An Evolutionary Search Paradigm that Learns with Past Experiences

Liang Feng, Yew-Soon Ong, Ivor Wai-Hung Tsang, and Ah-Hwee Tan
Center for Computational Intelligence
School of Computer Engineering
Nanyang Technology University, Singapore
Email: {feng0039, asysong, ivortsang, asahtan}@ntu.edu.sg

***Abstract*—A major drawback of evolutionary optimization approaches in the literature is the apparent lack of automated knowledge transfers and reuse across problems. Particularly, evolutionary optimization methods generally start a search from scratch or ground zero state, independent of how similar the given new problem of interest is to those optimized previously. In this paper, we present a study on the transfer of knowledge in the form of useful structured knowledge or latent patterns that are captured from previous experiences of problem-solving to enhance future evolutionary search. The essential contributions of our present study include the *meme learning* and *meme selection* processes. In contrast to existing methods, which directly store and reuse specific problem solutions or problem sub-components, the proposed approach models the structured knowledge of the strategy behind solving problems belonging to similar domain, i.e., via learning the mapping from problem to its corresponding solution, which is encoded in the form of identified knowledge representation. In this manner, knowledge transfer can be conducted across problems, from differing problem size, structure to representation, etc. A demonstrating case study on the capacitated arc routing problem (CARP) is presented. Experiments on benchmark instances of CARP verified the effectiveness of the proposed new paradigm.**

I. Introduction

Evolutionary algorithm (EA) draws inspiration from biological evolution. In spite of the great deal of attention it has received in the last decade for its efficiency and effectiveness in solving complex problems that are otherwise deem as non-optimal or intractable for conventional approaches [1], [2], it is worth noting that the successful evolution of useful traits to date has been restricted to the particular problem instance of interest at the point in time. More precisely, existing evolutionary optimization approaches generally start the search for a given new problem from scratch. There is the common practice to start the search on given new problem of interest from ground-zero state, independent of how similar the new problem instance of interest is to those encountered in the past. Thus a major drawback of existing evolutionary search methodology in the literature is the apparent lack of automated knowledge transfers and reuse from experiences on past problems. *In other words, the transfer of useful traits across similar tasks or problems or the study of optimization that evolves with problems has been significantly under-explored.* This paper thus presents an initial attempt to fill in this gap.

It is well established that problems seldom exist in isolation, and past problems encountered may yield useful information for more effective future problem-solving. In the context of evolutionary computation, for instance, Louis *et al.* [3] presented a study to memorize problem specific knowledge and subsequently using them to aid in the genetic algorithm (GA) search. Rather than starting anew on each problem, appropriate intermediate solutions drawn from similar problems previously solved are periodically injected into the GA population. However, if the given new problem happens to differ in problem size, structure, representation, etc., what have been previously memorized from past problems cannot be directly injected into the search for reuse. In a separate study, Cunningham and Smyth [4] explored the reuse of established high quality schedules from past problems to bias the search on new scheduling problems of the traveling salesman problem (TSP). Likewise, the drawback is that the approach proposed is also not invariant to the unique features of the differing problem instances, including solution representations, tasks and demands. In CARP, for example, a service order found for a CARP instance cannot be directly reused in another problem instance, despite the two sharing equal problem or node size and connectivities of nodes, but differ in demand criterion. In spite of the significant benefits to reuse useful traits from previous experiences in search, few successful attempts to emulate the learning and evolution of search across problems of differing properties have been reported. Any that even exists today are achieved based on a simple memory-based approach as opposed to generalizations across problems, since the solution to this task is non-trivial.

Like gene in genetics, a meme is synonymous to memetic as being a building block of cultural know-how that is transmissible and replicable [2]. While genes form the "instruction for building proteins", memes are "instructions for carrying out behavior, stored in brains". In the context of evolutionary computation, while genes are encoded as solutions of the problem of interest, meme shall denote the structured knowledge or latent pattern which is helpful for effective problem solving. In contrast to existing works, in this paper, we propose a novel study on the transfer of memes as building blocks of useful information that are captured from past problem solving experiences to enhance evolutionary search. In this manner, any knowledge transfer can be conducted across problems of differing sizes, structures or representations, etc. The core ingredients of our proposed new paradigm involve a *learning*

978-1-4673-1509-8/12/$31.00 ©2012 IEEE

phase to capture memes in the form of structured knowledge or latent pattern from past experiences of problem-solving, and subsequently a *selection* phase to identify a suitable meme from accumulated meme pool for future reuse. In the present study, we focus on the class of combinatorial search problems, particularly, capacitated arc routing problem as the demonstrating case study. Meme is manifested as the instruction of task assignment captured from past solved CARP instances. *Meme learning* is realized using the *Hilbert-Schmidt Independence Criterion* (HSIC) [5], while *meme selection* is achieved according to both the HSIC and *Maximum Mean Discrepancy* criteria [6]. Last but not least, experimental study on CARP benchmarks verified the effectiveness of our proposed methodology for enhancing evolutionary search.

The rest of the paper is organized as follows: a brief introduction on the core components of our proposed study on a paradigm that transfers useful traits as memes from past problem-solving experiences to enhance the evolutionary search, is given in Section II. Section III presents the brief mathematical formulation of CARP and discusses typical evolutionary approaches for solving CARP. The detailed designs of the *meme learning* and *meme selection* in evolutionary search are then described in Section IV. Section V presents and analyzes the experimental results on commonly used CARP benchmarks. Finally, section VI summarizes the paper with the brief conclusion.

II. Memes as Building Blocks of Problem Solving Experiences

"Meme" was first defined by Dawkins as "the basic unit of cultural transmission via imitation", in his book entitled "The Selfish Gene" [7]. The term has inspired the new science of memetics which today has served as a motivational pillar and inspiration toward the possibility of meme developing into a proper hypothesis of the human mind. Particularly, memes are modeled as recurring real-world patterns or domain-specific knowledge encoded in computational representations for the purpose of effective problem-solving.

