AUTOTUNING USANDO BUSCA ESTOCÁSTICA LOCAL E PARALELISMO NA LINGUAGEM JULIA

Pedro Bruel
phrb@ime.usp.br

Alfredo Goldman
gold@ime.usp.br

29 de Setembro de 2015

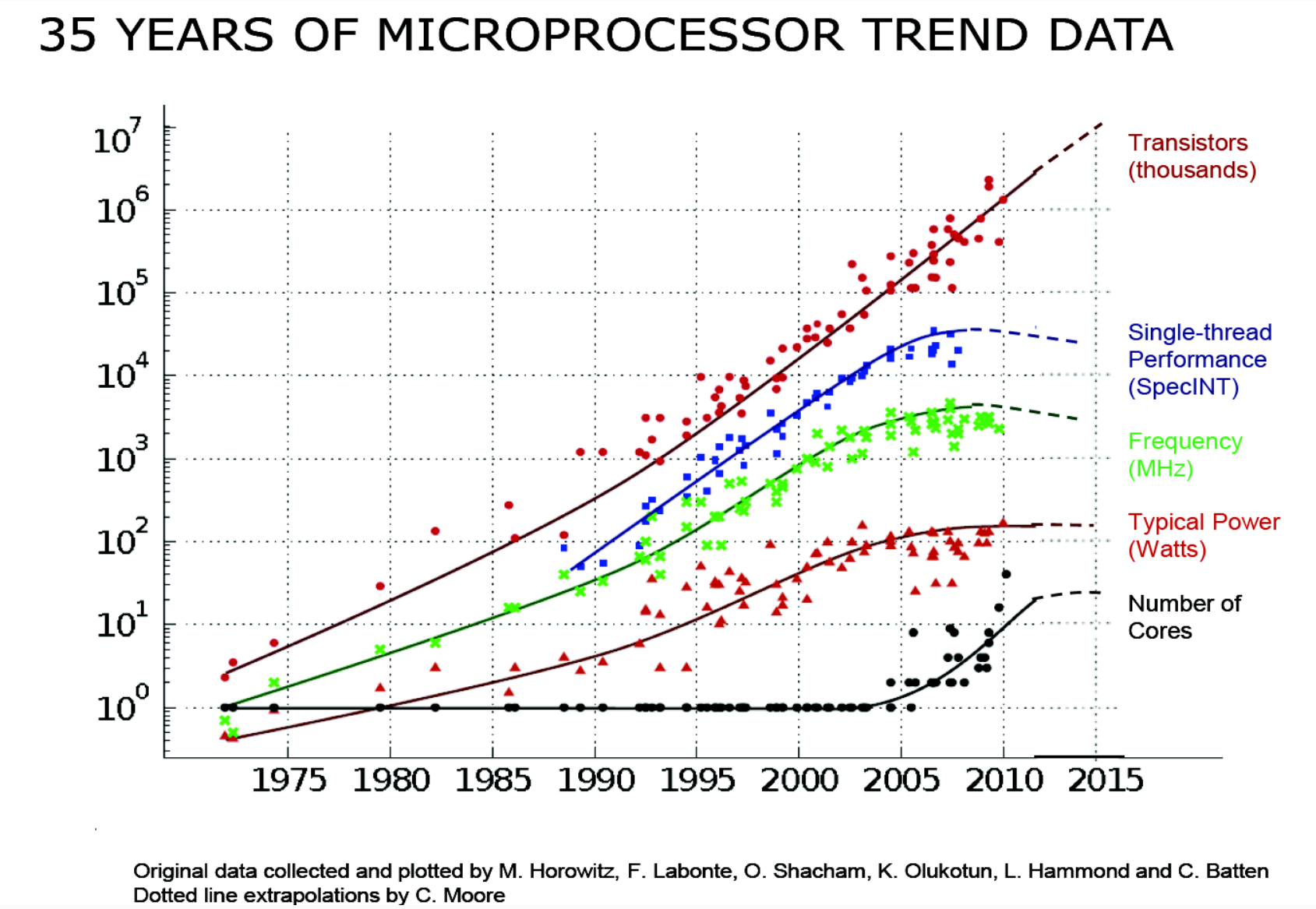
Instituto de Matemática e Estatística
Universidade de São Paulo

