is that future studies should focus on practical requirements and the environment to which the solution methods will be applied. Theoretical studies are important but they should be guided towards facilitating the applications of EC methods and providing insights on how EC methods should behave in practice. The combination of techniques from different fields such as OR, AI, and machine learning would be the key for the development of powerful EC methods. Besides handling major issues such as dynamic changes, multiple interdependent decisions, and multiple objectives, more concerns should be given to modeling, adaptability, scalability, reusability and systematic approaches to comparing different solution methods. In this sense, hyper-heuristics (e.g. GP) have a great potential to automatically learn generic, scalable and competitive heuristics and meta-heuristics.

References

[1] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algorithms," *Evolutionary Computation*, vol. 7, no. 1, pp. 1–17, Mar. 1999.

[2] J. Fang and Y. Xi, "A rolling horizon job shop rescheduling strategy in the dynamic environment," *The International Journal of Advanced Manufacturing Technology*, vol. 13, no. 3, pp. 227–232, 1997.

[3] J. Branke and D. C. Mattfeld, "Anticipation and flexibility in dynamic scheduling," *International Journal of Production Research*, vol. 43, no. 15, pp. 3103–3129, 2005.

[4] X.-N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Information Sciences*, vol. 298, pp. 198–224, 2015.

[5] J. Branke, S. Nguyen, C. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, 2015, DOI:10.1109/TEVC.2015.2429314.

[6] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Dynamic multi-objective job shop scheduling: a genetic programming approach," in *Automated Scheduling and Planning*. Springer, 2013, vol. 505, pp. 251–282.

[7] C. W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter, "Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems," *International Journal of Production Economics*, vol. 145, no. 1, pp. 67–77, 2013.

[8] H. N. Psaraftis, "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, vol. 14, no. 2, pp. 130–154, 1980.

[9] Z.-L. Chen and H. Xu, "Dynamic column generation for dynamic vehicle routing with time windows," *Transportation Science*, vol. 40, no. 1, pp. 74–88, 2006.

[10] P. Kilby, P. Prosser, and P. Shaw, "Dynamic vrps: A study of scenarios," *University of Strathclyde Technical Report*, pp. 1–11, 1998.

[11] J. Yang, P. Jaillet, and H. Mahmassani, "Real-time multivehicle truckload pickup and delivery problems," *Transportation Science*, vol. 38, no. 2, pp. 135–148, 2004.

[12] M. Gendreau, F. Guertin, J.-Y. Potvin, and E. Taillard, "Parallel tabu search for real-time vehicle routing and dispatching," *Transportation science*, vol. 33, no. 4, pp. 381–390, 1999.

[13] S. Ichoua, M. Gendreau, and J.-Y. Potvin, "Diversion issues in real-time vehicle dispatching," *Transportation Science*, vol. 34, no. 4, pp. 426–438, 2000.

[14] G. Fleury, P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Improving robustness of solutions to arc routing problems," *Journal of the operational research society*, vol. 56, no. 5, pp. 526–538, 2005.

[15] Y. Mei, K. Tang, and X. Yao, "Capacitated arc routing problem in uncertain environments," in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, 2010.

[16] ——, "Evolutionary computation for dynamic capacitated arc routing problem," in *Evolutionary Computation for Dynamic Optimization Problems*, S. Yang and X. Yao, Eds. Springer, 2013.

[17] J. Wang, K. Tang, and X. Yao, "A memetic algorithm for uncertain capacitated arc routing problems," in *Memetic Computing (MC), 2013 IEEE Workshop on*. IEEE, 2013, pp. 72–79.

[18] J. Wang, K. Tang, J. Lozano, and X. Yao, "Estimation of distribution algorithm with stochastic local search for uncertain capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2015.2428616

[19] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10951-008-0090-8

[20] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[21] D.-J. Wang, F. Liu, Y.-Z. Wang, and Y. Jin, "A knowledge-based evolutionary proactive scheduling approach in the presence of machine breakdown and deterioration effect," *Knowledge-Based Systems*, vol. 90, pp. 70 – 80, 2015.

[22] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Learning reusable initial solutions for multi-objective order acceptance and scheduling problems with genetic programming," in *Genetic Programming*, ser. LNCS. Springer, 2013, vol. 7831, pp. 157–168.

[23] J. W. M. Bertrand and J. C. Fransoo, "Operations management research methodologies using quantitative modeling," *International Journal of Operations & Production Management*, vol. 22, no. 2, pp. 241–264, 2002.

[24] M. Bonyadi, Z. Michalewicz, and L. Barone, "The travelling thief problem: The first step in the transition from theoretical problems to realistic problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, pp. 1037–1044.