In the present study, we propose a novel manifestation of meme, in the form of useful traits learned and captured from past evolutionary search experiences on different problem instances. The society of memes then form the domain knowledge that may be activated to solve future evolutionary search more effectively, when appropriately harnessed. In particular, the two core ingredients of the proposed current search paradigm of interest are summarized here:

- *Meme Learning*: In contrast to a simple storage or memory of specific problem $\mathbf{X}$ with associated solution $\mathbf{y}*$ as considered in the past studies with case-based reasoning [3], meme learning models the mapping of $\mathbf{X}$ to $\mathbf{y}*$. The process proceeds in an incremental manner, and builds up the knowledge from solving a series of related problems.
- *Meme Selection*: All prior knowledge introduces various kinds of bias into the search. Hence a certain biases would make the search more efficient on some classes of problem instances but not for others. Inappropriately harnessed knowledge, on the other hand, may lead to the possible impairments of the search. The meme selection process thus serves to identify the suitable building blocks or meme(s) from the society of memes to operate on future unseen problems.

III. Case Study on Capacitated Arc Routing Problem

To demonstrate the detailed designs for the transfer of useful traits from past experiences in problem-solving, here we present the case study on a challenging combinatorial search problem, in particular, the capacitated arc routing problem. In this section, we present the mathematical formulation of the CARP followed by a brief discussion on some methods for solving CARPs.

A. CARP Formulation

The capacitated arc routing problem (CARP) was first proposed by Golden and Wong [8] in 1981. It can be formally stated as follows: Given a connected undirected graph $G = (V, E)$, where vertex set $V = \{v_i\}, i = 1 \dots n$, n is the number of vertices, edge set $E = \{e_i\}, i = 1 \dots m$ with m denoting the number of edges. Consider a demand set $D = \{d(e_i)|e_i \in E\}$, where $d(e_i) > 0$ implies edge e_i requires servicing, a travel cost vector $C_t = \{c_t(e_i)|e_i \in E\}$ with $c_t(e_i)$ representing the cost of traveling on edge e_i, a service cost vector $C_s = \{c_s(e_i)|e_i \in E\}$ with $c_s(e_i)$ representing the cost of servicing on edge e_i. A solution of CARP can be represented as a set of travel circuits $\mathcal{S} = \{\mathcal{C}_i\},\ i = 1 \dots k$ which satisfies the following constraints:

1) Each travel circuit $\mathcal{C}_i,\ i \in [1, k]$ must start and end at the depot node $v_d \in V$.
2) The total load of each travel circuit must be no more than the capacity W of each vehicle, $\sum_{\forall e_i \in \mathcal{C}} d(e_i) \leq W$.
3) $\forall e_i \in E\ and\ d(e_i) > 0$, there exists one and only one circuit $\mathcal{C}_i \in \mathcal{S}$ such that $e_i \in \mathcal{C}_i$.

The cost of a travel circuit is then defined by the total service cost for all edges that needed service together with the total travel cost of the remaining edges that formed the circuit:

$$cost(\mathcal{C}) = \sum_{e_i \in \mathcal{C}_s} c_s(e_i) + \sum_{e_i \in \mathcal{C}_t} c_t(e_i) \tag{1}$$

where $\mathcal{C}_s$ and $\mathcal{C}_t$ are edge sets that required servicing and those that do not, respectively. And the objective of CARP is then to find a valid solution $\mathcal{S}$ that minimizes the total cost:

$$C_{\mathcal{S}} = \sum_{\forall \mathcal{C}_i \in \mathcal{S}} cost(\mathcal{C}_i) \tag{2}$$

B. CARP Handling Methodology

The problem of solving CARP is typically addressed as two phases. The first phase assigns the arcs requiring services (otherwise known as tasks) to the appropriate vehicles. Subsequently, the objective is to find the optimal service order of each vehicle for the assigned tasks. An example solution to a CARP is illustrated in Fig. 1, with v_d representing the

depot, full line denoting arcs that require servicing and dashed lines representing arcs that do not. Each task is labeled with a unique integer number (e.g., the task between v_2 to v_1 is labeled as index 2), while the index enclosed in brackets denotes the inversion of each task (i.e., direction of arc) accordingly. In Fig. 1, three vehicle routes $\mathcal{C}_1 = \{0, 4, 2, 0\}$, $\mathcal{C}_2 = \{0, 5, 7, 0\}$, and $\mathcal{C}_3 = \{0, 9, 11, 0\}$ can be observed, each composing of two tasks. A '0' index value is assigned at the beginning and end of circuits to indicate that each circuit starts and ends at the depot.

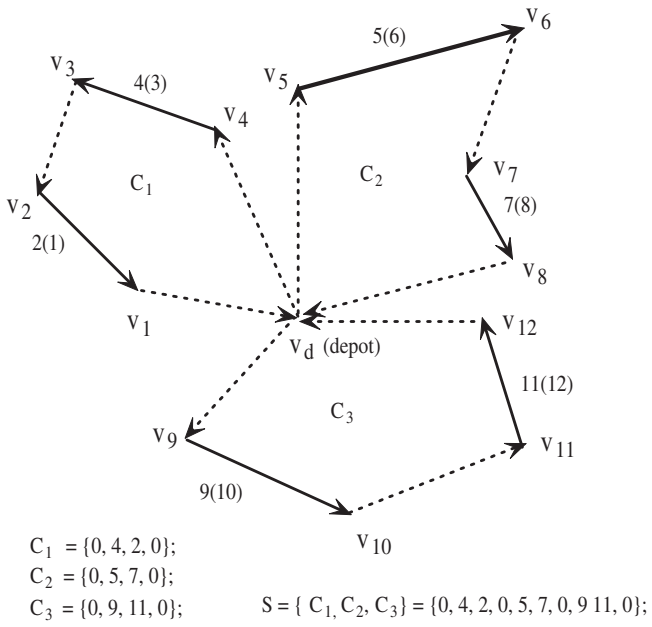
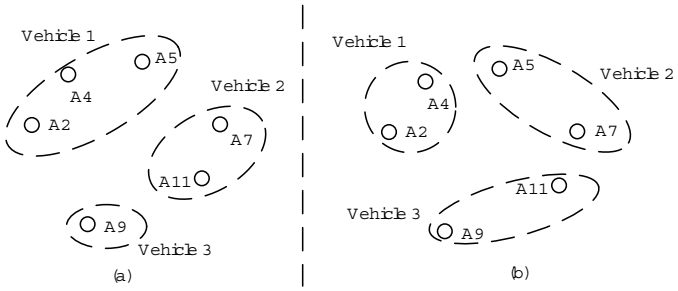