ROTEIRO

1. Motivação e Perguntas de Pesquisa
2. Autotuning & OpenTuner
3. Busca Estocástica Local
4. StochasticSearch.jl
5. Trabalhos Futuros

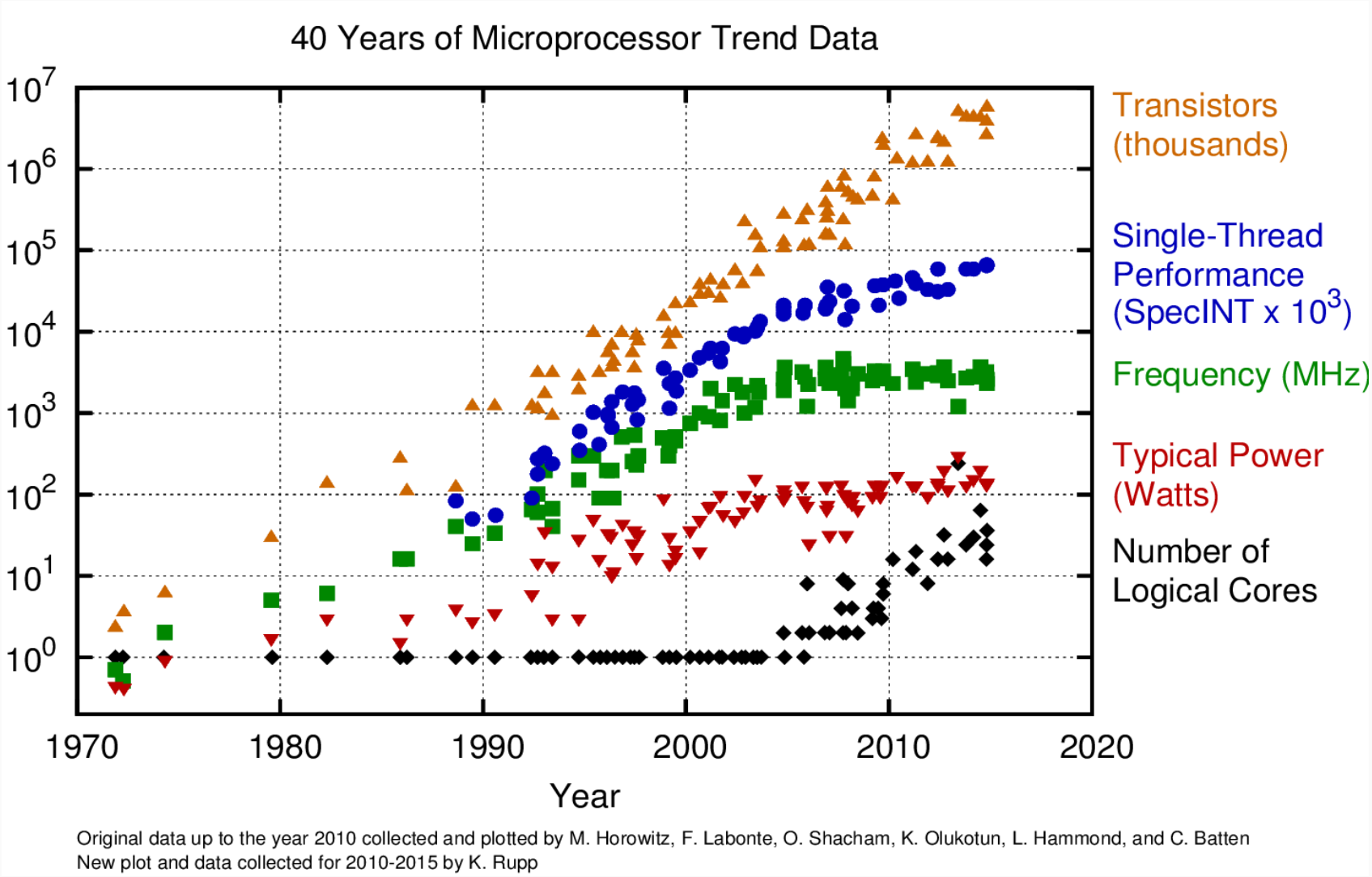
MOTIVAÇÃO

- Modelos de programação paralela e distribuída
- Arquiteturas multi-core e co-processadores
- Autotuning com programação paralela e distribuída
- Problemas computacionalmente difíceis
- Estudar o desempenho da Busca Estocástica Local



http://karlrupp.net/2015/06/40-years-of-microprocessor-trend-data

MOTIVAÇÃO



http://karlrupp.net/2015/06/40-years-of-microprocessor-trend-data

PERGUNTAS DE PESQUISA

Visão geral:

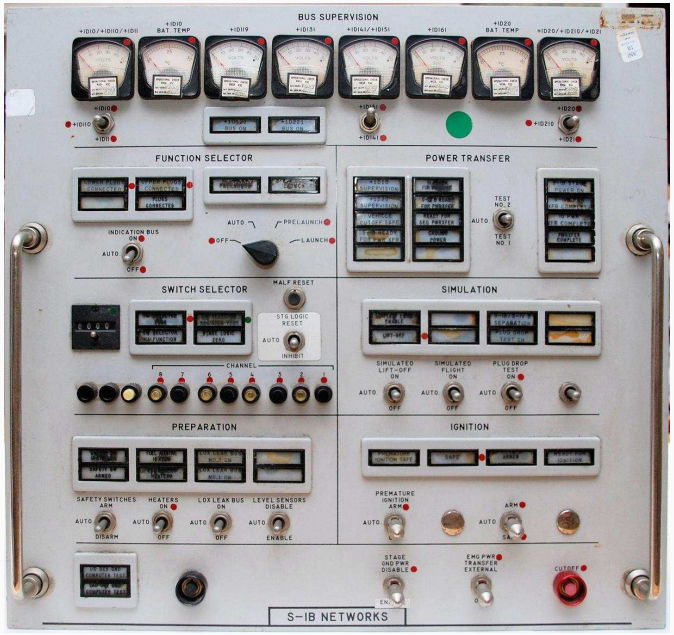
- RQ0: Como e quando aplicar técnicas de autotuning a problemas difíceis e arquiteturas heterogêneas?
- RQ1: Técnicas de busca estocástica apresentam alguma vantagem em relação a outras técnicas de autotuning?
- RQ2**a**: Como utilizar programação paralela e distribuída para melhorar o desempenho de autotuners?
- RQ2**b**: Para quais domínios de problema isso é vantajoso?
- RQ3: Como aproveitar medições de desempenho obtidas em arquiteturas diferentes da arquitetura alvo?

PERGUNTAS DE PESQUISA

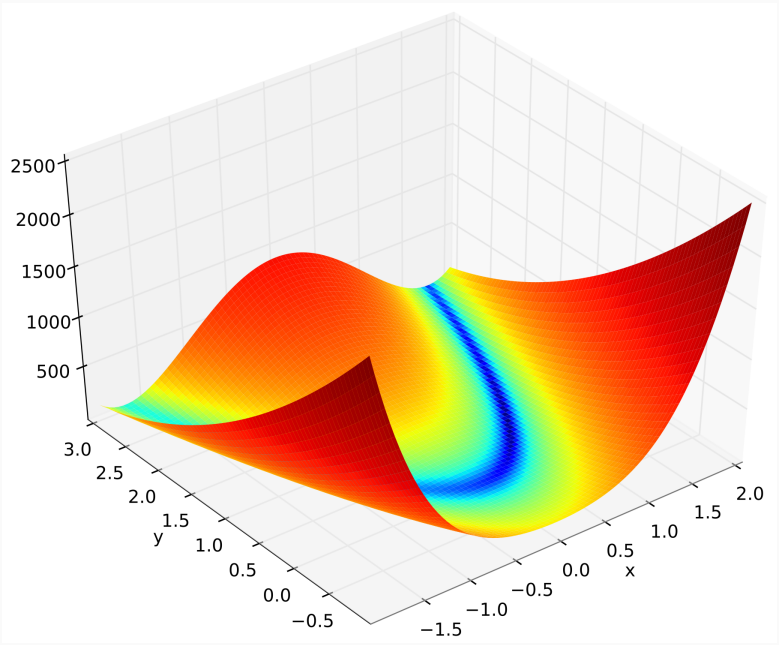
Com este trabalho em andamento, gostaríamos de avançar em direção a responder:

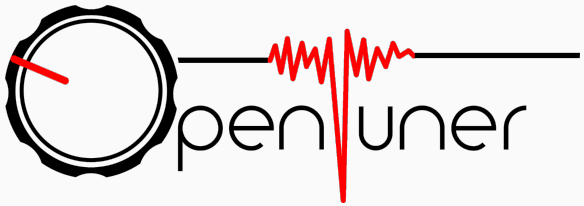
- RQ1: Técnicas de busca estocástica apresentam alguma vantagem em relação a outras técnicas de autotuning?
- RQ2a: Como utilizar programação paralela e distribuída para melhorar o desempenho de autotuners?
- RQ2b: Para quais domínios de problema isso é vantajoso?

