Towards a Formal Specification of Local Search Neighborhoods from a Constraint Satisfaction Problem Structure

Mateusz Ślażyński
AGH University of Science and Technology
Poland
mslaz@agh.edu.pl

Salvador Abreu
University of Évora and LISP
Portugal
spa@uevora.pt

Grzegorz J. Nalepa
AGH University of Science and Technology
Poland
gjn@agh.edu.pl

ABSTRACT

Neighborhood operators play a crucial role in defining effective Local Search solvers, allowing one to limit the explored search space and prune the fitness landscape. Still, there is no accepted formal representation of such operators: they are usually modeled as algorithms in procedural language, lacking in compositionality and readability. In this paper we outline a new formalization capable of representing several neighborhood operators eschewing their coding in a full Turing complete language. The expressiveness of our proposal stems from a rich problem representation, as used in Constraint Programming models. We compare our system to competing approaches and show a clear increment in expressiveness.

CCS CONCEPTS

• **Software and its engineering → Constraint and logic languages**; • **Theory of computation → Optimization with randomized search heuristics**; Program schemes; • **Computing methodologies** → *Search methodologies*;

KEYWORDS

Local Search, Neighborhood, Constraint Programming

ACM Reference Format:
Mateusz Ślażyński, Salvador Abreu, and Grzegorz J. Nalepa. 2019. Towards a Formal Specification of Local Search Neighborhoods from a Constraint Satisfaction Problem Structure. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3321968

1 INTRODUCTION

Local Search is a family of heuristic algorithms, typically used to find approximate solutions for hard optimization problems. The common core of those methods consists in finding an initial suboptimal solution (called configuration in the rest of the paper) and iteratively improving it by exploring similar configurations, called neighbors. The most efficient neighborhood operators are problem specific and have to be defined and tuned by hand, often directly in the form of a computer program. Such an approach suffers from poor reusability and does not generalize well to new domains. The goal of this paper is to introduce a new formal representation for neighborhood operators, one which exploits the structure of the problem under consideration.

2 NDL LANGUAGE

The NDL (Neighborhood Definition Language, pronounced *noodle*) language discourses over points in the search space of a heuristic search process. It is designed to exploit syntactical structure of a problem represented as a Constraint Programming model.

All first-order terms in the NDL language operate on a modified constraint graph (called the *Typed Constraint Graph*), enriched with vertex and edge labels corresponding accordingly to variables' and constraints' types identified in the problem. We say that variables share a type when they occur in the same data structure (e.g. array) at the modeling language level. Constraints are labeled based on the context, they have been stated in, i.e. if they were asserted in a same aggregation function or came from decomposition of a same global constraint.

The atomic NDL operations are called moves and consist of selectors and modifiers chained via functional composition. Selector is a non-deterministic function able to focus on a specific part of the typed constraint graph, selecting variable or constraint based on their type and position in the graph. Modifiers perturb the current configuration by changing value of the selected variables. Due to the non-deterministic nature, single move is able to express the whole neighborhood, but is limited to changes of a fixed size only as there is no control structures involved in the process.

To express more interesting neighborhood operators, moves can be combined by means of the second-order combinators:

- *Selector Quantifiers* exploit selectors' non-determinism and performs a given move over all possible results of given selector function.
- *The Least Fixpoint Operator* explores graph in a breadth-first search manner, performing given move on each visited edge (constraint).

In general, move combinators are primitively recursive and apply specific moves repeatedly over different parts the graph, bounded eventually to finish because of its' finite size. Despite this limitation, they are sufficient to represent non-trivial neighborhood of variable size, like 2-opt operator used commonly in traveling salesman problem or kempe chain, a popular graph coloring neighborhood operator.

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3321968

3 COMPARISON WITH RELATED SOLUTIONS

There is existing work which aims at extracting helpers for a local search procedure, from the structure of the underlying problem. However, many of the existing solutions are incomparable to NDL because of the different design goals. Neighborhood Combinators [4] extend OscaR [6] solver with a declarative language enabling user to combine existing neighborhoods, but with no means to define new operators. On the other hand, recently there has been proposed system based on the Essence modeling language [2], which samples many promising operators using a portfolio strategy, avoiding representing the complex neighborhoods explicitly. Rest of this section will focus on two, most relevant systems and compare them with NDL in terms of the expressive power.

OptaPlanner [1] is a hybrid solver configurable by means of XML files. It supports several search strategies, including local search, and allows to define the neighborhood in a declarative manner as a composition of basic moves. There are several predefined moves applicable to common optimization problems, but to add new ones, one has to implement them in the Java language.