Fig. 1. An example of the CARP ([9])

In the evolutionary search literature, the task assignment in the first phase is typically addressed during population initialization, with random task assignment being the most simple and common scheme used [10]. In addition to random assignment, popular heuristic methods to arrive at a population of solutions with more informed assignments of tasks to vehicles have also been introduced [11], [10]. It is worth noting that clustering has been posed as an effective approach for the assignment of tasks in routing problems, particularly in vehicle routing [12], [13]. In CARP however, few or no works have attempted to do so to date. The core challenge lies in the identification and transformation of present problem representations of CARPs, to one that enables clustering approaches to be applied directly for task assignments.

Using the example aforementioned in Fig. 1, Fig. 2 is redrawn with each task in the form now represented as a node, and the tasks assigned to a vehicle enclosed by a dashed circle. Fig. 2(a) shows the tasks assignment based on the well-established K-Means clustering with simple Euclidean distance metric. The associated table in the figure thus provides a representation of the CARP problem in the form of a feature vector $\mathbf{X}$, and the corresponding assigned tasks as $\mathbf{y}_{kmeans}$ and $\mathbf{y}_{optimized}$ via the K-Means clustering approach and by means of evolutionary optimization, respectively.



$\mathbf{X}$	X_{A2}	X_{A4}	X_{A5}	X_{A7}	X_{A9}	X_{A11}
$\mathbf{Y}_{kmeans}$	1	1	1	2	3	2
$\mathbf{Y}_{optimized}$	1	1	2	2	3	3

Fig. 2. Task assignment by means of clustering and evolutionary optimization (The indexes in $\mathbf{y}_{kmeans}$ and $\mathbf{y}_{optimized}$ (i.e., 1, 2, 3) label the vehicle that a corresponding tasks belongs to.)

In the rest of this paper, we demonstrate the idea and mechanisms for learning and capturing of useful traits from past problem solving experiences as memes or building blocks that are useful for enhancing future evolutionary optimization. In particular, we model the mappings of solved CARP instances to their optimized clusters of tasks assignment as memes that can then be deployed to appropriately bias the task assignments on future unseen CARP instances.

IV. Memes Learning and Selection in CARP

The essential ingredients of the proposed study is depicted in Fig. 3. In the first step, memes of previously solved CARP instances are captured via the *meme learning* process, which are stored in the meme pool or society of memes notated here by $\mathbf{M}$s. For any given new unseen CARP instance, the *meme selection* process kicks in to identify a suitable meme from $\mathbf{M}$s, which is subsequently used to bias the task assignment during population initialization. Next, the conventional evolutionary search operators then proceed until the user-specified stopping condition is satisfied. The attained optimized solution together with the CARP instance will also be archived for subsequent learning via the *meme learning* process.

In the following sections, we present the realizations of the proposed meme learning and selection processes for enhancing the evolutionary search, using the context of CARP.

A. Meme Learning

This section describes the capturing of memes, in the form of useful traits, from a given CARP instance $\mathbf{X}$ with corresponding optimized task assignment $\mathbf{y}*$. Thus, a meme carries the instruction for mapping the distribution of tasks, according to previously optimized task assignments. In particular, for two given tasks $\mathbf{x}_a = (x_{a1}, \ldots, x_{ap})^T$ and $\mathbf{x}_b = (x_{b1}, \ldots, x_{bp})^T$,