Configurações e Otimizações



Espaço de Busca





- Arcabouço para autotuning
- Independente de domínio
- Conjuntos de técnicas de busca
- Compartilhamento de resultados entre técnicas
- Execução sequencial

Código sob Licença MIT: http://github.com/jansel/opentuner

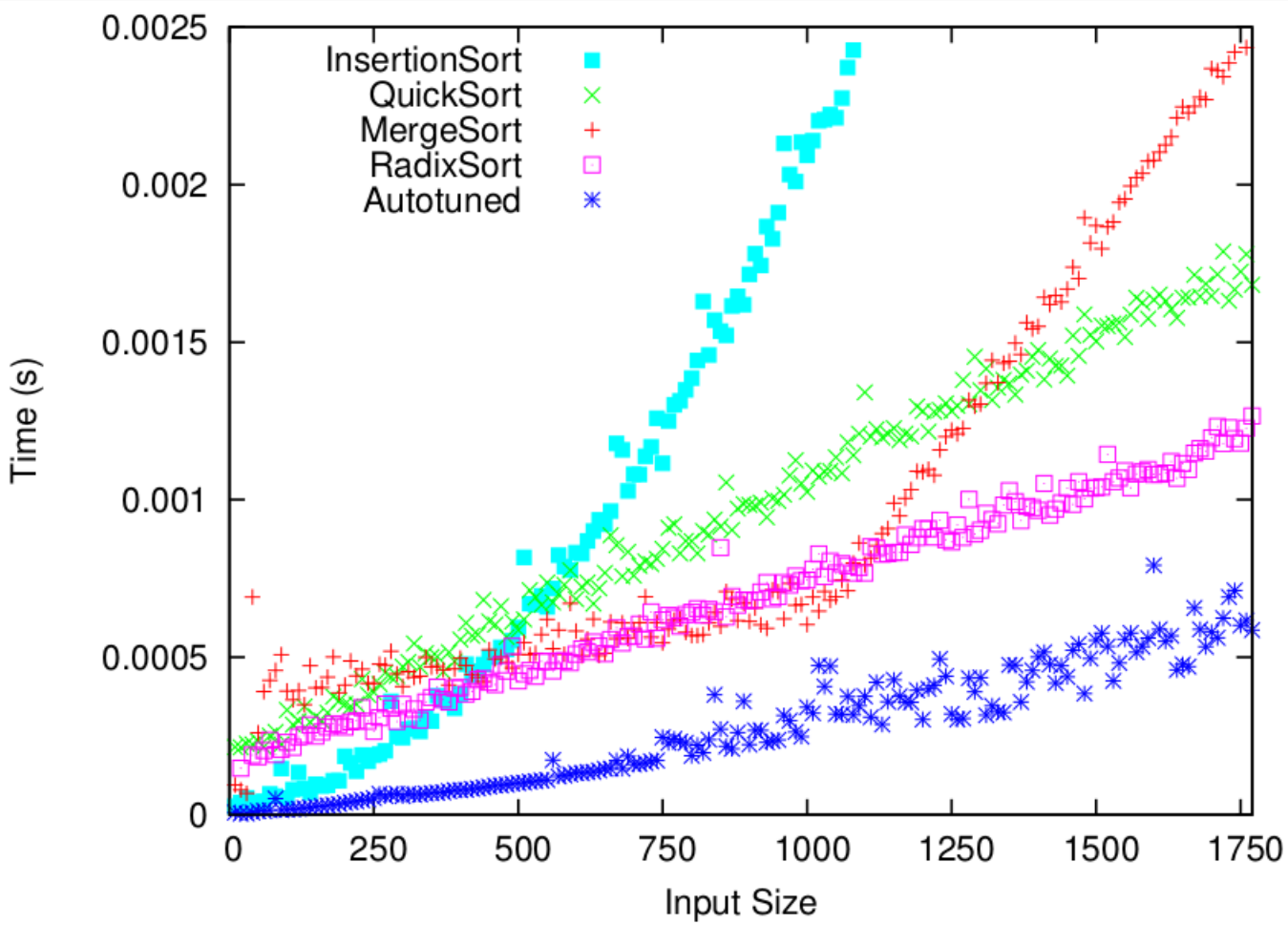


Figura 1: Otimizando combinações de algoritmos recursivos de ordenação

---

Ansel, Jason, et al. "Opentuner: An extensible framework for program autotuning."Proceedings of the 23rd ICPAC. ACM, 2014.

