8-2013

FloTra: Flower-shape trajectory mining for instance-specific parameter tuning

Lindawati LINDAWATI
Singapore Management University, lindawati.2008@smu.edu.sg

Feida ZHU
Singapore Management University, fdzhu@smu.edu.sg

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

FloTra: Flower-shape Trajectory Mining for Instance-specific Parameter Tuning

Lindawati, Feida Zhu, Hoong Chuin Lau

Singapore Management University
80 Stamford Road, 178902, Singapore
lindawati, fdzhu, hclau@smu.edu.sg

1 Introduction

Meta-heuristic algorithms play an important role in solving Combinatorial Optimization Problems (COP) in many real-life applications. The caveat is that the performance of a meta-heuristic algorithm is highly dependent on its parameter configuration which controls the algorithm behavior. Furthermore, finding the optimal parameter configuration, especially instance-specific configuration, is often a difficult, tedious and frustrating task. Among the proposed approaches for automated parameter tuning, CluPaTra [3] and SufTra [4] address the requirements of generic instance-specific automated parameter tuning. It introduces the notion of *search trajectory* as a generic feature. Search Trajectory, modeled as a sequence, is a series of solutions discovered by meta-heuristic algorithm as it searches for the best solutions over its neighborhood search space.

Although CluPaTra and SufTra have been proven to give a significant improvement over one-size-fits-all approach, it suffers from descriptiveness issue due to their sequence representation model. CluPaTra and SufTra may oversimplify the search trajectory and lose finer-granularity details in some structural patterns. For example, Fig. 1 shows the sequence and graph representation for three search trajectories of Quadratic Assignment Problem (QAP) instances. The three sequences have many similar subsequences (Fig. 1a) but the real search trajectories (as shown in Fig. 1b) are different; two search trajectories have a smoother search while the other one has many cycles.

In this work, we introduce FloTra, a technique to uncover important patterns from *search trajectory* graph for generic instance-specific automated parameter tuning. FloTra is an extension of CluPaTra and SufTra that overcomes their limitation on descriptiveness. FloTra constructs a graph representation of search trajectory and conducts a graph pattern mining to discover specific and important patterns in search trajectory. Using these patterns, FloTra then clusters the instances and computes a corresponding optimal parameter configuration for each cluster. We have applied our approach on QAP and SCP and show that FloTra gives an encouraging improvement for the overall performance.

2 Solution Approach

FloTra is designed as a cluster-based instance-specific automated parameter tuning framework which works in two stages: training and testing. The first step in the training phase is the clustering process where we record the search trajectory of an instance and transform it into a graph. We continue to extract relevance features, calculate similarities and perform clustering. We then apply a tuning procedure to derive the best parameter configuration for each cluster. In the testing phase, we match the search trajectory of testing instance against the clusters to find the most similar cluster. We then return the parameter configuration found for that cluster (during the training phase) as the recommended parameter configuration. In this paper, we focus on clustering process by introducing a new feature extraction method. For tuning, we simply use the existing one-size-fits-all configurator, ParamILS [1].

In search trajectory graph, a node is a solution presented as a symbol containing two solution attributes: position type and performance metric [4], and an edge is the movement from one solution to another. Search trajectory graph is a special graph that has two distinctive structures which are: (1) a long skinny path representing solution movement from an initial solution to an end solution and (2) multiple short paths and loops representing cycles to/from local optimal. We consider the skinny long path as a

stem and the short paths and loops as petals and thorns. To differentiate petals and thorns from stem, we assume that petal and thorn length should be shorter than that of stem.

FloTra aims is to find a set of frequent patterns (subgraphs) from a set of search trajectory graphs. For that, FloTra has two parameters: $\min_{length}$ and $\min_{support}$. $\min_{length}$ determines the minimum length of subgraph (which is translated to a minimum length of a stem and maximum length of thorns and petals) whereas $\min_{support}$ determines minimum number of graphs that contains a frequent subgraph. In this paper, the values of $\min_{length}$ and $\min_{support}$ are fixed beforehand.

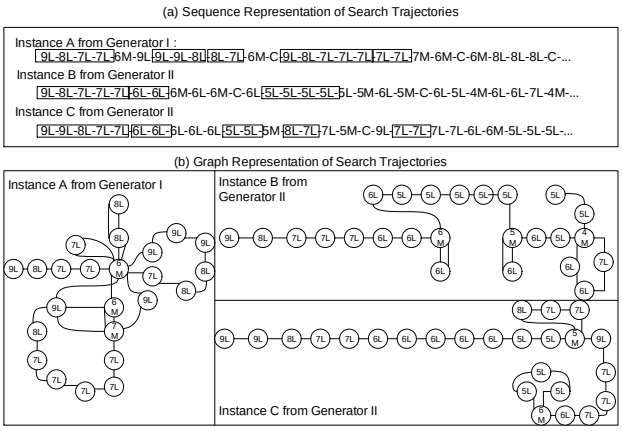

Figure 1: Search Trajectories Representation for 3 Quadratic Assignment Problem (QAP) instances, (a) sequence representation and (b) graph representation.