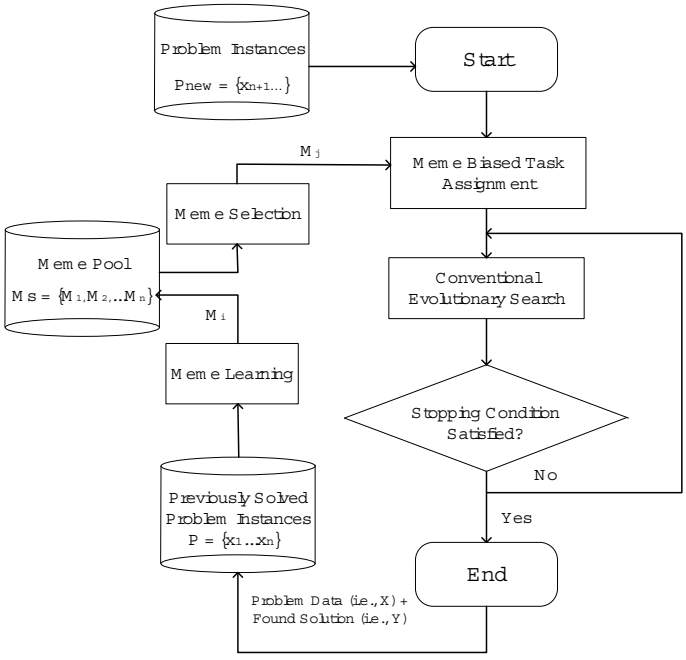


Fig. 3. Evolutionary Optimization by Learning from Past Experiences

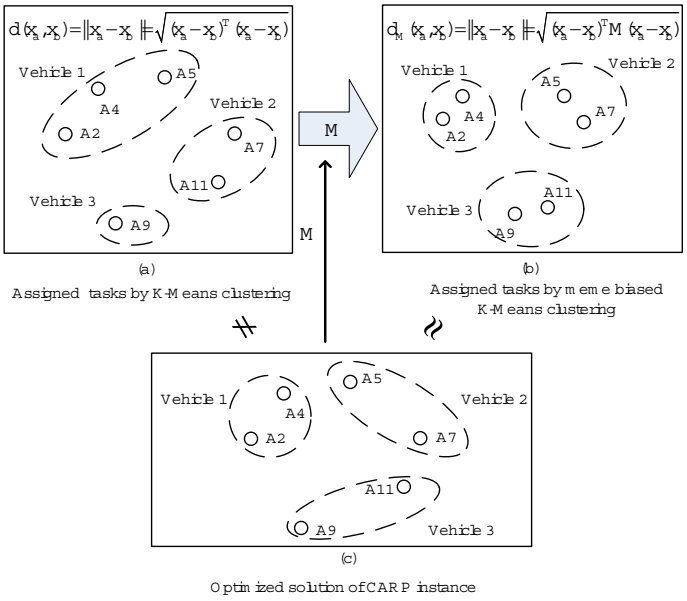


Fig. 4. Task assignment by means of optimization and clustering with and without incorporation of the learned meme

the distance between them in the p-dimensional space $\mathbb{R}^p$ is given by:

$$d_M(\mathbf{x}_a, \mathbf{x}_b) = ||\mathbf{x}_a - \mathbf{x}_b||_M = \sqrt{(\mathbf{x}_a - \mathbf{x}_b)^T \mathbf{M} (\mathbf{x}_a - \mathbf{x}_b)}$$

where T denotes the transpose of a matrix or vector. $\mathbf{M}$ is positive semidefinite and, it can be decomposed as $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ by means of singular value decomposition (SVD). Substituting this decomposition into $d_M(\mathbf{x}_a, \mathbf{x}_b)$, we arrive at:

$$d_M(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{(\mathbf{L}^T\mathbf{x}_a - \mathbf{L}^T\mathbf{x}_b)^T (\mathbf{L}^T\mathbf{x}_a - \mathbf{L}^T\mathbf{x}_b)} \quad (3)$$

Equation 3 shows that the distances among tasks are scaled by $\mathbf{M}$. Thus, meme $\mathbf{M}$ gives the instruction of reinforcements to the tasks representations in the problem space. Particularly, tasks that are served by a common vehicle are reinforced to be closer to one another while tasks served by different vehicles are kept further apart. Using the CARP example of Fig. 2, Fig. 4 illustrates the process of the meme learning at work, with meme $\mathbf{M}$ learned from the optimized CARP instance (see Fig. 4(c)). The incorporation of the learned meme into the task assignment process then led to clustered tasks that are closer to the optimized solution, as shown in Fig. 4(b).

To learn and capture meme $\mathbf{M}$ from a given CARP instance $\mathbf{X}$ with corresponding optimized task assignment $\mathbf{y}*$, we maximize the dependence between $\mathbf{X}$ and $\mathbf{y}*$ using the *Hilbert-Schmidt Independence Criterion* (HSIC) [5], which is defined by:

$$\begin{aligned} \max_{\mathbf{K}} \quad & tr(\mathbf{HKHY}) \quad (4) \\ \text{s.t.} \quad & \mathbf{K} = \mathbf{X}^T * \mathbf{M} * \mathbf{X} \\ & \mathbf{K} \succeq 0 \end{aligned}$$

where $tr(\cdot)$ denotes the trace operation of a matrix. $\mathbf{K}$, $\mathbf{Y}$ are the kernel matrices for the CARP instance $\mathbf{X}$ and the corresponding task assignment $\mathbf{y}*$, respectively. Further, if task i and task j are served by the same vehicle, $\mathbf{Y}(i,j) = 1$, otherwise, $\mathbf{Y}(i,j) = -1$. $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{I}\mathbf{I}'$ centers the data and the labels in the feature space, $\mathbf{I}$ denotes the identity matrix, n equals to the number of tasks. A higher value of Equation 4 indicates greater dependency between CARP instance $\mathbf{X}$ and corresponding optimized task assignment $\mathbf{y}*$.

Taking the constraints into considerations, Equation 4 becomes:

$$\begin{aligned} \max_{\mathbf{M}} \quad & tr(\mathbf{XHYHX}^T\mathbf{M}) \quad (5) \\ \text{s.t.} \quad & \mathbf{M} \succeq 0 \\ & tr(\mathbf{M}^p) \leq B \end{aligned}$$

where the first constraint enforces the learned matrix $\mathbf{M}$ to be positive semi-definite, and the second constraint regulates the capacity of $\mathbf{M}$, while p and B are pre-defined constants.

To solve the learning problem in Equation 5, let $\mathbf{A} = \mathbf{XHYHX}^T$, it is easy to confirm that $\mathbf{A}$ is a symmetric matrix. So $\mathbf{A}$ can be decomposed as $\mathbf{V}diag(\boldsymbol{\delta})\mathbf{V}^T$ by eigen-decomposition, where $\mathbf{V}$ contains columns of orthonormal eigenvectors of $\mathbf{A}$ and $\boldsymbol{\delta}$ is a vector of the corresponding eigenvalues. According to the Proposition in [14], $\mathbf{M}$ in Equation 5 possesses a closed-form solution:

$$\mathbf{M} = \left(\frac{B}{trace(\mathbf{A}_+^{\frac{p}{p-1}})} \right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}} \quad (6)$$

where $\mathbf{A}_+ = \mathbf{V}diag(\boldsymbol{\delta}_+)\mathbf{V}^T$, and $\boldsymbol{\delta}_+$ is a vector with entries equal to $max(0, \boldsymbol{\delta}[i])$.

