Introduction to automated design of scheduling heuristics with genetic programming

Marko Đurasević[1], Domagoj Jakobović[1], Yi Mei [2], Su Nguyen[3], and Mengjie Zhang [2]

[1] University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
[2] School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand
[3] Centre for Data Analytics and Cognition (CDAC), La Trobe University, Melbourne, Australia

marko.durasevic@fer.hr domagoj.jakobovic@fer.hr yi.mei@ecs.vuw.ac.nz nguyenphanbachsu@gmail.com mengjie@ecs.vuw.ac.nz

https://gecco-2022.sigevo.org/

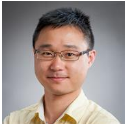
Instructors

Marko Đurasević received his PhD degree from the Faculty of Electrical Engineering and Computing, University of Zagreb in February 2018 on the subject of generating dispatching rules for the unrelated machines environment. He is currently employed as an Assistant Professor at the Department of Electronics, Microelectronics, Intelligent and Computer and Intelligent Systems of the Faculty of Electrical Engineering and Computing. His research interests include the field of evolutionary computing, optimization methods, machine learning, and scheduling problems. He has published nineteen journal and conference papers.

Domagoj Jakobović received his PhD degree in 2005 at the Faculty of Electrical Engineering and Computing, University of Zagreb, on the subject of generating scheduling heuristics with genetic programming. He is currently full professor at the Department of Electronics, Microlelectronics, Computer and Intelligent Systems. His research interests include evolutionary algorithms, optimization methods and parallel algorithms. Most notable contributions are in the area of machine supported scheduling, optimization problems in cryptography, parallelization and improvement of evolutionary algorithms. He has published more than 100 papers, lead several research projects and served as a reviewer for many international journals and conferences. He has supervised seven doctoral theses and more than 170 bachelor and master theses.

Instructors

Yi Mei is a Senior Lecturer at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. He received his BSc and PhD degrees from University of Science and Technology of China in 2005 and 2010, respectively. His research interests include evolutionary computation and learning in scheduling and combinatorial optimisation, hyper-heuristics, genetic programming, automatic algorithm design. Yi has more than 150 fully refereed publications, including the top journals in EC and Operations Research (OR) such as IEEE TEVC, IEEE Transactions on Cybernetics, European Journal of Operational Research, ACM Transactions on Mathematical Software, and top EC conferences (GECCO). He serves as a reviewer of over 50 international journals including the top journals in EC and OR.



Su Nguyen is a Senior Research Fellow and Algorithm Lead at the Centre for Data Analytics and Cognition (CDAC), La Trobe University, Australia. He received his Ph.D. degree in Artificial Intelligence and Data Analytics from Victoria University of Wellington (VUW), Wellington, New Zealand, in 2013. His expertise includes computational intelligence, optimization, data analytics, large-scale simulation, and their applications in energy, operations management, and social networks. His current research focuses on novel people-centric artificial intelligence to enhance explainability and human-AI interaction by combining the power of evolutionary computation techniques and advanced machine learning algorithms. His works have been published in top peer-reviewed journals in evolutionary computation and operations research.

Mengjie Zhang is a Fellow of Royal Society of New Zealand, a Fellow of IEEE, and currently Professor of Computer Science at Victoria University of Wellington, where he heads the interdisciplinary Evolutionary Computation Research Group. He is a member of the University Academic Board, a member of the University Postgraduate Scholarships Committee, Associate Dean in the Faculty of Engineering, and Chair of the Research Committee of the Faculty of Engineering and School of Engineering and Computer Science. His research is mainly focused on evolutionary computation, particularly genetic programming, particle swarm optimisation and learning classifier systems with application areas of feature selection/construction and dimensionality reduction, computer vision and image processing, evolutionary deep learning and transfer learning, job shop scheduling, multi-objective optimisation, and clustering and classification with unbalanced and missing data. He is also interested in data mining, machine learning, and web information extraction. Prof Zhang has published over 700 research papers in refereed international journals and conferences in these areas. He has been serving as an associated editor or editorial board member for over 10 international journals including IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics, the Evolutionary Computation Journal, ACM Transactions on Evolutionary Learning and Optimisation, Genetic Programming and Evolvable Machines, IEEE Transactions on Emergent Topics in Computational Intelligence, Applied Soft Computing, and Engineering Applications of Artificial Intelligence, and as a reviewer of over 30 international journals. He has been a major chair for eight international conferences.

Outline

- Introduction to scheduling
- Solution methods
  - Dispatching rules
- Automated design of dispatching rules
  - Representation
  - Terminal nodes
- Advanced topics
  - Improving performance
  - Ensemble learning
  - Multi-objcevice optimistaion
  - Interpretability
  - ...
- Conclusions
- Resources

Introduction to scheduling

- Allocation of certain activities (jobs) to a limited set of resources (machines) [42]
- Goal: optimise one or more user defined criteria
- NP-hard in most scenarios
- Different applications:
  - Manufacturing [4]
  - Cloud [48]
  - Workforce [7]

Problem variants

- Single machine - all jobs need to be scheduled on a single machine
- **Parallel machines** - each jobs needs to be scheduled on one of the available machines
- Flow shop - each job needs to visit all machines and all jobs have the same route
- Job shop - each job needs to visit all machines but each job has its own route

Parallel machines environment

- n jobs need to be scheduled on one of the m available machines
- job properties:
  - processing time p_{ij} - how long does machine i process job j
  - weight w_j - how important job j is
  - release time r_j - when job j becomes available
  - due date d_j - until when job j should be completed
  - ...

Additional constraints

- *Setup times* - time required to adapt a machine for a job
- *Precedence constraints* - some jobs can be scheduled only after others finished executing
- *Machine unavailability* - machines are unavailable in some periods (breakdowns, or maintenance)
- *Machine eligibility* - jobs can only execute on some machines
- *Batch scheduling* - machines can process several jobs in parallel
- *Auxiliary resources* - additional resources are required for processing jobs (workers or material)
- ...

Optimisation criteria

- Makespan - the completion time of the last job
- Total (weighted) flowtime - the total time that jobs spend in the system
- Total (weighted) tardiness - the amount of time that jobs spend executing after their due date
- Maximum flowtime
- Maximum tardiness
- Arbitrary user defined criteria!
- Multi-objective scheduling

Scheduling conditions

- Parameter reliability:
  - *Deterministic* - all system parameters are known exactly
  - *Stochastic* - parameter values are not known exactly, they can only be approximated
- Parameter availability:
  - *Offline* - all system parameters are available before the execution of the system
  - *Online* - certain system parameters become available during the execution of the system (e.g. with the arrival of new jobs)
- Schedule construction:
  - *Static* - the schedule is constructed before the system begins executing (applicable with offline scheduling)
  - *Dynamic* - the schedule is constructed in parallel with the execution of the system

