Computers and Operations Research 94 (2018) 1–10

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

An Efficient Implementation of a Static Move Descriptor-based Local Search Heuristic

Onne Beek[a], Birger Raa[a,*], Wout Dullaert[b], Daniele Vigo[c,b]

[a] Department of Industrial Systems Engineering and Product Design, Ghent University, Belgium
[b] Department of Information, Logistics and Innovation, Vrije Universiteit Amsterdam, Belgium
[c] Department of Electrical, Electronic, and Information Engineering, University of Bologna,Belgium

ARTICLE INFO

Article history:
Received 15 September 2017
Revised 11 December 2017
Accepted 6 January 2018

Keywords:
Efficient Local Search
Vehicle Routing
Static Move Descriptors
Heuristic Priority Queue

ABSTRACT

This paper proposes several strategies for a more efficient implementation of the concept of Static Move Descriptors (SMDs), a recently developed technique that significantly speeds up Local Search-based algorithms. SMDs exploit the fact that each local search step affects only a small part of the solution and allow for efficient tracking of changes at each iteration, such that unnecessary reevaluations can be avoided. The concept is highly effective at reducing computation times and is sufficiently generic to be applied in any Local Search-based algorithm. Despite its significant advantages, the design proposed in the literature suffers from high overhead and high implementational complexity. Our proposals lead to a much leaner and simpler implementation that offers better extendibility and significant further speedups of local search algorithms. We compare implementations for the Capacitated Vehicle Routing Problem (CVRP) - a well-studied, complex problem that serves as a benchmark for a wide variety of optimization techniques.

1. Introduction

Local Search is one of the main optimization techniques used to tackle NP-hard problems. Its popularity comes from its simplicity: by iteratively applying small changes to a solution, it thoroughly explores the solution space surrounding the current (or 'incumbent') solution and gradually improves towards a local optimum. The nature of these small changes makes Local Search a very flexible technique. Local Search operators can vary from very basic operators that affect only a few solution characteristics to complex subroutines that combine multiple changes to perform a significant restructuring of the solution. These operators embody a trade-off: complex operators can perform a more extensive search of the solution space and thus can reach higher quality solutions, but the number of possible changes and thus the effort required to find improving changes increases with their complexity. As a result, most Local Search procedures usually only adopt operators that affect a very small number of characteristics.

Local Search operators are also used as building blocks in metaheuristic solution approaches. Pure Local Search metaheuristics generally make use of a larger set of simple operators, combined with a guiding strategy to steer the search out of local optima. The potential of these pure Local Search metaheuristics has been demonstrated by multiple powerful algorithms, such as Tabu Search (Glover, 1989), which allows worsening moves and prevents cycling by blocking moves with specific ('tabu') characteristics; Guided Local Search (Voudouris and Tsang, 1999), which guides the search away from local optima by penalizing undesirable, or overly frequent, characteristics; and Variable Neighborhood Search (Mladenović and Hansen, 1997), which escapes from local optima of one operator by invoking another. All these metaheuristics are capable of diversifying the search using only Local Search operators.

In contrast to pure Local Search-based metaheuristics, there is also widespread use of Local Search in so-called hybrid metaheuristics. These algorithms use the power of Local Search as a tool for intensification - a strong, localized search to improve a solution without drastically altering its structure. This Local Search phase is then alternated with a diversification method. Popular examples of such hybrid metaheuristics are Memetic Algorithms (Norman and Moscato, 1989), where the Local Search is wrapped into an evolutionary, population-based optimization framework, and Iterated Local Search (Martin et al., 1992), which applies a strongly disruptive *shake* method to a solution stuck in a local optimum, and then restarts the Local Search on this diversified solution.

* Corresponding author at: Technologiepark 903, 9052 Zwijnaarde, Ghent, Belgium.
E-mail address: Birger.Raa@UGent.be (B. Raa).

https://doi.org/10.1016/j.cor.2018.01.006
0305-0548/© 2018 Elsevier Ltd. All rights reserved.

The underlying principle of these hybrid algorithms is clear: use Local Search to improve the incumbent solution, then apply a different method to the solution to escape from the local optimum. Although many authors focus on novel ideas for the diversification, it is still the Local Search phase that transforms candidate solutions into high-quality solutions. E.g., Nagata and Bräysy (2008) note that their powerful memetic algorithm spends 80-90% of its time on the Local Search phase. Johnson and McGeoch (1997) note that in case of the hybrid GA, spending more time on the Local Search phase is more valuable than increasing the population size, as shown by the 'population of one' genetic algorithm that lead to Iterated Local Search.

Finding efficient ways of performing Local Search thus has a critical effect on the performance of any heuristic or metaheuristic algorithm. In this paper, we therefore focus on this important and often overlooked aspect. We do this by applying Local Search to the well-known Capacitated Vehicle Routing Problem, a fundamental model for transportation problems with multiple vehicles. Given a set of locations each requiring the delivery of a given volume, the goal is to design a set of routes so that each location is visited exactly once in the minimal total distance or time travelled. Every route is limited to a maximum capacity (storage space, service time, etc.). The VRP is conceptually simple, but computationally challenging. Its practical relevance and computational challenges make the VRP an excellent problem for benchmarking experiments.