For $p = 1$, the optimal solution $\mathbf{M}$ can be expressed as closed-form solution:

$$\mathbf{M} = B\mathbf{A}_1 \tag{7}$$

where $\mathbf{A}_1 = \mathbf{V}diag(\boldsymbol{\delta}_1)\mathbf{V}^T$, and $\boldsymbol{\delta}_1$ is a vector with entries equal to $\frac{1}{\sum_{i:\boldsymbol{\delta}_i = max(\boldsymbol{\delta})} 1}$ for all i that $\boldsymbol{\delta}[i] = max(\boldsymbol{\delta})$, otherwise, the entries are zeros.

B. Meme Selection

To reuse useful traits of past problem solving experience, a straightforward approach is to use the captured past experiences as training data, and build a predictive model for future problems. However, this training process requires sufficient training data to cover all the possible data distributions. In CARP, the source of solved CARP instances and their optimized solutions is limited in reality, hence it may be hard to obtain a generalized model that represents the class of CARP reliably. Further, each CARP instance processes different characteristics and latent structures, hence the captured memes of CARP instances can lead to unique biases in task assignment. Thus, the *meme selection* process plays an important role in the identifications of suitable meme for future unseen problem in the spirit of transfer learning [15].

Suppose there is a set of n different $\mathbf{M}$, $\mathbf{M}_s = \{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\}$, the selected meme $\mathbf{M}$ is defined as a linear combination of the m memes:

$$\mathbf{M}_t = \sum_{i=1}^{n} \mu_i \mathbf{M}_i$$

$$\mathcal{N} = \{[\mu_i \ldots \mu_n] \mid \sum_{i=1}^{n} \mu_i = 1, \mu_i \in \{0, 1\}, \forall i = 1 \ldots n\}$$

In this way, the task of choosing the most suitable meme has been formulated as learning the coefficient vector $\boldsymbol{\mu} \in \mathcal{N}, \boldsymbol{\mu} = [\mu_i \ldots \mu_n]$.

Further, we formulate the meme selection problem according to the HSIC and Maximum Mean Discrepancy criteria [6] as follows:

$$\begin{aligned} \max_{\mu} \quad & tr(\mathbf{H}\mathbf{X}^T\mathbf{M}_t\mathbf{X}\mathbf{H}\mathbf{Y}) + \sum_{i=1}^{n} \mu_i S_i \\ \text{s.t.} \quad & \mathbf{M}_t = \sum_{i=1}^{n} \mu_i \mathbf{M}_i \\ & \boldsymbol{\mu} \in \mathcal{N} \\ & \mathbf{M}_i \succeq 0 \end{aligned} \tag{8}$$

where S_i is the similarity measure between two given problem instance. In the context of CARP, $S_i = -(c1 * MMD_i + c2 * Dif_i)$, where MMD_i denotes the Maximum Mean Discrepancy (MMD) [6], which is used to compare the distribution similarity between two given instances by measuring the distance between their corresponding means. $MMD(D_s, D_t) = ||\frac{1}{n_s}\sum_{i=1}^{s}\phi(x_i^s) - \frac{1}{n_t}\sum_{i=1}^{t}\phi(x_i^t)||$, where $\phi(\cdot)$ maps the original input to a high dimensional space. Here, for simplicity, we use linear mapping, so $\phi(\mathbf{x}) = \mathbf{x}$. Dif_i denotes the difference in vehicle capacity for two given CARP instances, while $c1$ and $c2$ are the coefficients to balance the importance of differences in the tasks distribution and vehicle demand. In Equation 8, the first term maximizes the statistical dependence between input $\mathbf{X}$ and output label $\mathbf{Y}$ for clustering [16]. The second term measures the similarity between the past instances and the current new unseen problem of interest.

Taking the constraints into consideration, Equation 8 can be reformulated as:

$$\max_{\boldsymbol{\mu}\in\mathcal{N}} tr(\mathbf{H}\mathbf{X}^{\mathbf{T}}\mathbf{M}\mathbf{X}\mathbf{H}\mathbf{Y}) + \sum_{i=1}^{n} \mu_i S_i \tag{9}$$

$$\Longrightarrow \quad \max_{\boldsymbol{\mu}\in\mathcal{N}} tr(\mathbf{H}\mathbf{X}^T \sum_{i=1}^{n} \mu_i \mathbf{M}_i \mathbf{X}\mathbf{H}\mathbf{Y}) + \sum_{i=1}^{n} \mu_i S_i \tag{10}$$

$$\Longrightarrow \quad \max_{\boldsymbol{\mu}\in\mathcal{N}} \sum_{i=1}^{n} \mu_i (tr(\mathbf{H}\mathbf{X}'\mathbf{M}_i\mathbf{X}\mathbf{H}\mathbf{Y}) + S_i) \tag{11}$$

From Equation 11, it is straightforward to see that, the matrix $\mathbf{M}_i$ that maximizes $tr(\mathbf{H}\mathbf{X}^T\mathbf{M}_i\mathbf{X}\mathbf{H}\mathbf{Y}) + S_i$ denotes the most suitable meme, and gives the corresponding $\mu_i = 1$. With two unknown variables (i.e., $\boldsymbol{\mu}$ and $\mathbf{Y}$) in Equation 11, we first perform clustering (e.g., K-Means) on input $\mathbf{X}$ directly to obtain the label matrix $\mathbf{Y}$. By keeping $\mathbf{Y}$ fixed, we obtained $\boldsymbol{\mu}$ by maximizing $tr(\mathbf{H}\mathbf{X}^T\mathbf{M}_i\mathbf{X}\mathbf{H}\mathbf{Y}) + S_i$. Next, by maintaining the chosen $\mathbf{M}$ fixed, clustering is performed on the new $\mathbf{X}$ (i.e., transformed by selected $\mathbf{M}$. $\mathbf{X}' = \mathbf{L}^T\mathbf{X}$, where $\mathbf{L}$ is obtained by SVD on $\mathbf{M}$) to obtain label matrix $\mathbf{Y}$. The whole process is performed iteratively until convergence is reached.