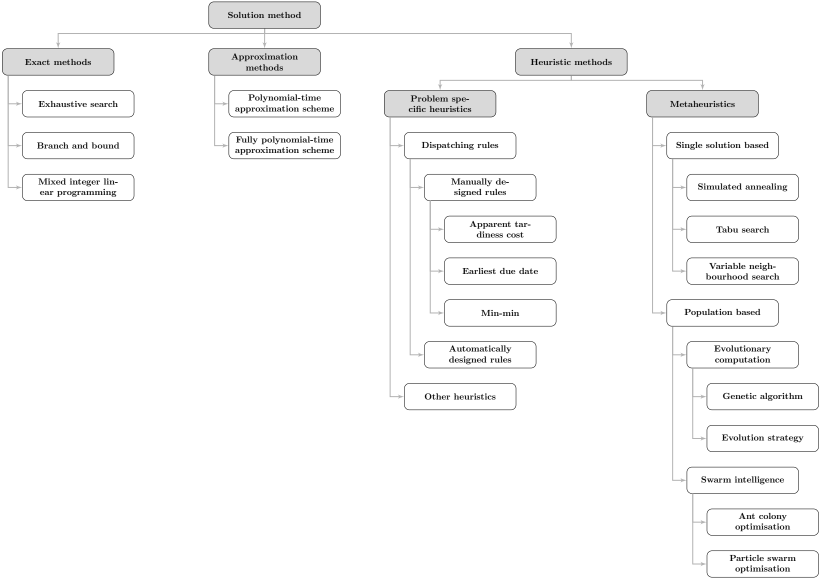
Solving Scheduling Problems

1. Exact algorithms [8]
   - Can obtain optimal solutions
   - High computational cost $\rightarrow$ can be used only for smaller problems
2. Approximate algorithms [23]
   - Obtain a solution within a given bound from the optimal solution
   - Difficult to design and applicable only to static problems
3. Heuristic methods
   - Provide no guarantee that they will achieve optimal results
   - Fast and flexible
   - Two variants
     - Improvement heuristics [13]
     - Constructive heuristics - dispatching rules [53]

Improvement heuristics

- Start with a complete schedule (usually created randomly or by some simple heuristic)
- Iteratively improve it using various operators
- Since they search the solution space, usually only applicable for static scheduling problems
- Various metaheuristics are most commonly used [13]:
  - Genetic algorithms [60, 61]
  - Simulated annealing [24]
  - Tabu search [22]
  - Iterated local search [49]
  - ...

Solving Scheduling Problems

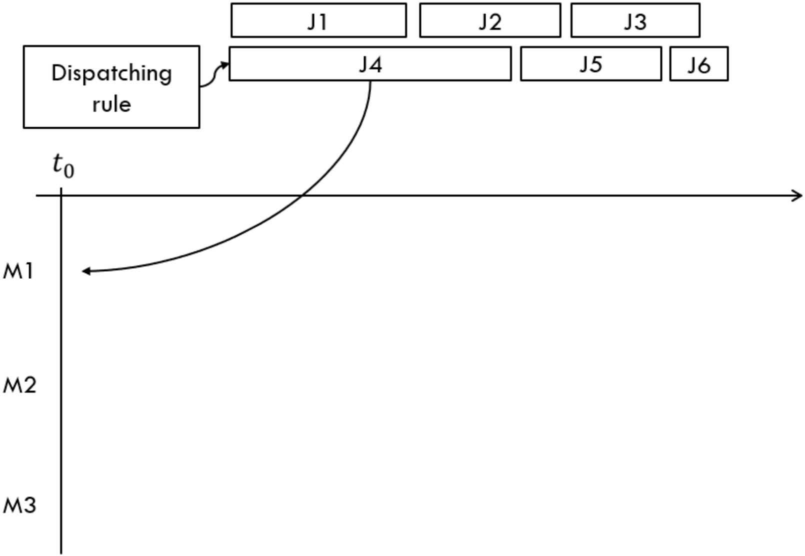


Dispatching rules

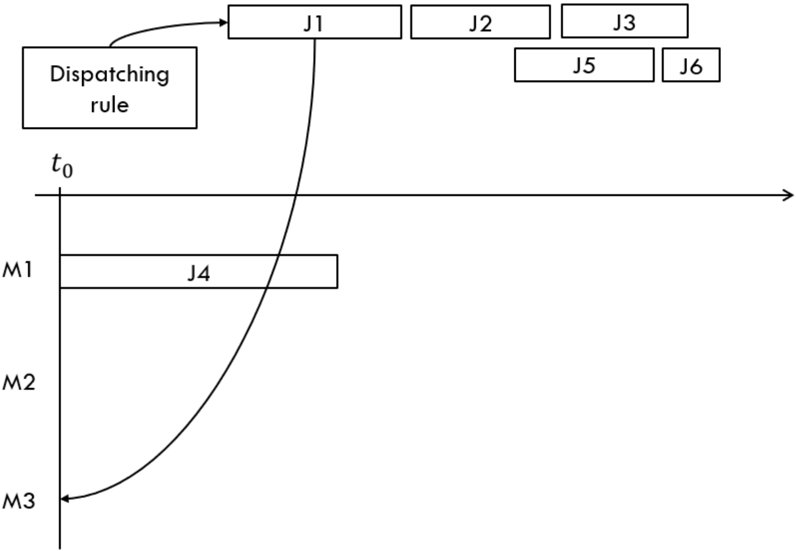
Dispatching rules (DRs)

- Build the schedule iteratively
- At each decision point (when a machine and job are available) determine which job should be scheduled
- Only the information available at the decision point is used (only released jobs)
- Can quickly react to changes in the schedule (arrival of jobs, breakdown of a machine, etc.)
- A plethora of DRs have been proposed for various scheduling problems and criteria [53]
- For example: *earliest due date* (EDD) - schedule the job which has the earliest due date