Trabalho com OpenTuner em andamento:

- Computação Distribuída (Google Compute Engine):

  github.com/phrb/gce_autotuning_example

  github.com/phrb/measurement_client

  github.com/phrb/gce_interface

  github.com/phrb/measurement-server

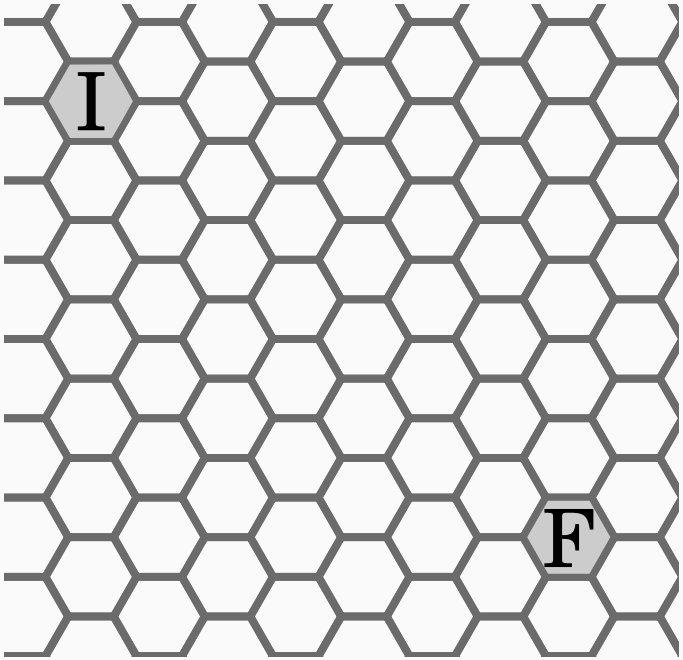
- Parâmetros de compilação em GPUs (CUDA):

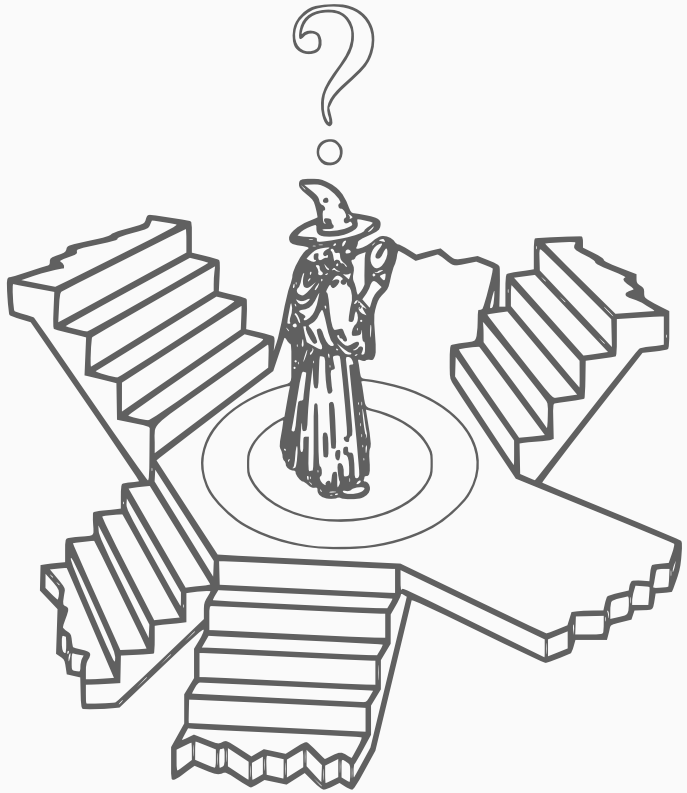
  github.com/phrb/gpu-autotuning

- Resolvedores de SAT e TSP:

  github.com/phrb/sat-opentuner github.com/phrb/stochasticsearch-docs

BUSCA ESTOCÁSTICA LOCAL





Adaptado de: Hoos, Holger H., and Thomas Stützle. *Stochastic local search: Foundations & applications.* Elsevier, 2004.