V. Empirical Study

To verify the effectiveness of the proposed search paradigm, an empirical study conducted on the commonly used CARP benchmark is presented in this section.

A. Experimental Configuration

1) Data Set: The well-established *egl* CARP benchmark is used in the present experimental study. The data set was generated by Eglese based on data obtained from the winter gritting application in Lancashire [17], [18], [19]. This data set has been commonly used as test benchmark in the literature for CARP solving [9], [10]. It consists of two series data sets (i.e., "E" and "S" series) with a total of 24 instances. In particular, CARP instances in "E" series have smaller number of vertices, task or edges than that in "S" series, thus the problem structures of "E" series are deem to be simpler than "S" series.

In what follows, we illustrate the benefits of useful traits reuse from past experiences by taking the state-of-the-art evolutionary search method recently proposed by Mei *et al.* [10] for solving CARP, as the baseline algorithm (labeled here as *ILMA*), which we build upon. In this manner, any improvements attained in the search can be attributed to the meme learning and selection processes in fulfilling the evolutionary optimization across problems. To generate the pool of memes from past experiences, the CARP instance in *egl*,

namely "E1A", "E1B", "E2A", "E3A", "E4A", "S1A", "S1B", "S2A", "S3A", "S4A" are considered here as the previously solved problem instances by using the results reported in [10].

2) ILMA with Variants of Population Initialization Procedures: Four population initialization procedures building on *ILMA* are investigated in the present study. The first is a simple random task assignment during population initialization, which is labeled here as *ILMA-R*. The second is the informed heuristic based population initialization procedure used in baseline *ILMA* [10]. There, the initial population is formed by a fusion of chromosomes generated from Augment_Merge [8], Path_Scanning [20], Ulusoy's Heuristic [21] and the simple random initialization procedures. The third procedure involves clustering strategy commonly used for the assignment of tasks in vehicle routing problems. Particularly, the popular K-Means clustering is considered and notated here by *ILMA-K*. The last is the proposed task assignment via clustering that is guided by knowledge learned from past problem solving experiences, and labeled here as *ILMA-L*.

Further, to facilitate a fair comparison and verify the benefits of the learning from past experiences, the operator settings of *ILMA* and its variants are kept consistent to that of [10].

B. Pre-Processing of CARP

In traditional CARP, each task is represented by a corresponding *head vertex*, *tail vertex*, *travel cost* and *demand (service cost)*. The shortest distance matrix of the vertices is first derived by means of the Dijkstra's algorithm [22], i.e., using the distances available between the vertices of a CARP. The coordinate features (i.e., locations) of each task are then approximated by means of multidimensional scaling [23]. In this manner, each task is represented as a node in the form of coordinates. A CARP instance in the current setting is thus represented by input vector $\mathbf{X}$ composing of the coordinate features of all tasks in the problem. Such a representation would allow the standard clustering approaches, such as the K-Means algorithm to be conducted on the CARP in task assignment. The label information of each task, i.e., in $\mathbf{Y}$ belonging to the CARP instance is given by the optimized solution of baseline *ILMA*.

Furthermore, the MMD of Equation 8 is also augmented with the demand of each task as additional problem feature. Coefficients $c1$ and $c2$ of Equation 8 and parameters p and B of Equation 4 are configured as 0.8, 0.2, 2 and 100 in all the experiments, respectively.

C. Results and Discussion

Table I presents the detailed properties of the unseen or unsolved CARP benchmarks used in the study. "$|V|$", "$|E_R|$", "E" and "LB" denote the number of vertices, number of tasks, total number of edges and lower bound, of each problem instance, respectively. Table II then tabulates the statistical performances of the algorithms considered, including the *ILMA*, *ILMA-R*, *ILMA-K* and our proposed search paradigm *ILMA-L* that learns from past problems solved. Particularly, columns "*B. Cost*", "*Ave.Cost*" and "*Std.Dev*" report the *Best Cost*, *Averaged Cost*, and *Standard Deviation* obtained by the corresponding algorithms across 30 independent runs.

Further, to provide a concise summary on the overall performances of *ILMA* and its variants on the CARP benchmarks, results on two measures, namely $Gap\ Average$ and $Gap\ Best$ are presented in Fig. 5 and Fig. 6, respectively. The $Gap\ Average$ and $Gap\ Best$ are defined as follows:

- $Gap\ Average$ gives the difference between the attained *average cost* value and *lower bound* value of given CARP instance.

$$Gap\ Average = Average\ Cost - Lower\ Bound$$

- $Gap\ Best$ gives the difference between the attained *best cost* value and *lower bound* value of given CARP instance.

$$Gap\ Best = Best\ Cost - Lower\ Bound$$

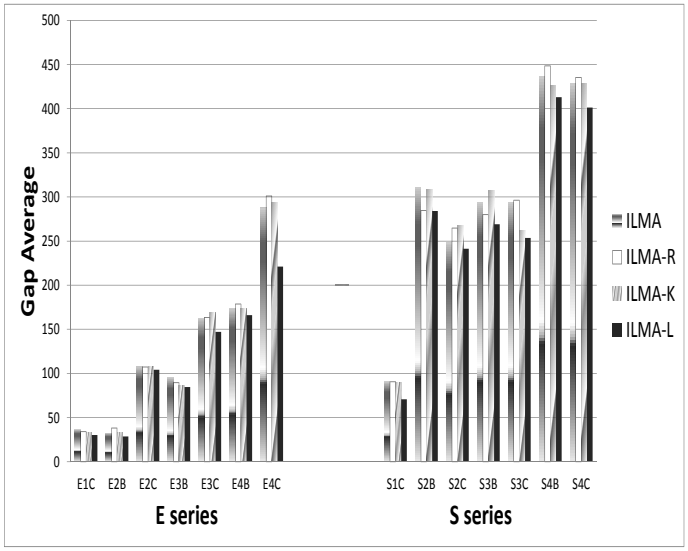


Fig. 5. $Gap\ Average$ of *ILMA*, *ILMA-R*, *ILMA-K* and *ILMA-L* on *egl* CARP benchmarks