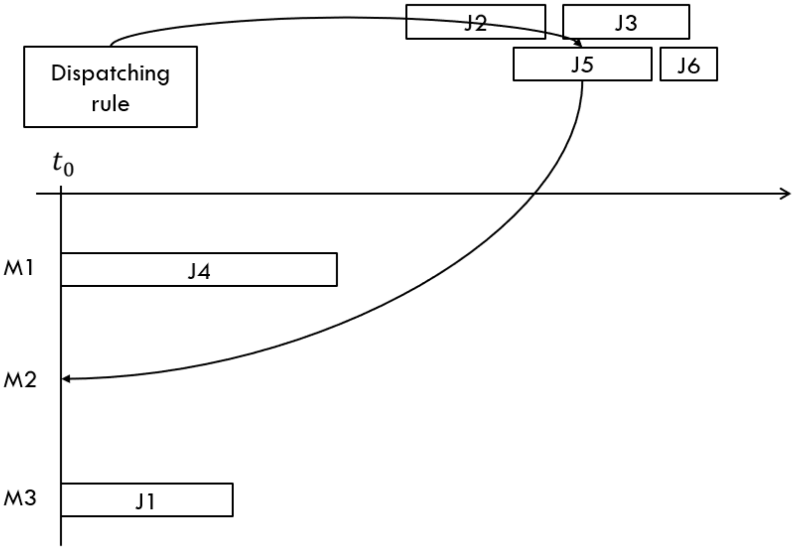
Dispatching rule execution



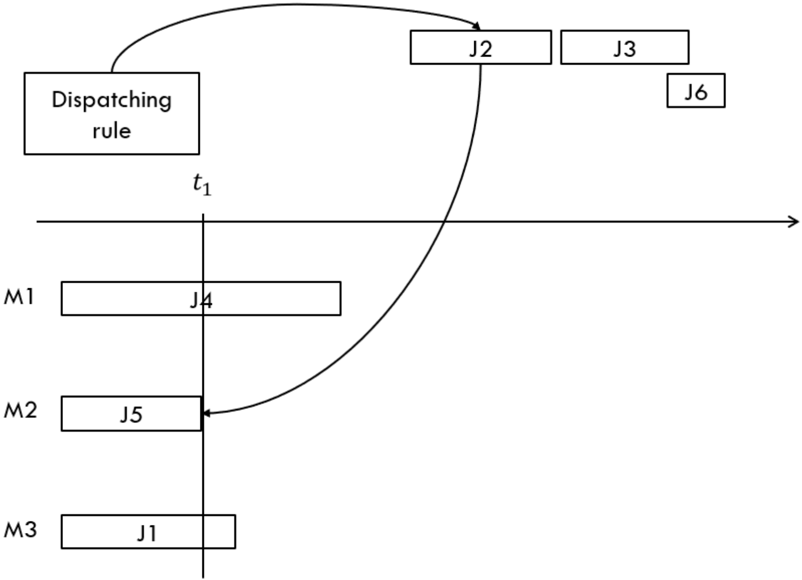
Dispatching rule execution



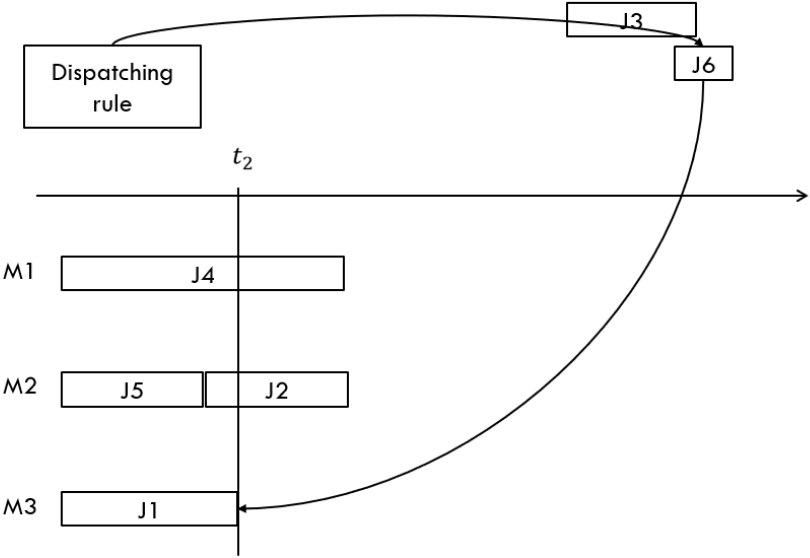
Dispatching rule execution



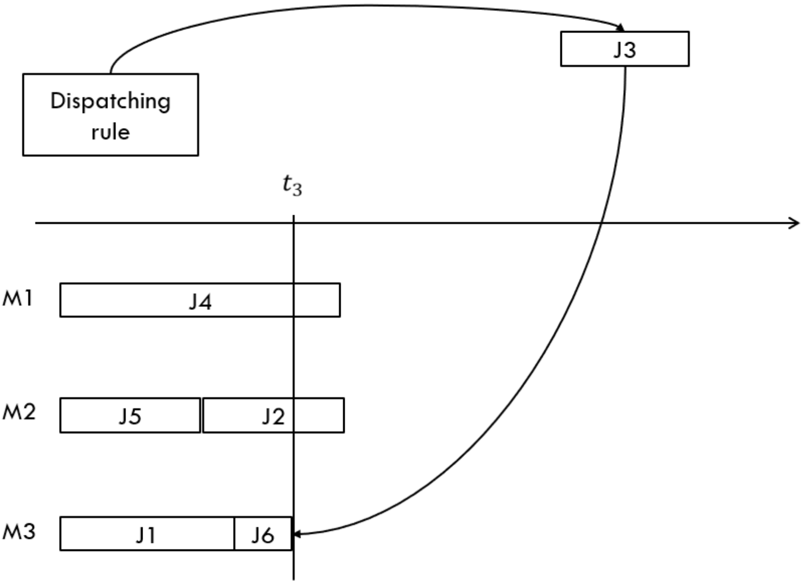
Dispatching rule execution



Dispatching rule execution



Dispatching rule execution



Dispatching rules (DRs)

- Consist of two parts: schedule generation scheme (SGS) and priority function (PF)
  - Schedule generation scheme (SGS) - constructs the schedule (determines when to schedule jobs)

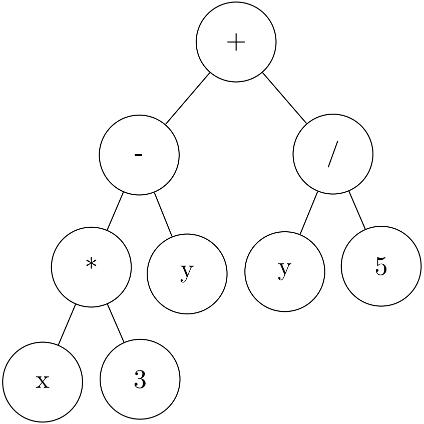
```
while unscheduled jobs are available do
    Wait until at least one job and one machine are available
    Calculate priority π_ij for scheduling job j on machine i
    Schedule the job with best priority
end while
```

  - Priority function (PF) - assigns priorities to jobs; e.g. WSPT:

$$\pi_{ij} = \frac{w_j}{p_{ij}}$$

Automated design of dispatching rules

- Metaheuristic optimisation method similar to genetic algorithms [47]
- Individuals represented in the form of expression trees:
  - Inner nodes - functions (arithmetic, Boolean, etc.)
  - Leaf nodes - terminals (variables and constants)



Automatic design of dispatching rules

- The SGS is usually defined manually [56]
- GP is used to evolve a new PF
- Problem specific terminals need to be provided:
  - processing time
  - due date
  - remaining time to tardiness
  - time until the most suitable machine is available
- A customised feature construction may be utilised to evolve better rules [16]
  - expert knowledge may be beneficial! (but difficult to obtain)

Dispatching rule execution

Priority rule:

$$\pi_j = \frac{p_j * (d_j - time)}{w_j}$$

Job 1:

- $p = 10$
- $d = 17$
- $w = 0.8$

$\pi_1 = 212.5$

Job 2:

- $p = 7$
- $d = 30$
- $w = 0.5$

$\pi_2 = 420$

Schedule:

Machine 1

Time = 0

Dispatching rule execution

Priority rule:

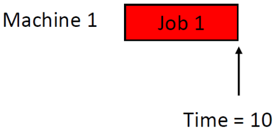
$$\pi_j = \frac{p_j * (d_j - time)}{w_j}$$

Job 2:

- $p = 7$
- $d = 30$
- $w = 0.5$

$\pi_2 = 280$

Schedule:



Job 3:

- $p = 13$
- $d = 25$
- $w = 0.7$

$\pi_3 = 278.6$

Dispatching rule execution

Priority rule:

$$\pi_j = \frac{p_j * (d_j - time)}{w_j}$$

Job 2:

- $p = 7$
- $d = 30$
- $w = 0.5$

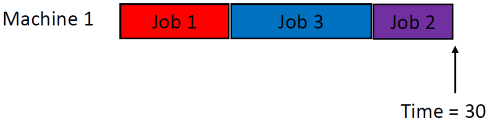
$\pi_2 = 98$

Schedule:

Priority rule:

$$\pi_j = \frac{p_j * (d_j - time)}{w_j}$$

Schedule:



How to evolve and evaluate rules?