Os algoritmos de Busca Estocástica Local consistem de heurísticas para a exploração de um espaço de busca.

- Naturalmente paralelas e distribuídas
- Podem ser combinadas para gerar novas heurísticas
- Fáceis de implementar
- Estado da Arte[1] na solução de problemas computacionalmente difíceis

[1] Hutter, Frank, et al. *ParamILS: an automatic algorithm configuration framework.* Journal of Artificial Intelligence Research 36.1 (2009): 267-306.

COMPOSIÇÃO

Alguns blocos de construção:

- First Improvement
- Best Improvement
- Probabilistic Improvement
- Greedy Construction
- Random Walk

Exemplo de composição:

$ProbabilisticImprovement(Temperatura) \rightarrow SimulatedAnnealing$

Principais referências:

[1] Hoos, Holger H., and Thomas Stützle. *Stochastic local search: Foundations & applications.* Elsevier, 2004.

[2] Hoos, Holger H., and Thomas Stützle. *Stochastic Local Search Algorithms: An Overview.* Springer Handbook of Computational Intelligence. Springer Berlin Heidelberg, 2015. 1085-1105.

[3] Hutter, Frank, et al. *ParamILS: an automatic algorithm configuration framework.* Journal of Artificial Intelligence Research 36.1 (2009): 267-306.

Por que uma nova implementação?

OpenTuner:

- Permite pouco controle do fluxo de execução
- Python: Global Interpreter Lock
- Busca e Medições sequenciais
- Não é fácil de paralelizar

Outros motivos:

- Estudar a composição e o desempenho de técnicas de Busca Estocástica Local
- Tentativa de reproduzir resultados

Por que em Julia?

- Linguagem de alto nível
- Programação concorrente, paralela e distribuída
- Ótimo desempenho
- É um brinquedo novo
- Comunidade bastante ativa

O paralelismo em Julia é feito através da troca de mensagens entre processos:

- Diferente do MPI, o usuário controla explicitamente apenas um processo
- Remote calls: executam uma expressão em outro processo e devolvem uma referência remota (RemoteRef)
- Remote references: Contém uma referência para objetos em outros processos

---

http://docs.julialang.org/en/stable/stdlib/parallel
http://docs.julialang.org/en/stable/manual/parallel-computing/

Um ClusterManager é responsável por:

- Lançar processos (workers) Julia numa rede
- Gerenciar eventos e comunicação entre processos
- Gerenciar o transporte de dados

Algumas funções e objetos disponíveis:

```
LocalManager::ClusterManager
SSHManager::ClusterManager

addprocs()
rmprocs()
procs()
```

---

http://docs.julialang.org/en/stable/stdlib/parallel
http://docs.julialang.org/en/stable/manual/parallel-computing/

Chamadas de funções:

```
Task(func)
remotecall(id, func, args..)
remotecall_fetch(id, func, args..)
remotecall_wait(id, func, args..)

put!(RemoteRef, value)
take!(RemoteRef)

fetch(RemoteRef)
wait(RemoteRef)

produce(value)
consume(Task, values..)
```

---

http://docs.julialang.org/en/stable/stdlib/parallel
http://docs.julialang.org/en/stable/manual/parallel-computing/

