The Journal of Open Source Software

Chips-n-Salsa: A Java Library of Customizable, Hybridizable, Iterative, Parallel, Stochastic, and Self-Adaptive Local Search Algorithms

Vincent A. Cicirello[1]

1 Computer Science, School of Business, Stockton University, Galloway, NJ 08205

DOI: 10.21105/joss.02448

Software

- Review
- Repository
- Archive

Editor: Gabriela Alessio Robles

Reviewers:

- @nnadeau
- @mbdemoraes
- @mbdemoraes

Submitted: 19 June 2020
Published: 11 August 2020

Summary

Discrete optimization problems are often of practical real-world importance as well as computationally intractable. For example, the traveling salesperson, bin packing, and longest common subsequence problems are NP-Hard, as is resource constrained scheduling, and many single-machine scheduling problems (Garey & Johnson, 1979). Polynomial time algorithms for such problems are unlikely to exist, and the best known algorithms that guarantee optimal solutions have a worst-case exponential runtime. It is thus common to use stochastic local search and other metaheuristics (Gonzalez, 2018). Stochastic local search algorithms begin at a random search state, and apply a sequence of neighbor transitions to nearby search states. This includes perturbative (Hoos & Stützle, 2018) algorithms like simulated annealing (Delahaye, Chaimatanan, & Mongeau, 2019) and hill climbers (Hoos & Stützle, 2018), where each search state is a complete candidate feasible solution, and a mutation operator makes a small random modification to move to another local candidate solution; and also includes constructive (Hoos & Stützle, 2018) algorithms like stochastic samplers (Bresina, 1996; Cicirello & Smith, 2005; Grasas, Juan, Faulin, de Armas, & Ramalhinho, 2017; Langley, 1992; Reyes-Rubiano, Calvet, Juan, Faulin, & Bové, 2020), where each search state is a partial solution that is iteratively transformed into a complete solution. Stochastic local search algorithms do not guarantee optimal solutions. However, they often find near-optimal solutions in much less time than systematic search. They also offer an anytime property (Jesus, Liefooghe, Derbel, & Paquete, 2020; Zilberstein, 1996), where solution quality improves with runtime.

Chips-n-Salsa is a Java library of customizable, hybridizable, iterative, parallel, stochastic, and self-adaptive local search algorithms. Its focus is discrete optimization, but also supports continuous optimization. The library provides a variety of solution representations, including BitVector and IntegerVector classes, indexable vectors of bits and integers, respectively, along with corresponding mutation operators. The library utilizes our significant prior research on permutation optimization problems, providing an extensive set of mutation operators for permutations (Cicirello, 2016; Cicirello & Cernera, 2013), including window-limited mutation (Cicirello, 2014). It uses the JavaPermutationTools (JPT) (Cicirello, 2018a) library for efficiently representing solutions to such problems. For continuous problems, Chips-n-Salsa provides a RealVector class, Gaussian mutation (Petrowski & Ben-Hamida, 2017), Cauchy mutation (Petrowski & Ben-Hamida, 2017), and uniform mutation. The library includes optimization problems useful for benchmarking metaheuristic implementations, such as the well-known OneMax problem (Doerr & Neumann, 2019), BoundMax (generalization of OneMax to integers), the Permutation in a Haystack problem (Cicirello, 2016), and polynomial root finding.

The repository (https://github.com/cicirello/Chips-n-Salsa) contains the library source code, and programs with examples of key functionality. API and other documentation is hosted on the web (https://chips-n-salsa.cicirello.org/). The library can be integrated into projects using maven via GitHub Packages.

Chips-n-Salsa Features and Functionality:

