ALNS: a Python implementation of the adaptive large neighbourhood search metaheuristic

**Niels A. Wouda** ●[1]¶ **and Leon Lan** ●[2]

1 Department of Operations, University of Groningen, Groningen, The Netherlands
2 Department of Mathematics, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
¶ Corresponding author

Summary

The alns Python package provides a complete implementation of the adaptive large neighbourhood search (ALNS) metaheuristic algorithm (Pisinger & Røpke, 2010; Røpke & Pisinger, 2006). ALNS has quickly become a favourite in the field of operations research for solving difficult combinatorial problems, including the vehicle routing problem and various scheduling problems. Our package has an easy-to-use API and includes various stopping criteria, a large set of acceptance criteria based on Santini et al. (2018), and multiple operator selection schemes. Furthermore, it supports many other single-trajectory neighbourhood search algorithms as special cases, including iterated local search (ILS), variable neighbourhood search (VNS), and the greedy randomised adaptive search procedure (GRASP). The package has already been successfully used for research into methodological improvements of ALNS itself (Reijnen et al., 2022), and for the development of a high-quality ALNS metaheuristic solving an industry problem (Wouda et al., 2023). Because of this success, we expect the package to be useful to the wider operations research community.

Statement of need

Several software libraries exist that facilitate the implementation of metaheuristics (Parejo et al., 2012). The most widely-used libraries generally focus on population-based evolutionary algorithms and multi-objective optimization, for example DEAP (Fortin et al., 2012), ECJ (Scott & Luke, 2019), jMetal (Durillo & Nebro, 2011), Metaheuristics.jl (Mejía-de-Dios & Mezura-Montes, 2022), and PySwarms (Miranda, 2018). As such, these libraries provide limited functionality for single-trajectory algorithms like ALNS. Among libraries that focus on single-trajectory algorithms are Paradiseo (Dreo et al., 2021), and Chips-n-Salsa (Cicirello, 2020). Neither provides an implementation of the ALNS algorithm out of the box. Finally, the ALNS library by Santini (2019) is a problem-agnostic implementation of ALNS in C++. This library provides a number of acceptance criteria, but supports only the roulette wheel operator selection scheme of Røpke & Pisinger (2006).

Despite the existence of these software libraries, it remains common in the operations research community to re-implement heuristics (Swan et al., 2022). Such implementations are relatively limited, are typically tied tightly to one particular problem domain, and often implement just a single acceptance criterion and operator selection scheme. The survey of Windras Mara et al. (2022) corroborates these claims: 205 out of the 251 papers they survey only consider a simulated annealing acceptance criterion, and only one paper uses an operator selection scheme that is not based on the roulette wheel mechanism of Røpke & Pisinger (2006). This inhibits experimentation with different aspects of the algorithm, and makes re-use by others or in other problem domains difficult. Our alns package, by contrast, offers a clear and problem-agnostic API for using the ALNS algorithm, and provides many acceptance criteria and

operator selection schemes. Additionally, we provide diagnostic statistics, plotting methods, logging, and the ability to register custom callbacks at various points of the search. These allow researchers and practitioners to rapidly develop state-of-the-art metaheuristics in a wide range of problem domains.

Features

At its core, ALNS is an iterative ruin-and-recreate algorithm that runs until some stopping criterion is met. The algorithm starts with some initial solution. In each iteration, the current solution is transformed into a new candidate solution using problem-specific destroy and repair operators, which are selected via an operator selection scheme. The candidate solution is then evaluated by an acceptance criterion, and possibly replaces the current solution. Based on the outcome of that evaluation, the operator selection scheme updates the likelihood that the applied operators are selected again in the next iteration.

The `alns` Python package offers:

- A complete ALNS implementation, supported by an extensive test suite. This implementation supports user-defined callbacks whenever a new solution is found, including when that new solution is a new global best, which could be used to support additional intensification methods. Furthermore, it can be supplied with arbitrary user-defined destroy and repair operators that are tailored to the user's problem domain.
- Multiple acceptance criteria in `alns.accept`. These include standard ones like hill-climbing and simulated annealing, and several variants of record-to-record travel and the great deluge criteria (Dueck, 1993).
- Several operator selection schemes in `alns.select`. These include the original (segmented) roulette wheel mechanism of Røpke & Pisinger (2006), and an upper confidence bound bandit algorithm adapted from Hendel (2022).
- Various stopping criteria in `alns.stop` based on maximum run-times or iterations. This includes a criterion that stops after a fixed number of iterations without improvement, which could be used to restart the search.
- Diagnostic statistics collection and plotting methods that can be accessed after solving.