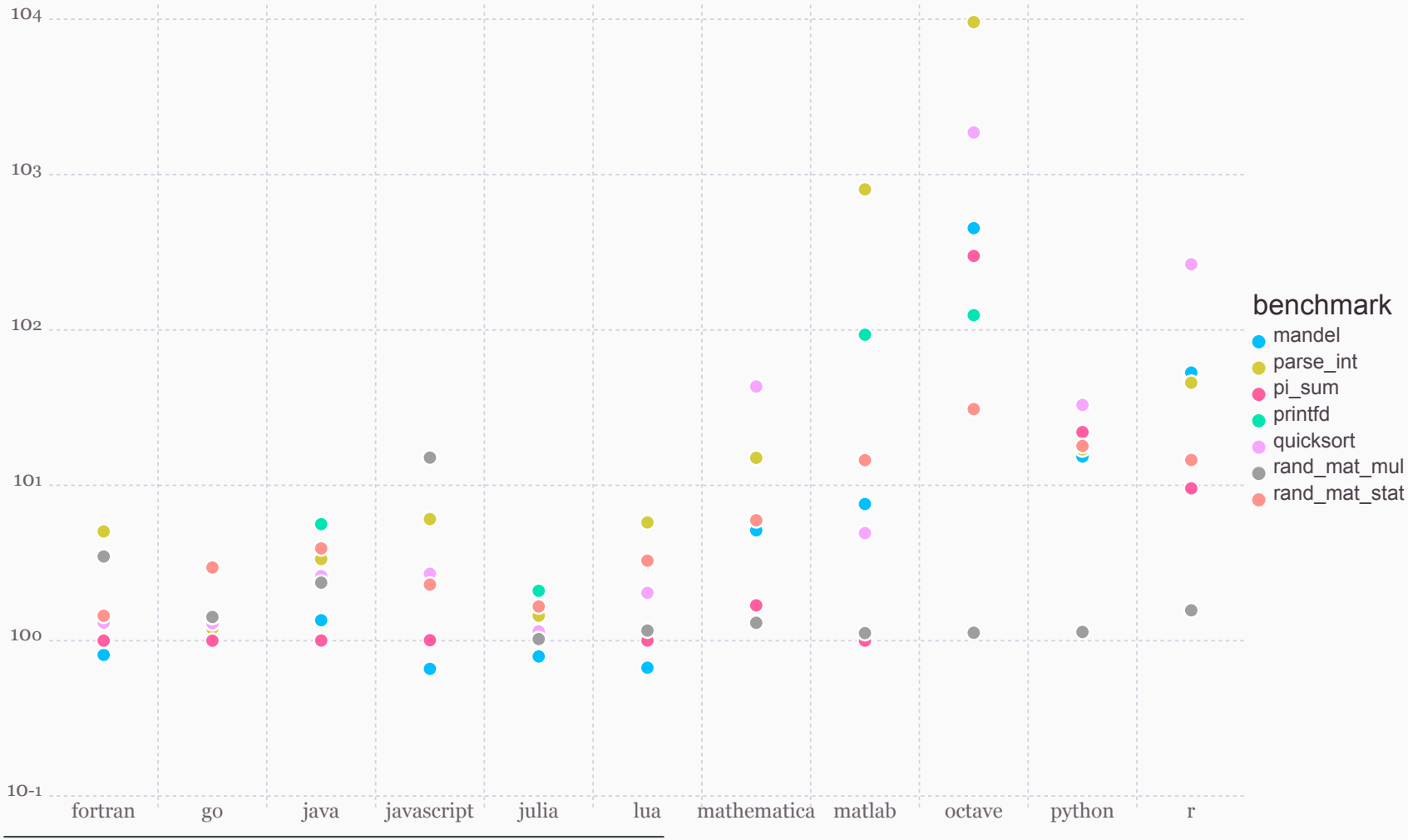
Usando macros:

```
@task()
@spawn()
@spawnat()

@fetch()      = fetch(@spawn expr)
@fetchfrom() = fetch(@spawnat p expr)

@async()
@sync()

@parallel()
@everywhere()
```

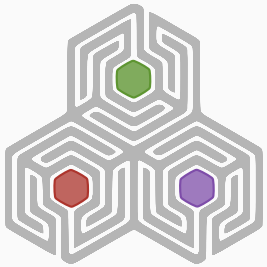
---

http://docs.julialang.org/en/stable/stdlib/parallel
http://docs.julialang.org/en/stable/manual/parallel-computing/

Comparações de desempenho relativo à linguagem C:



http://julialang.org/benchmarks



StochasticSearch.jl

- Pacote para Busca Estocástica Local
- Independente de domínio
- Conjuntos de técnicas de busca
- Técnicas de execução independente
- Técnicas de busca modulares
- Execução paralela e distribuída
- Maior controle do fluxo de execução

- Código sob Licença MIT:

  http://github.com/phrb/StochasticSearch.jl

- Disponível no repositório oficial:

```
julia> Pkg.add("StochastichSearch")
```