- Machine learning: using at least two data sets
- Training set:
  - Used during evolution to train dispatching rules
  - Needs to be general enough (!)
  - Different ways of using it:
    - using same instances all the time [57]
    - Cycle through the instances [68]
- Test set:
  - Used to test the evolved DRs
  - Unseen instances, must not have been used during training
  - Must be to a certain degree similar to training instances, otherwise the rule will not perform well
- Potential problem: overfitting

Representations and terminals

Which representation to use?

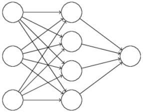
- GP vs. artificial neural networks [2]
  - Similar performance, but neural networks are not interpretable
- Different GP representations: tree, gene expression programming, Cartesian genetic programming, etc. [44]
  - Usually achieve similar performance
  - Some representations are less inclined towards evolving large expressions - better interpretability
- What should GP evolve? [31]
  - A function for selecting existing DRs (selective hyper-heuristic)
  - A new DR (constructive hyper-heuristic)
  - A combination of both

Dispatching rule representations

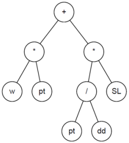
Linear representation

$$\sum_i w_i s_i$$

Neural network



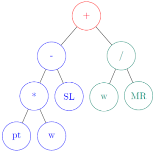
Genetic programming



Gene expression programming

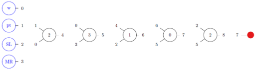
- * pt w SL | / w MR w dd

Gene 1   Gene 2



Cartesian genetic programming

2 1 0 3 0 3 1 4 2 0 6 5 2 2 5 7



Cartesian genetic programming

0 0 4 1 0 6 1 2 2 1 0 0 1 3 3 0 7 1 2 3 1 5 8 3 1

<expr> : <expr> <op> <expr> | <subexpr> | (<expr>)
<subexpr> : <func> (<expr>) | <var>
<func> : pos
<op> : + | - | * | /
<var> : pt | dd | w | SL | pmin | pavg | PAT | MR | age

- many others could be used! (AP, stack GP, LGP, ...)

Terminal node examples

- Simple terminals - represent some system properties of jobs or machines
  - processing time of job
  - due date of job
  - weight of job
  - ...
- Complex terminals - represent combinations of simple job properties
  - slack - time left until the job becomes tardy
  - time that the job spent in the system
  - time when the machine which can process the job the fastest becomes available
  - ...

How to select the right terminal nodes?

- More terminals → larger search space
- Simple terminals → large expression, GP wastes time to obtain good subexpressions
- Using a too restricted terminal set can lead to "myopic" rules [16] - DRs consider only a single scheduling decision
- Solutions:
  - Manually construct and select features - slow and time consuming
  - Introduce feature selection in the search process
    - Select features by importance based on adapted preliminary runs [26]
    - Constructing new features during the evolution [62]
    - Two stage approach where the first stage evolves rules to determine useful features and the second stage builds on those results [69, 67]

Improving performance

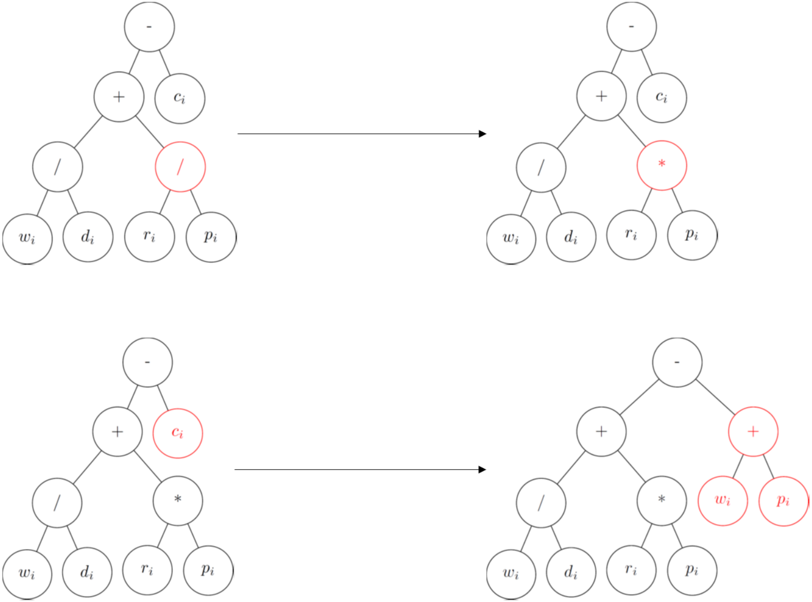
How to improve performance?

- Improve the evolutionary process
  - Local search
  - Improve genetic programming elements
  - Surrogate models
- Improve performance of generated rules
  - Ensemble learning methods
  - Adaptation to static conditions
  - Using the appropriate DR for a given problem instance

Local search

- Apply local search procedures to search the neighbourhood of good expressions
- How to define a neighbour of an expression tree?
- Customized neighbourhood structures and LS operators are proposed for this purpose - depend on the representation!
- Promising initial results [29, 12]
- Still open for further research

Local search operators



Improving genetic programming elements

- Application of ϵ-lexicase selection [46] - possible to apply to any selection scheme
- Adaptive recombination operators [66] - use of a *decision vector* to characterize a (sub)tree
- Calculation of correlation of subtrees within a tree to select crossover points
- Subtree selection mechanisms in genetic operators [64]
- Hyper-heuristic parameter configuration using fitness landscape analysis - concentrates on genotype space [59]

Surrogate models

- Evaluation in GP is usually slow, especially for difficult combinatorial problems
- Many instances have to be used in training to ensure the generalisation capability of the evolved rules
- Solution: surrogate models [14, 37, 65]
- Use simple scheduling problems to estimate the quality of generated DRs without having to evaluate them on the entire training set
- Use a subset of instances, possible with instance rotation
- Advantages:
  - Better convergence
  - Simpler rules
  - Some improvements in execution time

Multitask genetic programming

- Novel application for hyper-heuristics [63]
- The (scheduling) problem is divided in several variants: *tasks* (e.g. different utilization levels)
- Individuals are divided into subpopulations and evolved for separate tasks
- Transfer knowledge between the subpopulations during the evolution process
- Represents a fruitful new research direction; requires a sensible division in tasks