[25] J. Stolk, I. Mann, A. Mohais, and Z. Michalewicz, "Combining vehicle routing and packing for optimal delivery schedules of water tanks," *OR Insight*, vol. 26, no. 3, pp. 167–190, 2013.

[26] V. F. Yu, P. Jewpanya, and V. Kachitvichyanukul, "Particle swarm optimization for the multi-period cross-docking distribution problem with time windows," *International Journal of Production Research*, pp. 1–17, 2015.

[27] G. Zpfel and M. Bgl, "Multi-period vehicle routing and crew scheduling with outsourcing options," *International Journal of Production Economics*, vol. 113, no. 2, pp. 980 – 996, 2008, special Section on Advanced Modeling and Innovative Design of Supply Chain.

[28] L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing*, vol. 10, no. 3, pp. 888 – 896, 2010.

[29] J. Tay and D. Wibowo, "An effective chromosome representation for evolving flexible job shop schedules," in *Genetic and Evolutionary Computation GECCO 2004*, ser. Lecture Notes in Computer Science, K. Deb, Ed., 2004, vol. 3103, pp. 210–221.

[30] P. Unachak and E. Goodman, "Solving multiobjective flexible job-shop scheduling using an adaptive representation," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 2010, pp. 737–742.

[31] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Computers & Operations Research*, vol. 30, no. 8, pp. 1151 – 1171, 2003.

[32] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 193–208, 2014.

[33] X. Hao, X. Chen, H. Lin, and T. Murata, "Cooperative bayesian optimization algorithm: A novel approach to simultaneous multiple resources scheduling problem," in *Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second International Conference on*, Dec 2011, pp. 212–217.

[34] Q.-K. Pan, "An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling," *European Journal of Operational Research*, vol. 250, no. 3, pp. 702 – 714, 2016.

[35] Y. Mei, X. Li, and X. Yao, "On investigation of interdependence between sub-problems of the travelling thief problem," *Soft Computing*, pp. 1–16, 2014.

[36] Y. Mei, X. Li, F. Salim, and X. Yao, "Heuristic evolution with genetic programming for traveling thief problem," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 2753–2760.

[37] L. Ai, M. Tang, and C. Fidge, "Resource allocation and scheduling of multiple composite web services in cloud computing using cooperative coevolution genetic algorithm," in *Proceedings of the 18th International Conference on Neural Information Processing - Volume Part II*, ser. ICONIP'11, 2011, pp. 258–267.

[38] J. Park, S. Nguyen, M. Zhang, and M. Johnston, "Genetic programming for order acceptance and scheduling," in *2013 IEEE Congress on Evolutionary Computation (CEC)*, C. A. Coello Coello and L. G. De la Fraga, Eds. Piscataway, NJ: IEEE Press, 2013, pp. 1005–1012.

[39] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[40] C. Rajendran and H. Ziegler, "A multi-objective ant-colony algorithm for permutation flowshop scheduling to minimize the makespan and total flowtime of jobs," in *Computational Intelligence in Flow Shop and Job Shop Scheduling*. Springer, 2009, pp. 53–99.

[41] B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Applied Intelligence*, vol. 24, no. 1, pp. 17–30, 2006.

[42] A. S. da Silva, H. Ma, and M. Zhang, "A graph-based particle swarm optimisation approach to qos-aware web service composition and selection," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 3127–3134.

[43] X. Wang, L. Gao, C. Zhang, and X. Shao, "A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 5, pp. 757–767, 2010.

[44] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[45] K. Tan, Y. Chew, and L. Lee, "A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems," *European Journal of Operational Research*, vol. 172, no. 3, pp. 855–885, 2006.

[46] B. Tan, Y. Mei, H. Ma, and M. Zhang, "Particle swarm optimization for multi-objective web service location allocation," in *The 16th European Conference on Evolutionary Computation in Combinatorial Optimisation 2016*. Springer, to appear.

[47] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, 2007.

[48] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[49] K. Tang, Y. Mei, and X. Yao, "Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.

[50] Y. Mei, X. Li, and X. Yao, "Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 435–449, 2014.

[51] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European Journal of Operational Research*, vol. 137, no. 1, pp. 50–71, 2002.

[52] P. Lacomme, C. Prins, and M. Sevaux, "A genetic algorithm for a bi-objective capacitated arc routing problem," *Computers and Operations Research*, vol. 33, no. 12, pp. 3473–3493, 2006.

[53] H. Aydt, S. J. Turner, W. Cai, M. Y. H. Low, Y.-S. Ong, and R. Ayani, "Toward an evolutionary computing modeling language," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 2, pp. 230–247, 2011.

[54] X. S. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, to appear.

[55] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[56] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*, 2009, vol. 1, pp. 177–201.