As can be observed from Table II and Fig. 5, the *ILMA* is noted to have attained improved performance on solution quality over *ILMA-R* on 8 out of 14 CARP instances, in terms of *Average Cost* and *Gap Average*, respectively. Particularly, on "S4B" and "S4C", which denote the two most complex instances with largest number of vertices, tasks and edges among all the benchmarks considered, *ILMA* demonstrated superior performances, in terms of both *Average* and *Best Cost*. Note that the only difference between *ILMA-R* and *ILMA* lies in the additional heuristics employed for task assignment in the latter. From this, it is possible to infer that the heuristic-based initialization process is useful in narrowing down the search towards improved solutions, particularly so on the large-scale problems.

Clustering has demonstrated notable search performance for task assignment in vehicle routing problems, by taking the structure of task distributions into account during task

TABLE I
PROPERTIES OF THE CARP BENCHMARKS

	"E" Series							"S" Series						
Data Set	E1-C	E2-B	E2-C	E3-B	E3-C	E4-B	E4-C	S1-C	S2-B	S2-C	S3-B	S3-C	S4-B	S4-C
V	77	77	77	77	77	77	77	140	140	140	140	140	140	140
E_r	51	72	72	87	87	98	98	75	147	147	159	159	190	190
E	98	98	98	98	98	98	98	190	190	190	190	190	190	190
LB	5566	6305	8243	7704	10163	8884	11427	8493	12968	16353	13616	17100	16093	20375

TABLE II
STATISTICS OF *ILMA*, *ILMA-R*, *ILMA-K* AND *ILMA-L* ON *egl* CARP BENCHMARKS

Data	$ILMA$			$ILMA-R$			$ILMA-K$			$ILMA-L$		
Set	$B.Cost$	$Ave.Cost$	$Std.Dev$	$B.Cost$	$Ave.Cost$	$Std.Dev$	$B.Cost$	$Ave.Cost$	$Std.Dev$	$B.Cost$	$Ave.Cost$	$Std.Dev$
1.E1-C	5595	5602.10	8.89	5595	5599.87	8.44	5595	5599.07	7.53	5595	**5596.26**	6.93
2.E2-B	6317	6337.20	9.94	6317	6343.03	12.68	6317	6337.63	19.72	6317	**6333.63**	14.59
3.E2-C	8335	8350.60	29.22	8335	8350.13	25.48	8335	8351.23	27.88	8335	**8347.23**	22.73
4.E3-B	7777	7799.40	29.31	7777	7793.50	27.56	**7775**	7790.87	19.53	**7775**	**7788.50**	16.69
5.E3-C	10292	10325.37	28.78	10292	10326.50	40.06	10292	10331.60	48.54	10292	**10310.10**	16.95
6.E4-B	8998	9057.37	37.44	8990	9062.47	53.69	8998	9058.07	34.42	**8988**	**9050.00**	49.81
7.E4-C	11609	11714.60	78.79	11594	11728.03	82.64	11606	11721.20	76.73	**11542**	**11648.13**	65.24
8.S1-C	8519	8583.90	47.68	8518	8583.47	38.65	8518	8582.18	42.21	8518	**8563.67**	37.30
9.S2-B	13190	13278.73	63.81	13167	13252.50	50.14	13158	13276.80	74.88	**13157**	**13252.17**	60.43
10.S2-C	16490	16601.80	77.18	16477	16617.67	77.34	16482	16620.80	82.93	**16456**	**16594.33**	70.35
11.S3-B	13784	13910.27	60.59	13779	13895.93	78.82	13784	13923.50	74.22	**13757**	**13885.07**	78.78
12.S3-C	17285	17393.30	81.13	17282	17396.37	57.03	17291	17362.27	63.25	**17253**	**17353.50**	54.81
13.S4-B	16394	16529.57	66.74	16435	16541.77	70.79	16379	16518.90	66.64	**16358**	**16506.10**	68.53
14.S4-C	20608	20804.17	77.44	20653	20810.47	79.62	20625	20803.47	73.81	**20581**	**20776.43**	74.96

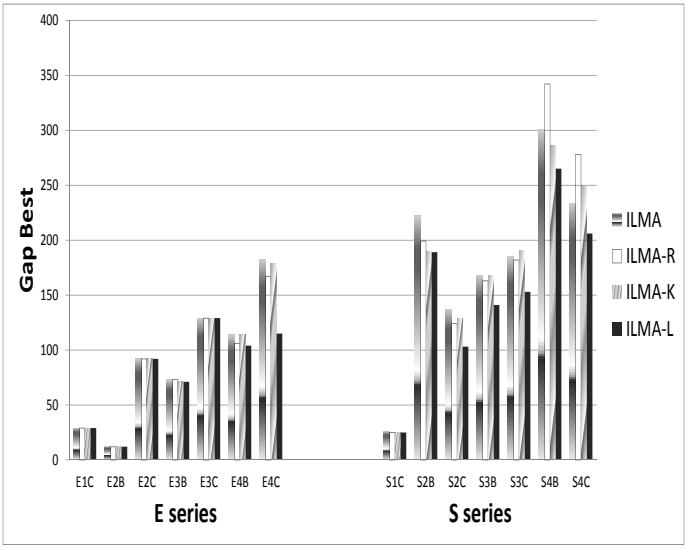


Fig. 6. *Gap Best* of *ILMA*, *ILMA-R*, *ILMA-K* and *ILMA-L* on *egl* CARP benchmarks

assignment. With the clustering of tasks in CARP, *ILMA-K* can be noted to have attained improved solutions on "E3B", "E4C", "S1C", "S2B", "S2C" and "S4B", but poor performance on "S3C" and "S4C" in terms of *Best Cost* or *Gap Best* compared to the heuristic based *ILMA*. This is because clustering being an unsupervised learning approach, does not use any label information available to differentiate between good or bad task assignments. The quality of the clustered tasks would thus depends on the distribution of the tasks in the problem instance. Therefore, if the task distribution models the optimal task assignment well, *ILMA-K* could then attain better solutions than the baseline *ILMA*, such as in the cases of "E3B", "S2B", etc. In the event that the distribution of the tasks does not align well with the notion of good task assignment for the given problem instance, *ILMA* could likely fare poorer than using the simple random task assignment *ILMA-R* (e.g., "E2C", "E3C", etc.).