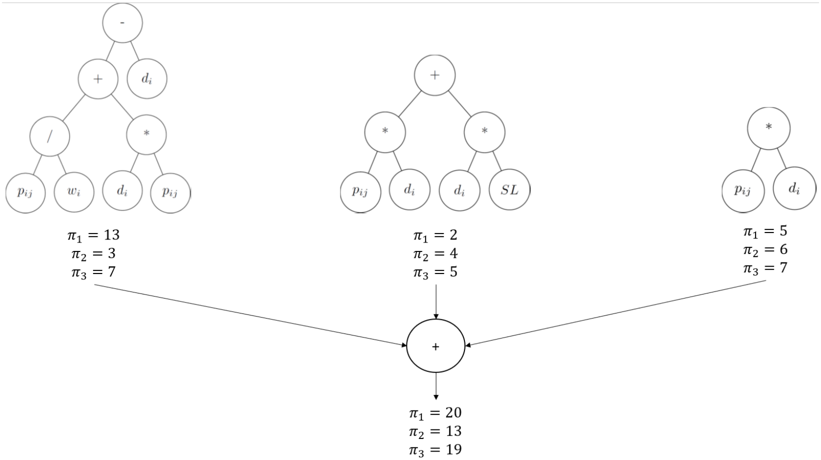
Ensembles of DRs

- A single heuristic will not work well across all the different problems
- Why not use several DRs in synergy?
- Idea from machine learning
- Collect DRs into ensembles and use them in synergy to perform scheduling decisions

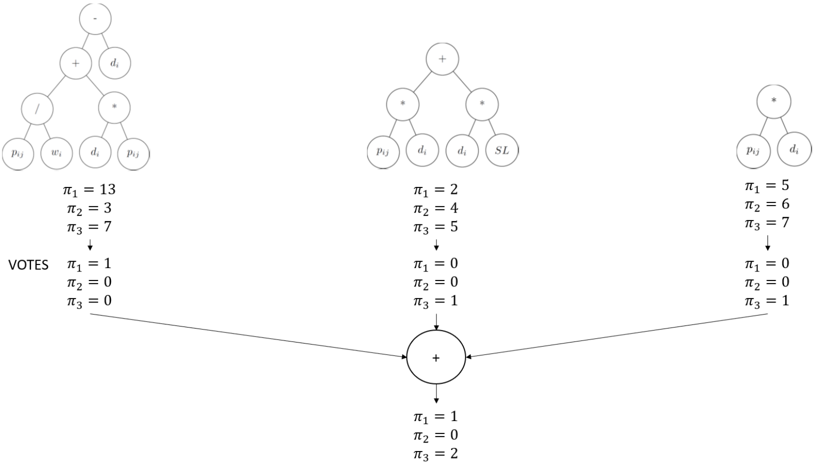
Ensembles of DRs

- Different methods can be used to construct ensembles:
  - Cooperative coevolution [41]
  - BagGP [52, 50]
  - BoostGP [52, 50]
  - Simple ensemble combination - relies on previously evolved rules: simple and efficient [52, 54, 50]
  - Genetic algorithms - GA optimizes rule selection in ensemble [11, 10]
- Different ways of aggregating their decisions:
  - sum, vote, weighted vote, weighted sum combination methods [40]
  - each rule in the ensemble creates the schedule and then the best solution is selected [11, 10]

Sum ensemble combination



Vote ensemble combination



Tackling static problems

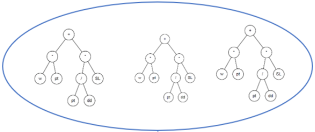
- DRs can also be applied in static conditions when all information is available beforehand
- Idea: DRs should use all the information about the problem
- Approaches:
  - Look-ahead – calculate priorities for unreleased jobs [15]
  - Iterative DRs – rebuild the schedule several times [33]
  - Rollout – at each decision point determine the best option via a DR [55]
- Some methods can even match those of improvement heuristics!

- GP evolves a lot of DRs
- Open question: which should we select (i.e. which performs best for the given problem kind or problem instance)?
- Impossible to know beforehand in dynamic problems
- Idea: based on properties of the problem that become known during execution, try to select the most appropriate DR using some classification algorithms [71]
- Good initial results, but the methods need a lot of fine tuning

Problem instances for learning

Problem instance	p_0	r_0	dd_0	…	p_n	r_n	dd_n
0	15	15	150	…	12	35	45
1	98	104	170	…	78	64	5
2	1	43	79	…	66	89	9
3	25	76	151	…	55	31	24
4	47	96	137	…	85	75	47
5	31	70	255	…	13	82	19
6	59	84	173	…	22	93	92
7	42	78	130	…	78	24	62
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k	13	24	60	…	88	35	178

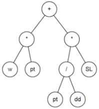
Automatically generated DRs



Learn

Classifier (k-nn, ANN, Bayes, C4.5, etc)

Appropriate DR



New problem instance

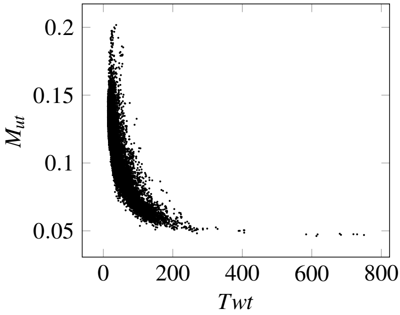
p_0	r_0	dd_0	…	p_m	r_m	dd_m
54	78	104	…	55	41	103

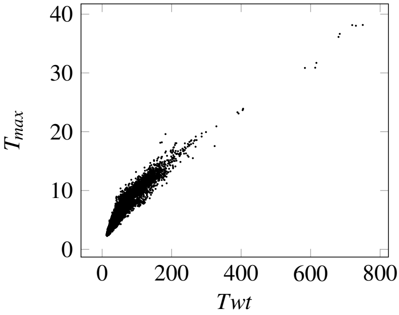
Other topics

What if we want to optimise multiple objectives?

- Usually several criteria need to optimised in real world problems
- Manually designed DRs are mostly adapted for optimising only a single criterion
- Criteria are usually conflicting - impossible to design a rule which optimises all criteria well
- Various multi-objective genetic algorithms can be used to design DRs for optimising several criteria: NSGA-II, NSGA-III, MOEA/D, etc. [32, 36, 58]
- The automatically generated DRs show a much better performance than manually designed rules for various multi-objective problems

What if we want to optimise multiple objectives?





What if we do not know all parameters exactly?

- In many problems parameters are stochastic
- We do not know exact values of parameters until they are executed
  - For example, we do not know the exact processing time until the job finishes processing
- Processing times mostly considered as uncertain [21]
- Uncertain parameters are modelled with stochastic variables
- Uncertainty about the parameters is included in genetic programming with additional terminals [20]

But what about more complex problems?