Local Search is a popular choice of optimization technique for many variants of the VRP. The problem structure of the VRP can be exploited in relatively simple Local Search operators that yet prove to be very powerful. E.g., moving a customer to a different position in the visiting sequence, or swapping two customers in the sequence, are simple operators that are easy to evaluate both in terms of how they change the solution quality and in terms of solution feasibility. Additionally, the scope of such Local Search operators can easily be limited (e.g., to nearby customers only) in order to speed up the search (see Section 2.2).

In the following section, we will explain the strengths and challenges of efficient Local Search strategies applied to the VRP. In Section 3 we discuss the concept of Static Move Descriptors, a technique recently introduced by Zachariadis and Kiranoudis (2010) to speed up Local Search. We analyze the performance and identify the drawbacks of their implementation in Section 4. In Section 5, we outline the changes we made in order to create a leaner, simpler implementation that outperforms the original. In Section 6 we empirically show the benefits of our version over the original by performing extensive experiments on VRP benchmark instances, followed by concluding remarks in Section 7.

2. Efficient Local Search

The application of Local Search is straightforward for most problems. As soon as a suitable solution representation is determined, operators such as moving or swapping elements in the solution can quickly be implemented. A common, yet naive implementation of Local Search simply loops over all possible moves for a set of operators in order, identifies the best feasible move and applies the corresponding change to the solution; this cycle repeats until no more improving moves can be found.

As an example, consider the Swap operator for the CVRP that swaps the position of two locations in the sequence. This operator has a cardinality of $O(n^2)$ since any combination of two locations unambiguously defines a specific move. Enumerating all Swap moves is as simple as having two nested loops go over all locations. Similarly, the Relocate operator moves a location to a different position in the sequence. This operator also has a cardinality of $O(n^2)$ since any location could be moved to any position in the sequence. A naive Local Search implementation would evaluate the effect on the solution and feasibility of all Swap and Relocate moves (possibly along with other Local Search moves that have been identified).

The goal of Efficient Local Search, and also the main contribution this paper wants to make, is to come up with a less naive, more powerful implementation of Local Search that reduces the computational effort required to achieve high-quality solutions. In previous research, various techniques to accomplish this goal have already been suggested, each with their own advantages and disadvantages. These techniques can be categorized into three groups: Acceptance and Search strategies, Candidate Set strategies and Locality Tracking strategies.

2.1. Acceptance and Search strategies

As explained above, a naive implementation evaluates all possible moves at each iteration, after which the best feasible move is selected and applied to the solution. This known as the 'best-accept' strategy. An alternative, known as the 'first-accept' strategy, is to immediately apply an improving feasible move as soon as it is encountered without first evaluating all other possible moves. This can drastically reduce the search time per iteration. However, the downside is that smaller improvements are applied and thus more iterations are usually required to reach a similar solution quality. Therefore, an actual speedup using the first-accept strategy can only be achieved if the order in which moves are evaluated somehow corresponds to the moves' improvement potential, i.e., if the best improving moves are encountered first. This can be achieved with a clever Search strategy. A good example of this is the Sequential Search of Irnich et al. (2006), in which edges are sorted by their cost in a pre-processing step. This allows the Local Search to look at moves involving nearby nodes first and effectively discards (partial) non-improving moves (involving nodes that are far apart). The one-time pre-processing step requires $O(n^2 \log n)$ time, but leads to significant speedups: commonly used $O(n^2)$ operators reach speedups of a factor 100, whereas the $O(n^3)$ operator 3-Opt obtains a speedup factor up to 14,000 under ideal circumstances.

Another common search strategy is to only consider one operator at a time and apply best-accept per operator, which became known as the Variable Neighborhood Descent (VND) strategy (Mladenović and Hansen, 1997). Only a single operator is evaluated until no more improving moves are found. Then, the search switches to a different operator. The underlying insight is that a local optimum for one operator is not necessarily locally optimal for another operator. This strategy works particularly well when operators of different cardinality are used: first, the simple operators are exhausted and only then the higher-order operators are used.

2.2. Candidate Set strategies

The search strategy determines the order in which moves are evaluated, and whether a single or multiple operators are considered at once. However, even with first-accept, the entire search space has to be evaluated eventually. This can be very expensive, especially for large scale instances. Candidate Set strategies therefore limit the search to a subset of moves that seem more promising, or conversely, skip the evaluation of moves that are deemed less likely to result in improvements.

For the Traveling Salesman Problem, Lin and Kernighan (1973) introduced the K-nearest neighbors strategy, in which the search only considers the K nearest neighbors of a vertex, i.e., the K cheapest incident edges (with $K < n$). Johnson and McGeoch (1997) extended this strategy by selecting the K-nearest neighbors of each quadrant around a vertex, ensuring the existence of