In our proposed new search paradigm, on the other hand, task assignment via clustering is guided by knowledge learned from past problem solving experiences. Label information of past optimized CARP instances are transferred to the current new unseen problem instance of interest. The information thus serves as previous tasks assignment experiences that could then be imitated to enhance the search on problem instances of similar task distributions. With the proposed new search paradigm, it can be seen from Table II and Fig. 5 that the proposed *ILMA-L* led to superior performances over *ILMA-R*, *ILMA* and *ILMA-K* on all the tested CARP instances in terms of *Average Cost* and *Gap Average*, respectively. Considering also the *Best Cost* solution and *Gap Best* criterions, *ILMA-L* is noted to also attain superior solutions on 8 out of 14 CARP instances, relative to the other algorithms considered.

VI. CONCLUSION

In this paper, we have presented a novel study on the transfer of knowledge in the form of useful structures or latent patterns that are captured from previous experiences of problem-solving to enhance future evolutionary search. In particular, the knowledge has been modeled as the instruction for conducting task assignment in the context of CARP.

Subsequently, the processes of *meme learning* and *meme selection* designed for acquiring structured knowledge from past problem solving experiences and the identifications of suitable knowledge to bias the task assignment on future CARP instances, respectively, have been presented in details. In contrast to existing works, the proposed novel search paradigm enables knowledge transfer and reuse in evolutionary optimization across problems of different size, structure, or representation, etc. Empirical study conducted on commonly used CARP benchmarks confirmed the effectiveness of the proposed new paradigm for enhancing evolutionary search with past problem solving experiences.

ACKNOWLEDGMENT

This research is partially supported by the Singapore National Research Foundation under its Interactive & Digital Media (IDM) Public Sector R&D Funding Initiative and administered by the IDM Programme Office.

REFERENCES

[1] Q. C. Nguyen, Y. S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 604–623, 2009.

[2] 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, In Press*, 2011.

[3] S. J. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, pp. 316–328, 2004.

[4] P. Cunningham and B. Smyth, "Case-based reasoning in scheduling: Reusing solution components," *The International Journal of Production Research*, vol. 35, no. 4, pp. 2947–2961, 1997.

[5] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," *Proceedings Algorithmic Learning Theory*, pp. 63–77, 2005.

[6] K. M. Borgwardt, A. Gretton, M. J. Rasch, H. P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Proceedings of the 14th International Conference on Intelligent Systems for Molecular Biology*, pp. 49–57, 2006.

[7] R. Dawkins, "The selfish gene," *Oxford: Oxford University Press*, 1976.

[8] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.

[9] L. Feng, Y. S. Ong, Q. H. Nguyen, and A.-H. Tan, "Towards probabilistic memetic algorithm: An initial study on capacitated arc routing problem," *IEEE Congress on Evolutionary Computation*, pp. 18–23, 2010.

[10] Y. Mei, K. Tang, and X. Yao, "Improved memetic algorithm for capacitated arc routing problem," *IEEE Congress on Evolutionary Computation*, pp. 1699–1706, 2009.

[11] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Competitive memetic algorithms for arc routing problem," *Annals of Operational Research*, vol. 141, no. 1–4, pp. 159–185, 2004.

[12] N. Yoshiike and Y. Takefuji, "Vehicle routing problem using clustering algorithm by maximum neural networks," *Intelligent Processing and Manufacturing of Materials*, vol. 2, pp. 1109 – 1113, 1999.

[13] X. S. Chen, L. Feng, and Y. S. Ong, "A self-adaptive memeplexes robust search scheme for solving stochastic demands vehicle routing problem," *International Journal of Systems Science*, 2011.

[14] J. Zhuang, I. Tsang, and S. C. H. Hoi, "A family of simple non-parametric kernel learning algorithms," *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 1313–1347, 2011.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[16] L. Song, A. Smola, A. Gretton, and K. M. Borgwardt, "A dependence maximization view of clustering," *Proceedings of the 24th international conference on Machine learning*, pp. 815–822, 2007.

[17] R. W. Eglese, "Routing winter gritting vehicles," *Discrete Applied Mathematics*, vol. 48, no. 3, pp. 231–244, 1994.

[18] R. W. Eglese and L. Y. O. Li, "A tabu search based heuristic for arc routing with a capacity constraint and time deadline," *in Metaheuristics: theory and applications, I. H. Osman and J. P. Kelly, Eds. Boston: Kluwer Academic Publishers*, pp. 633–650, 1996.

[19] L. Y. O. Li and R. W. Eglese, "An interactive algorithm for vehicle routing for winter-gritting," *Journal of the Operational Research Society*, vol. 47, no. 2, pp. 217–228, 1996.

[20] B. L. Golden, J. S. DeArmon, and E. K. Baker, "Computational experiments with algorithms for a class of routing problems," *Computer & Operation Research*, vol. 10, no. 1, pp. 47–59, 1983.

[21] G. Ulusoy, "The fleet size and mix problem for capacitated arc routing," *Eur. J. Oper. Res.*, vol. 22, no. 3, pp. 329–337, 1985.

[22] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[23] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.