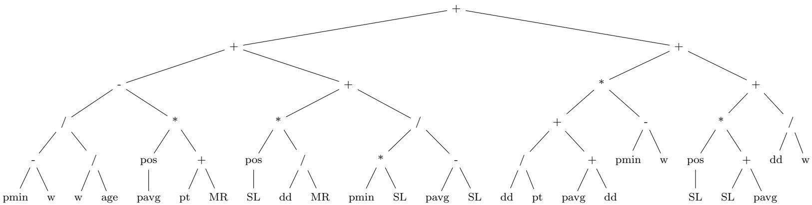
- Many papers consider additional constraints:
  - Setup times [19, 18]
  - Machine breakdowns [38, 39, 18]
  - Precedence constraints [19, 18]
  - Machine eligibility [19]

But what about more complex problems?

- Required to adapt the SGS and PF of the DRs
- SGS adaptation - needs to ensure that only feasible schedules are constructed
  - Schedule only jobs for which all predecessors have executed
  - Schedule jobs only on eligible machines
  - ...
- PF adaptation - provide information about the additional constraints
  - Setup time of job j on machine i
  - Number of predecessors/successors for job j
  - ...

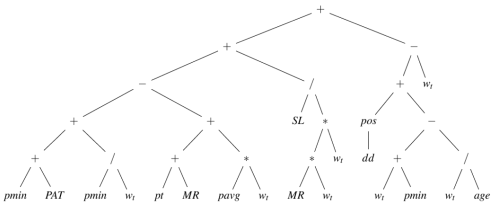
Interpretability

Let's interpret a priority function



- Not really interpretable...
- We can try to manually reduce the complexity

Manual simplification



Arithmetic representation:

$$pmin + PAT + \frac{pmin}{w_t} - (pt + MR + pavg * w_t) + \frac{SL}{MR * w_t^2} + dd + w_t + pmin - \frac{w_t}{age} - w_t$$

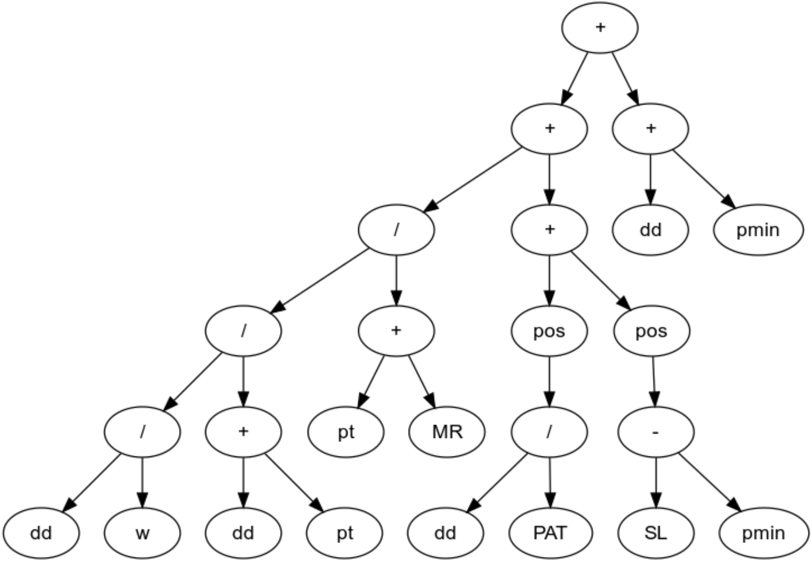
Arithmetic representation (after simplification):

$$PAT + \frac{pmin}{w_T} - pt - MR + \frac{SL}{MR * w_T^2} + dd$$

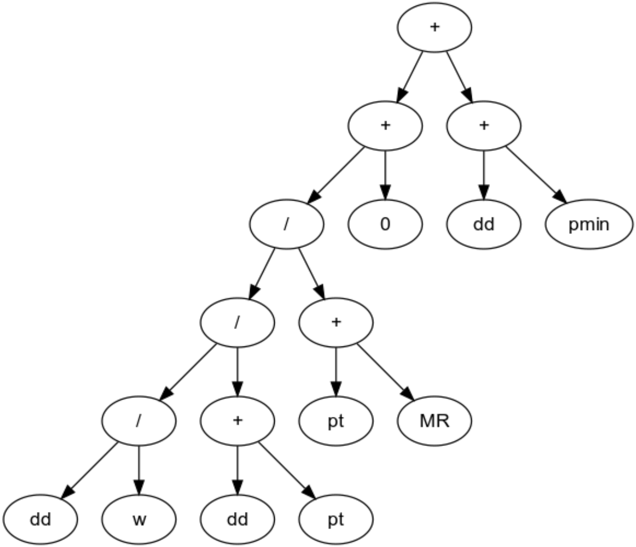
Let's interpret a priority function

- Expressions become bloated, difficult to interpret
- Possible remedies:
  - Exact and heuristic simplification [45]
  - Dimensionally aware GP - evolves expressions which follow rules of dimensionality [57, 27]
  - Multi-objective optimisation - one criterion is the size/complexity of the expression [43]

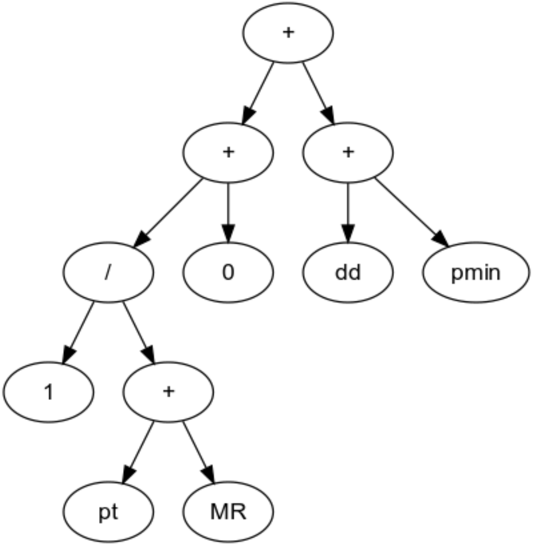
Example of simplification



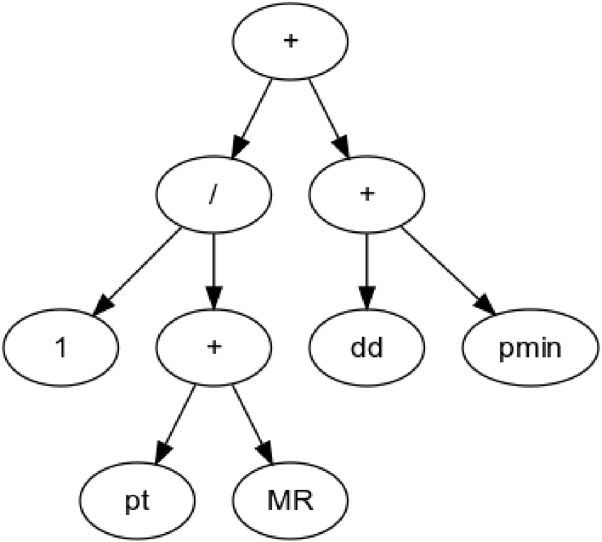
Example of simplification



Example of simplification



Example of simplification



Conclusions and outlook

Other notable scheduling problem variants

- Order acceptance and scheduling - each job can be rejected or accepted for scheduling [28]
- Resource constrained project scheduling problem - scheduling consumes additional limited resources [51, 5]
- One machine scheduling with variable capacity - the capacity of the machine (number of jobs it can process in parallel) varies over time [9, 12]
- Due date assignment rules [35, 34]