The package can easily be installed through `pip`, and our detailed documentation is available here. To get started using `alns`, a user must provide:

- A solution state for their problem that implements an `objective()` function.
- An initial solution using this solution state.
- One or more destroy and repair operators tailored to the problem.

We provide a quickstart template in our documentation, where these elements are listed as 'TODO'. The documentation also provides several complete implementations of ALNS metaheuristics solving instances of the travelling salesman problem, capacitated vehicle routing problem, cutting stock problem, permutation flow shop problem, and the resource-constrained project scheduling problem. These implementations will help users quickly get started solving their own problems using `alns`.

References

Cicirello, V. A. (2020). Chips-n-Salsa: A Java library of customizable, hybridizable, iterative, parallel, stochastic, and self-adaptive local search algorithms. *Journal of Open Source Software*, *5*(52), 2448. https://doi.org/10.21105/joss.02448

Dreo, J., Liefooghe, A., Verel, S., Schoenauer, M., Merelo, J. J., Quemy, A., Bouvier, B., & Gmys, J. (2021). Paradiseo: From a modular framework for evolutionary computation to the automated design of metaheuristics: 22 years of Paradiseo. *Proceedings of the Genetic*

and Evolutionary Computation Conference Companion, 1522–1530. https://doi.org/10.1145/3449726.3463276

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, *104*(1), 86–92. https://doi.org/10.1006/jcph.1993.1010

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014

Fortin, F.-A., Rainville, F.-M. D., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*(70), 2171–2175.

Hendel, G. (2022). Adaptive large neighborhood search for mixed integer programming. *Mathematical Programming Computation*, *14*(2), 185–221. https://doi.org/10.1007/s12532-021-00209-7

Mejía-de-Dios, J.-A., & Mezura-Montes, E. (2022). Metaheuristics: A Julia package for single- and multi-objective optimization. *Journal of Open Source Software*, *7*(78), 4723. https://doi.org/10.21105/joss.04723

Miranda, L. J. (2018). PySwarms: A research toolkit for particle swarm optimization in Python. *Journal of Open Source Software*, *3*(21), 433. https://doi.org/10.21105/joss.00433

Parejo, J. A., Ruiz-Cortés, A., Lozano, S., & Fernandez, P. (2012). Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Computing*, *16*(3), 527–561. https://doi.org/10.1007/s00500-011-0754-8

Pisinger, D., & Røpke, S. (2010). Large neighborhood search. In M. Gendreau (Ed.), *Handbook of metaheuristics* (2nd ed., pp. 399–420). Springer. https://doi.org/10.1007/978-1-4419-1665-5_13

Reijnen, R., Zhang, Y., Lau, H. C., & Bukhsh, Z. (2022). *Operator selection in adaptive large neighborhood search using deep reinforcement learning*. arXiv. https://doi.org/10.48550/arxiv.2211.00759

Røpke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

Santini, A. (2019). Adaptive large neighbourhood search. In *GitHub repository*. https://github.com/alberto-santini/adaptive-large-neighbourhood-search.

Santini, A., Røpke, S., & Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, *24*(5), 783–815. https://doi.org/10.1007/s10732-018-9377-x

Scott, E. O., & Luke, S. (2019). ECJ at 20: Toward a general metaheuristics toolkit. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 1391–1398. https://doi.org/10.1145/3319619.3326865

Swan, J., Adriaensen, S., Brownlee, A. E. I., Hammond, K., Johnson, C. G., Kheiri, A., Krawiec, F., Merelo, J. J., Minku, L. L., Özcan, E., Pappa, G. L., García-Sánchez, P., Sörensen, K., Voß, S., Wagner, M., & White, D. R. (2022). Metaheuristics "in the large." *European Journal of Operational Research*, *297*(2), 393–406. https://doi.org/10.1016/j.ejor.2021.05.042

Windras Mara, S. T., Norcahyo, R., Jodiawan, P., Lusiantoro, L., & Rifai, A. P. (2022). A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, *146*, 105903. https://doi.org/10.1016/j.cor.2022.105903

Wouda, N. A., Aslan, A., & Vis, I. F. A. (2023). An adaptive large neighbourhood search metaheuristic for hourly learning activity planning in personalised learning. *Computers & Operations Research*, *151*, 106089. https://doi.org/10.1016/j.cor.2022.106089