- **Stochastic Local Search Algorithms**: The library supports simulated annealing (Delahaye et al., 2019), with all of the common annealing schedules, as well as advanced annealing schedules. It includes common forms of hill climbing (Hoos & Stützle, 2018), and several stochastic sampling (Grasas et al., 2017) algorithms, such as iterative sampling (Langley, 1992), Heuristic Biased Stochastic Sampling (HBSS) (Bresina, 1996), Value Biased Stochastic Sampling (VBSS) (Cicirello, 2003; Cicirello & Smith, 2005), and stochastic sampling with acceptance bands (Gomes, Selman, & Kautz, 1998; Oddi & Smith, 1997). We optimized random number generation based on our prior research to minimize the runtime impact of one of the more costly operations of a metaheuristic (Cicirello, 2018b).
- **Customizable**: Chips-n-Salsa uses generic types enabling using the library to optimize other representations beyond what is provided in the library. It also enables easily integrating new mutation operators, and customizing all stages of the local search (e.g., choice of annealing schedule for simulated annealing).
- **Hybridizable**: Chips-n-Salsa supports integrating multiple forms of local search (e.g., hybrids of hill climbing with simulated annealing), creating hybrid mutation operators (e.g., combining multiple mutation operators), and running more than one type of search for the same problem concurrently using multiple threads as a form of algorithm portfolio (Gomes & Selman, 2001; Tong, Liu, & Yao, 2019).
- **Iterative**: Chips-n-Salsa supports multistart metaheuristics, including several restart schedules (Cicirello, 2017; Luby, Sinclair, & Zuckerman, 1993) for varying the run lengths across the restarts.
- **Parallel**: Chips-n-Salsa enables parallel execution of multiple instances of the same, or different, stochastic local search algorithms for a problem instance to accelerate search by exploiting multicore architectures (Cicirello, 2017).
- **Self-Adaptive**: Chips-n-Salsa includes adaptive annealing schedules (Hubin, 2019; Štefankovič, Vempala, & Vigoda, 2009) for simulated annealing, such as Modified Lam (Boyan, 1998; Cicirello, 2007), self-tuning variations of the simpler annealing schedules, and adaptive restart schedules (Cicirello, 2017).

Statement of Need

The target audience of Chips-n-Salsa includes those conducting computational research in diverse domains, wherever applications of NP-Hard optimization problems is important. Existing local search open source implementations are often intimately tied to a specific problem, and usually a specific form of local search. For example, a GitHub search finds countless solvers for problems like Boolean Satisfiability or the Traveling Salesperson. Such problem-dependent implementations are not easily adapted to new problems. There are notable exceptions. In particular, ECJ (Scott & Luke, 2019) and Jenetics (Jenetics, 2020) are both mature and well-maintained Java libraries for genetic algorithms and related forms of evolutionary computation. They are problem-independent, support multiple representations, and include multithreaded capabilities. Their focus, however, is on population-based evolutionary search; whereas Chips-n-Salsa is focused on single-solution algorithms. Another especially noteworthy project is emili (Pagnozzi & Stützle, 2019), which is a C++ framework supporting hybrid, single-solution, stochastic local search algorithms. Chips-n-Salsa is problem-independent, representation-independent, and supports hybrid local search, as well as parallel execution to easily exploit multicore architectures to speed up problem solving. Furthermore, the self-adaptive and self-tuning features streamline development, eliminating the need for the developer to tune control parameters.

References

Boyan, J. A. (1998). *Learning evaluation functions for global optimization* (PhD thesis). Carnegie Mellon University, USA.

Bresina, J. L. (1996). Heuristic-biased stochastic sampling. In *Proceedings of the thirteenth national conference on artificial intelligence - volume 1* (pp. 271–278). AAAI Press.

Cicirello, V. A. (2003). *Boosting stochastic problem solvers through online self-analysis of performance* (PhD thesis). The Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Cicirello, V. A. (2007). On the design of an adaptive simulated annealing algorithm. In *Proceedings of the international conference on principles and practice of constraint programming first workshop on autonomous search*. AAAI Press.

Cicirello, V. A. (2014). On the effects of window-limits on the distance profiles of permutation neighborhood operators. In *Proceedings of the 8th international conference on bio-inspired information and communications technologies* (pp. 28–35). doi:10.4108/icst.bict.2014.257872

Cicirello, V. A. (2016). The permutation in a haystack problem and the calculus of search landscapes. *IEEE Transactions on Evolutionary Computation*, *20*(3), 434–446. doi:10.1109/TEVC.2015.2477284

Cicirello, V. A. (2017). Variable annealing length and parallelism in simulated annealing. In *Proceedings of the tenth international symposium on combinatorial search (socs 2017)* (pp. 2–10). AAAI Press.

Cicirello, V. A. (2018a). JavaPermutationTools: A java library of permutation distance metrics. *Journal of Open Source Software*, *3*(31), 950. doi:10.21105/joss.00950

Cicirello, V. A. (2018b). Impact of random number generation on parallel genetic algorithms. In *Proceedings of the thirty-first international florida artificial intelligence research society conference* (pp. 2–7). AAAI Press.

Cicirello, V. A., & Cernera, R. (2013). Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms. In *Proceedings of the twenty-sixth international florida artificial intelligence research society conference* (pp. 46–51). AAAI Press.