Similar applications on other problems

- Vehicle routing problem [17]
- Capacitated arc routing problem [1]
- Travelling salesman problem [6]
- Bin packing problem [25]

Conclusion

- Many existing research directions [3, 30]
- A heavily investigated field
- Many different scheduling problem variants, most not yet investigated
- A lot of room for further improvement

Additional resources

Literature

- Survey papers about automated design of DRs
  - Jurgen Branke, Su Nguyen, Christoph W. Pickardt, and Mengjie Zhang. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, 20(1):110–124, February 2016
  - Su Nguyen, Yi Mei, and Mengjie Zhang. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, 3(1):41–66, February 2017
- Recent book on production scheduling:
  - Fangfang Zhang, Su Nguyen, Yi Mei, and Mengjie Zhang. *Genetic Programming for Production Scheduling*. Springer Singapore, 2021

Upcoming sessions and talks

- IEEE WCCI Special session on scheduling and combinatorial optimisation
  - https://meiyi1986.github.io/cec2022-esco/
- IEEE WCCI Special session on evolutionary machine learning for planning and scheduling
  - https://fangfang-zhang.github.io/CEC2022EMLPS/
- IEEE WCCI tutorial on evolutionary machine learning for combinatorial optimisation
  - https://fangfang-zhang.github.io/CEC2022Tutorial/

Codes and other resources

- IEEE Taskforce on Evolutionary Scheduling and Combinatorial Optimisation
  - https://homepages.ecs.vuw.ac.nz/~yimei/ieee-tf-esco/
- Codes and instances
  - https://github.com/meiyi1986/GPJSS
  - http://gp.zemris.fer.hr/hyddra/

Acknowledgement

- This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-4333

References I

[1] Mazhar Ansari Ardeh, Yi Mei, and Mengjie Zhang. Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2021.

[2] Jürgen Branke, Torsten Hildebrandt, and Bernd Scholz-Reiter. Hyper-heuristic evolution of dispatching rules: A comparison of rule representations. *Evolutionary Computation*, 23(2):249–277, June 2015.

[3] Jurgen Branke, Su Nguyen, Christoph W. Pickardt, and Mengjie Zhang. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, 20(1):110–124, February 2016.

[4] G. Celano, A. Costa, and S. Fichera. Scheduling of unrelated parallel manufacturing cells with limited human resources. *International Journal of Production Research*, 46(2):405–427, 2008.

[5] Shelvin Chand, Quang Huynh, Hemant Singh, Tapabrata Ray, and Markus Wagner. On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences*, 432:146–163, March 2018.

[6] Gabriel Duflo, Emmanuel Kieffer, Matthias R. Brust, Grégoire Danoy, and Pascal Bouvry. A gp hyper-heuristic approach for generating tsp heuristics. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 521–529, 2019.

[7] A.T Ernst, H Jiang, M Krishnamoorthy, and D Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004. Timetabling and Rostering.

[8] Luis Fanjul-Peyro. Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources. *Expert Systems with Applications: X*, 5:100022, 2020.

[9] Francisco J. Gil-Gala, Carlos Mencía, María R. Sierra, and Ramiro Varela. Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time. *Applied Soft Computing*, 85:105782, December 2019.

[10] Francisco J. Gil-Gala, Carlos Mencía, María R. Sierra, and Ramiro Varela. Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. *Integrated Computer-Aided Engineering*, 28(1):65–80, December 2020.

[11] Francisco J. Gil-Gala, María R. Sierra, Carlos Mencía, and Ramiro Varela. Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling. *Natural Computing*, June 2020.

[12] Francisco J. Gil-Gala, María R. Sierra, Carlos Mencía, and Ramiro Varela. Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity. *Swarm and Evolutionary Computation*, 66:100944, October 2021.

[13] Emma Hart, Peter Ross, and David Corne. Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines*, 6(2):191–220, June 2005.

[14] Torsten Hildebrandt and Jürgen Branke. On using surrogates with genetic programming. *Evolutionary Computation*, 23(3):343–367, September 2015.

[15] Torsten Hildebrandt, Jens Heger, and Bernd Scholz-Reiter. Towards improved dispatching rules for complex shop floor scenarios - a genetic programming approach. pages 257–264, 01 2010.

[16] Rachel Hunt, Mark Johnston, and Mengjie Zhang. Evolving "less-myopic" scheduling rules for dynamic job shop scheduling with genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, page 927–934, New York, NY, USA, 2014. Association for Computing Machinery.

[17] Josiah Jacobsen-Grocott, Yi Mei, Gang Chen, and Mengjie Zhang. Evolving heuristics for dynamic vehicle routing with time windows using genetic programming. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1948–1955, 2017.

[18] Kristijan Jaklinović, Marko Đurasević, and Domagoj Jakobović. Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. *Expert Systems with Applications*, 172:114548, June 2021.

[19] Domagoj Jakobović and Kristina Marasović. Evolving priority scheduling heuristics with genetic programming. *Applied Soft Computing*, 12(9):2781–2789, September 2012.

[20] Deepak Karunakaran, Yi Mei, Gang Chen, and Mengjie Zhang. Evolving dispatching rules for dynamic job shop scheduling with uncertain processing times. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 364–371, 2017.

[21] Deepak Karunakaran, Yi Mei, Gang Chen, and Mengjie Zhang. Toward evolving dispatching rules for dynamic job shop scheduling under uncertainty. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 282–289, New York, NY, USA, 2017. Association for Computing Machinery.

[22] Jae-Ho Lee, Jae-Min Yu, and Dong-Ho Lee. A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness. *The International Journal of Advanced Manufacturing Technology*, 69(9-12):2081–2089, July 2013.

[23] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1-3):259–271, January 1990.

[24] T.W. Liao, P.C. Chang, R.J. Kuo, and C.-J. Liao. A comparison of five hybrid metaheuristic algorithms for unrelated parallel-machine scheduling and inbound trucks sequencing in multi-door cross docking systems. *Applied Soft Computing*, 21:180–193, 2014.

[25] Eunice López-Camacho, Hugo Terashima-Marin, Peter Ross, and Gabriela Ochoa. A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889, 2014.

[26] Yi Mei, Su Nguyen, Bing Xue, and Mengjie Zhang. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(5):339–353, October 2017.

[27] Yi Mei, Su Nguyen, and Mengjie Zhang. Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In *Lecture Notes in Computer Science*, pages 435–447. Springer International Publishing, 2017.

[28] Su Nguyen. A learning and optimizing system for order acceptance and scheduling. *The International Journal of Advanced Manufacturing Technology*, 86(5-8):2021–2036, January 2016.