FloTra works in three stages. It first mines short frequent paths (thorns and petals) from all nodes, except the initial node. It then continues to mine long skinny path (long stem) from initial node. After having a set of thorns, petals and stems, FloTra then assembles the thorns, petals and stems together. The detail is as follows:

Stage 1: Mining Flower Thorns and Petals. To find petals and thorns, we only select nodes which are visited more than once in the search process. Hence, the number of edges must be greater than one. We first enumerate all the paths from the selected nodes using Depth-First Search (DFS) algorithm. For paths with length less than $\min_{length}$, we construct a Suffix Tree structure as in SufTra. This suffix tree is used to mine similar thorns and petals across different instances. To avoid redundancy, we only insert the same path once and run a checking mechanism before we insert it. We then retrieve frequent substrings from different search trajectory graphs that occur more than $min_{support}$ as frequent patterns for flower thorns and petals.

Stage 2: Mining Long Stem. Aside from flower thorns and petals, another important structure that we want to retrieve is a long stem structure. The process is similar to stage 1. We first enumerate all paths from initial node using a DFS algorithm. For paths with length equal or more than $\min_{length}$, we construct a Suffix Tree and find all frequent paths. We retrieve the frequent substrings from different search trajectory graphs that occur more than $min_{support}$ as frequent patterns for long stem.

Stage 3: Assembling the Flower. At this stage, we assemble the flower thorns and petals from stage 1 with the long stem set from stage 2. For each long stem set that contains the node in flower thorn and petal set, we attach the flower thorn and petal and consider it as a new candidate pattern. If the new candidate occurs no less than $min_{support}$ times, we accept it as a frequent pattern. Because frequent paths from both previous stages are generated from multiple segments in search trajectory, the assembling process may discover some gaps among those frequent paths. We allow these gaps and calculate the gap minimum and maximum number of node in between. After assembling the flower, we set all the found frequent pattern features if it occurs in at least $min_{support}$ number of graphs.

Table 1: Parameters for QAP and SCP

Problem	Parameter	Range
QAP	Temp	[100,5000]
	Alpha	[0.1,0.9]
	Length	[1,10]
	Pct	[0.01,0.1]
SCP	fTSLength	[1000,10000]
	iNonImprove	[5,200]
	iProbRandom	[1,20]
	iDeterministic	[0,1]

As in SufTra [4], we calculate the similarity for each pair of instances by Cosine Similarity. With the similarity score for each pair of instances, we cluster the instances using a well-known hierarchical clustering approach AGNES (AGglomerative NESting) with L method to determine the cluster number.

We then tune each cluster using existing one-size-fits-all configurator, ParamILS [1].

3 Experiment Result

We conducted an experiment on Quadratic Assignment Problem (QAP) using a hybrid Simulated Annealing and Tabu Search (SA-TS) algorithm [6] and Set Covering Problem using tabu search [5]. These two algorithms have four parameters to tune as shown in Table. 1. We use 500 generated instances for QAP and 80 instances from [2] for SCP. We compared the target algorithm performance using parameter configuration from SufTra and FloTra, as well as the existing instance-specific configurator ISAC [2] and one-size-fits-all configurator ParamILS [1]. Since ISAC requires problem-specific features, we used 2 features for QAP: *flow dominance* and *sparsity*. For SCP, we use clusters in [2].

Table 2: Performance Result

	Technique	Training	Testing	p-value
QAP	ParamILS	1.07	2.12	
	ISAC	0.83	1.12	
	SufTra	0.81	1.16	
	FloTra	**0.78**	**1.07**	0.0421
SCP	ParamILS	1.53	0.82	
	ISAC	0.42	0.77	
	SufTra	0.35	0.78	
	FloTra	**0.27**	**0.52**	0.0362

We show the average of percentage deviation from best known value in Table. 2. We also perform a t-test between SufTra and FloTra result and consider p-value below 0.05 to be statistically significant (confidence level 5%). Notice that FloTra outperforms other methods in both training and testing instances.

4 Conclusion

In this paper, we proposed a new generic instance-specific parameter tuning via clustering of patterns according to instance search trajectories graph using FloTra technique as a feature extraction technique. We verify FloTra's performance on QAP and SCP and observed a promising improvement compared to ISAC and SufTra. Up to this stage of our work, FloTra can only be applied to local search algorithms, since it uses local search trajectory as the generic feature. As future our work, we will investigate how to generate clusters from population-based-algorithm using generic features pertaining to population dynamics.

References

[1] F. Hutter, H.H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[2] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. Isac: Instance-specific algorithm configuration. In *19th European Conference on Artificial Intelligence*, 2010.

[3] Lindawati, H.C. Lau, and D. Lo. Clustering of search trajectory and its application to parameter tuning. *JORS Special Edition: Systems to Build Systems*, 2012.

[4] Lindawati, Yuan Zhi, H.C. Lau, and F. Zhu. Automated parameter tuning framework for heterogeneous and large instances: Case study in quadratic assignment problem. In *LNCS: 7nd Learning and Intelligent OptimizatioN Conference*, 2013.

[5] N. Musliu. Local search algorithm for unicost set covering problem. In *LNCS: Advances in Applied Artificial Intelligence*, 2006.

[6] K.M. Ng, A. Gunawan, and K.L. Poh. A hybrid algorithm for the quadratic assignment problem. In *International Conf. on Scientific Computing*, pages 14–17, 2008.