Cicirello, V. A., & Smith, S. F. (2005). Enhancing stochastic search performance by value-biased randomization of heuristics. *Journal of Heuristics*, *11*(1), 5–34. doi:10.1007/s10732-005-6997-8

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated annealing: From basics to applications. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of metaheuristics* (pp. 1–35). Springer. doi:10.1007/978-3-319-91086-4_1

Doerr, B., & Neumann, F. (2019). *Theory of evolutionary computation: Recent developments in discrete optimization*. Natural computing series. Springer. ISBN: 9783030294144

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness*. W. H. Freeman. ISBN: 0716710455

Gomes, C. P., & Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, *126*(1), 43–62. doi:10.1016/S0004-3702(00)00081-3

Gomes, C. P., Selman, B., & Kautz, H. (1998). Boosting combinatorial search through randomization. In *Proceedings of the fifteenth national conference on artificial intelligence* (pp. 431–437). AAAI Press.

Gonzalez, T. F. (Ed.). (2018). *Handbook of approximation algorithms and metaheuristics methologies and traditional applications* (2nd ed., Vol. 1). Chapman; Hall/CRC. doi:10.1201/9781351236423

Grasas, A., Juan, A. A., Faulin, J., de Armas, J., & Ramalhinho, H. (2017). Biased randomization of heuristics using skewed probability distributions: A survey and some applications. *Computers & Industrial Engineering*, *110*, 216–228. doi:10.1016/j.cie.2017.06.019

Hoos, H. H., & Stützle, T. (2018). Stochastic local search. In *Handbook of approximation algorithms and metaheuristics methologies and traditional applications* (2nd ed., Vol. 1). Chapman; Hall/CRC.

Hubin, A. (2019). An adaptive simulated annealing em algorithm for inference on non-homogeneous hidden markov models. In *Proceedings of the international conference on artificial intelligence, information processing and cloud computing* (pp. 1–9). ACM Press. doi:10.1145/3371425.3371641

Jesus, A. D., Liefooghe, A., Derbel, B., & Paquete, L. (2020). Algorithm selection of anytime algorithms. In *Proceedings of the 2020 genetic and evolutionary computation conference* (pp. 850–858). ACM Press. doi:10.1145/3377930.3390185

Langley, P. (1992). Systematic and nonsystematic search strategies. In *Proceedings of the first international conference on artificial intelligence planning systems* (pp. 145–152). Morgan Kaufmann. doi:10.1016/b978-0-08-049944-4.50022-7

Luby, M., Sinclair, A., & Zuckerman, D. (1993). Optimal speedup of las vegas algorithms. *Information Processing Letters*, *47*(4), 173–180. doi:10.1016/0020-0190(93)90029-9

Oddi, A., & Smith, S. F. (1997). Stochastic procedures for generating feasible schedules. In *Proceedings of the fourteenth national conference on artificial intelligence* (pp. 308–314). AAAI Press.

Pagnozzi, F., & Stützle, T. (2019). Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. *European Journal of Operational Research*, *276*(2), 409–421. doi:10.1016/j.ejor.2019.01.018

Petrowski, A., & Ben-Hamida, S. (2017). *Evolutionary algorithms*. Computer engineering: Metaheuristics. Wiley. doi:10.1002/9781119136378

Reyes-Rubiano, L. S., Calvet, L., Juan, A. A., Faulin, J., & Bové, L. (2020). A biased-randomized variable neighborhood search for sustainable multi-depot vehicle routing problems. *Journal of Heuristics*, *26*(3), 401–422. doi:10.1007/s10732-018-9366-0

Scott, E. O., & Luke, S. (2019). ECJ at 20: Toward a general metaheuristics toolkit. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 1391–1398). ACM Press. doi:10.1145/3319619.3326865

Štefankovič, D., Vempala, S., & Vigoda, E. (2009). Adaptive simulated annealing: A near-optimal connection between sampling and counting. *Journal of the ACM*, *56*(3), 18:1–18:36. doi:10.1145/1516512.1516520

Tong, H., Liu, J., & Yao, X. (2019). Algorithm portfolio for individual-based surrogate-assisted evolutionary algorithms. In *Proceedings of the genetic and evolutionary computation conference* (pp. 943–950). ACM Press. doi:10.1145/3321707.3321715

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, *17*(3), 73–83. doi:10.1609/aimag.v17i3.1232

Jenetics. (2020). Jenetics - genetic algorithm, genetic programming, evolutionary algorithm, and multi-objective optimization. *GitHub repository*. GitHub. Retrieved from https://jenetics.io/