[29] Su Nguyen, Yi Mei, Bing Xue, and Mengjie Zhang. A hybrid genetic programming algorithm for automated design of dispatching rules. *Evolutionary Computation*, 27:1–31, 06 2018.

[30] Su Nguyen, Yi Mei, and Mengjie Zhang. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, 3(1):41–66, February 2017.

References IV

[31] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 17(5):621–639, 2013.

[32] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. Dynamic multi-objective job shop scheduling: A genetic programming approach. In *Studies in Computational Intelligence*, pages 251–282. Springer Berlin Heidelberg, 2013.

[33] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. Learning iterative dispatching rules for job shop scheduling with genetic programming. *The International Journal of Advanced Manufacturing Technology*, 67(1-4):85–100, February 2013.

[34] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation*, 18(2):193–208, April 2014.

[35] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. Genetic programming for evolving due-date assignment models in job shop environments. *Evolutionary Computation*, 22(1):105–138, March 2014.

[36] Su Nguyen, Mengjie Zhang, and Kay Chen Tan. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2781–2788, 2015.

[37] Su Nguyen, Mengjie Zhang, and Kay Chen Tan. Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. *IEEE Transactions on Cybernetics*, 47(9):2951–2965, September 2017.

[38] John Park, Yi Mei, Su Nguyen, Gang Chen, and Mengjie Zhang. Investigating the generality of genetic programming based hyper-heuristic approach to dynamic job shop scheduling with machine breakdown. In *Lecture Notes in Computer Science*, pages 301–313. Springer International Publishing, December 2016.

References V

[39] John Park, Yi Mei, Su Nguyen, Gang Chen, and Mengjie Zhang. Investigating a machine breakdown genetic programming approach for dynamic job shop scheduling. In *Lecture Notes in Computer Science*, pages 253–270. Springer International Publishing, 2018.

[40] John Park, Yi Mei, Su Nguyen, Gang Chen, and Mengjie Zhang. An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Applied Soft Computing*, 63:72–86, February 2018.

[41] John Park, Su Nguyen, Mengjie Zhang, and Mark Johnston. Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. In *Lecture Notes in Computer Science*, pages 92–104. Springer International Publishing, 2015.

[42] Michael L. Pinedo. *Scheduling*. Springer US, 2012.

[43] Erik Pitzer, Andreas Beham, Michael Affenzeller, Helga Heiss, and Markus Vorderwinkler. Production fine planning using a solution archive of priority rules. In *3rd IEEE International Symposium on Logistics and Industrial Informatics*. IEEE, August 2011.

[44] Lucija Planinic, Hrvoje Backovic, Marko Durasevic, and Domagoj Jakobovic. A comparative study of dispatching rule representations in evolutionary algorithms. *IEEE Access*, pages 1–1, 2022.

[45] Lucija Planinic, Marko Durasevic, and Domagoj Jakobovic. Towards interpretable dispatching rules: Application of expression simplification methods. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, December 2021.

[46] Lucija Planinić, Marko Đurasević, and Domagoj Jakobović. On the application of ϵ-lexicase selection in the generation of dispatching rules. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 2125–2132, 2021.

[47] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza).

[48] Li-Ya Tseng, Yeh-Hao Chin, and Shu-Ching Wang. A minimized makespan scheduler with multiple factors for grid computing systems. *Expert Systems with Applications*, 36(8):11118–11130, 2009.

[49] Lucija Ulaga, Marko Đurasević, and Domagoj Jakobović. Local search based methods for scheduling in the unrelated parallel machines environment. *Expert Systems with Applications*, 199:116909, 2022.

[50] Mateja Đumić and Domagoj Jakobović. Ensembles of priority rules for resource constrained project scheduling problem. *Applied Soft Computing*, 110:107606, October 2021.

[51] Mateja Đumić, Dominik Šišejković, Rebeka Čorić, and Domagoj Jakobović. Evolving priority rules for resource constrained project scheduling problem with genetic programming. *Future Generation Computer Systems*, 86:211–221, September 2018.

[52] Marko Đurasević and Domagoj Jakobović. Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. *Genetic Programming and Evolvable Machines*, 19(1-2):53–92, April 2017.

[53] Marko Đurasević and Domagoj Jakobović. A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications*, 113:555–569, December 2018.

[54] Marko Đurasević and Domagoj Jakobović. Creating dispatching rules by simple ensemble combination. *Journal of Heuristics*, 25(6):959–1013, May 2019.

[55] Marko Đurasević and Domagoj Jakobović. Automatic design of dispatching rules for static scheduling conditions. *Neural Computing and Applications*, 33(10):5043–5068, August 2020.

[56] Marko Đurasević and Domagoj Jakobović. Comparison of schedule generation schemes for designing dispatching rules with genetic programming in the unrelated machines environment. *Applied Soft Computing*, 96:106637, November 2020.

[57] Marko Đurasević, Domagoj Jakobović, and Karlo Knežević. Adaptive scheduling on unrelated machines with genetic programming. *Applied Soft Computing*, 48:419–430, November 2016.

[58] Marko Đurasević and Domagoj Jakobović. Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. *Genetic Programming and Evolvable Machines*, 19(1-2):9–51, September 2017.

References VII

[59] Rebeka Čorić, Mateja Đumić, and Domagoj Jakobović. Genetic programming hyperheuristic parameter configuration using fitness landscape analysis. *Applied intelligence (Boston)*, 51 (10):7402–7426, 2021.

[60] Ivan Vlašić, Marko Đurasević, and Domagoj Jakobović. Improving genetic algorithm performance by population initialisation with dispatching rules. *Computers & Industrial Engineering*, 137:106030, November 2019.

[61] Ivan Vlašić, Marko Đurasević, and Domagoj Jakobović. A comparative study of solution representations for the unrelated machines environment. *Computers & Operations Research*, 123:105005, November 2020.

[62] Daniel Yska, Yi Mei, and Mengjie Zhang. Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, July 2018.

[63] Fangfang Zhang, Yi Mei, Su Nguyen, Kay Chen Tan, and Mengjie Zhang. Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics*, pages 1–14, 2021.

[64] Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling. In *Lecture Notes in Computer Science*, pages 262–278. Springer International Publishing, 2020.

[65] Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. *IEEE Transactions on Cybernetics*, pages 1–15, 2021.

[66] Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 25(3):552–566, 2021.

[67] Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics*, 51(4):1797–1811, April 2021.

References VIII

[68] Fangfang Zhang, Yi Mei, and Mengjie Zhang. Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. In *AI 2018: Advances in Artificial Intelligence*, pages 766–772. Springer International Publishing, 2018.

[69] Fangfang Zhang, Yi Mei, and Mengjie Zhang. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, July 2019.

[70] Fangfang Zhang, Su Nguyen, Yi Mei, and Mengjie Zhang. *Genetic Programming for Production Scheduling*. Springer Singapore, 2021.

[71] Marko Đurasević and Domagoj Jakobović. Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics. *Journal of Computational Science*, 61:101649, 2022.