The Neighborhood Definition Language shares many similarities with the OptaPlanner's XML approach. In both languages the moves are composed of variable/value selectors and modifiers ("move selectors" in the OptaPlanner terminology). Besides the basic perturbations, like a value swap/change, OptaPlanner also defines more complex domain-specific move selectors like 2-opt or group swaps/changes that can be used to implement row/column group operations. Another noticeable difference is that selectors do not exploit the constraint structure and can only refer to variable types and values. The lack of more generic combinators makes it impossible to define new complex neighborhoods such as kempe chain. Consequently, OptaPlanner definitions are more coarse-grained, requiring several interesting and useful operators to be directly implemented in the low-level Java programming language.

Declarative Neighborhoods [3] are designed to extend MiniZinc [5] with neighborhood operators represented in a declarative manner. The neighborhood operator is represented as a constraint satisfaction problem itself, with constraints corresponding to relations between the neighbors. Such representation integrates well into a Constraint Programming model, allowing one to put additional requirements on the neighborhood relation, i.e. every neighbor has to satisfy a set of constraints or that the neighborhood operator requires some constraints to be satisfied before it may be applied. This way, the neighborhood can be easily pruned and specific constraints can explicitly be made invariant. At the same time, such a pruning relies on solving a constraint satisfaction problem, which in the general case is an NP-complete procedure and may be computationally prohibitive. The main disadvantage of the *Declarative Neighborhoods* representation, is that neighborhood operator (being a Constraint Programming model itself), cannot involve any kind of recursion, even one as limited as the move combinators approach used in NDL. This greatly restricts the expressiveness of the language, limiting it only to a fixed number of perturbations, similar to a single NDL move. The language may still be extended to include more complex moves (as it is done in OptaPlanner) like column or row swaps, with the restriction that they perform only a fixed number of changes.

Table 1 compares the expressiveness of the aforementioned modeling languages on wide known neighborhood operators.

Table 1: Short comparison of the expressiveness based on the presented examples. Declarative Neighborhoods requires the column swap to be re-implemented per problem instance.

	NDL	OptaPlanner	DN
kempe chain	✓	x	x
column swap	✓	✓	✓
2-opt	✓	✓	x

4 SUMMARY

In this paper we have presented the elements of the Neighborhood Definition Language — a formal language capable of representing the Local Search neighborhood operators in a fine-grained manner. Compared to the general programming languages, NDL has a well defined semantics and is limited to primitive recursion, effectively leading to always terminating total programs. Compared to other declarative approaches, it is much more self-contained and expressive enough to represent even complex neighborhood operators. Such results have been achieved partly with a rich problem representation borrowed from the Constraint Programming formulation, and partly with a limited set of recursion schemes, effectively exploring a problem's structure.

Future research will focus on automated methods for finding problem-specific NDL operators, by means of program synthesis algorithms. Most notably, there is ongoing work on combining Genetic Programming algorithms with type information embedded in the model, by means of Grammar Evolution and related techniques. Finally, a prototype implementation is being worked on, allowing for some experimentation and evaluation of the NDL performance, to specify neighborhoods.

At the same time, we are pursuing the integration with modern modeling languages and solvers. The most important issues include extraction methods, capable of creating a Typed Constraint Network based on the Constraint Programming models and injecting arbitrary NDL operators into currently used Local Search routines.

REFERENCES

[1] [n. d.]. OptaPlanner User Guide. ([n. d.]). https://docs.optaplanner.org/7.15.0.Final/optaplanner-docs/html_single/index.html

[2] Özgür Akgün, Saad Attieh, Ian P. Gent, Christopher Jefferson, Ian Miguel, Peter Nightingale, András Z. Salamon, Patrick Spracklen, and James Wetter. 2018. A Framework for Constraint Based Local Search using Essence,. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

[3] Gustav Björdal, Pierre Flener, Justin Pearson, Peter J. Stuckey, and Guido Tack. 2018. Declarative Local-Search Neighbourhoods in MiniZinc. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. 98–105. https://doi.org/10.1109/ICTAI.2018.00025

[4] Renaud De Landtsheer, Yoann Guyot, Gustavo Ospina, and Christophe Ponsard. 2018. Combining Neighborhoods into Local Search Strategies. In *Recent Developments in Metaheuristics*. Springer, Cham, 43–57. https://doi.org/10.1007/978-3-319-58253-5_3

[5] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. 2007. MiniZinc: Towards a Standard CP Modelling Language. In *Principles and Practice of Constraint Programming – CP 2007*, Christian Bessière (Ed.). Vol. 4741. Springer Berlin Heidelberg, Berlin, Heidelberg, 529–543. https://doi.org/10.1007/978-3-540-74970-7_38

[6] OscaR Team. 2012. *OscaR: Scala in OR.*