- Testes de Unidade: 97% de cobertura:

  https://coveralls.io/github/phrb/StochasticSearch.jl

- Integração Contínua (TravisCI):

  https://travis-ci.org/phrb/StochasticSearch.jl

COMPOSIÇÃO

Blocos de construção:

- First Improvement
- Probabilistic Improvement
- Greedy Construction
- Random Walk
- TODO: Best Improvement

COMPOSIÇÃO

Técnicas de busca usando os blocos:

DONE:

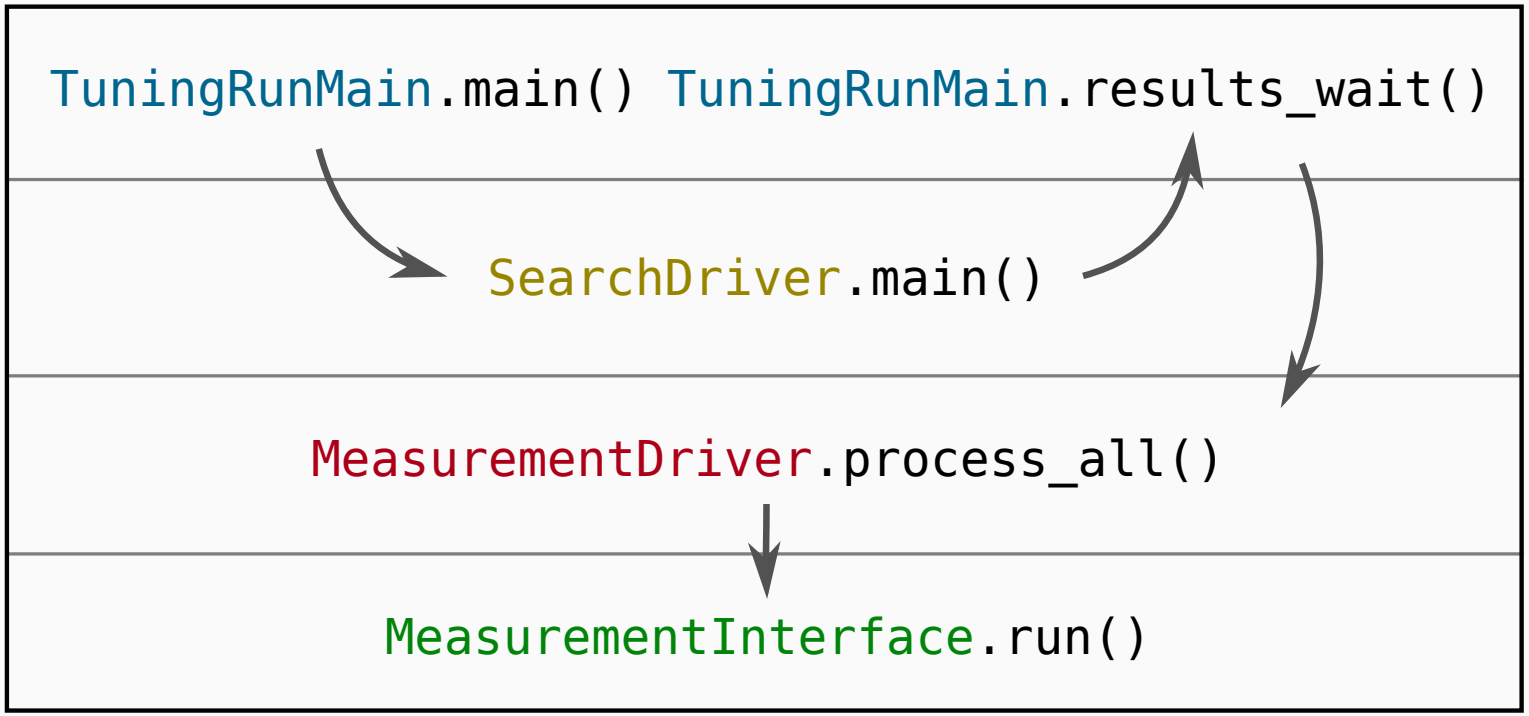
- Iterative First Improvement
- Iterative Probabilistic Improvement
- Iterative Greedy Construction
- Randomized First Improvement
- Simulated Annealing

TODO:

- Randomized Best Improvement
- Iterative Probabilistic Improvement
- Iterated Local Search
- Tabu Search
- Ant Colony Optimization
- Particle Swarm Optimization
- ...

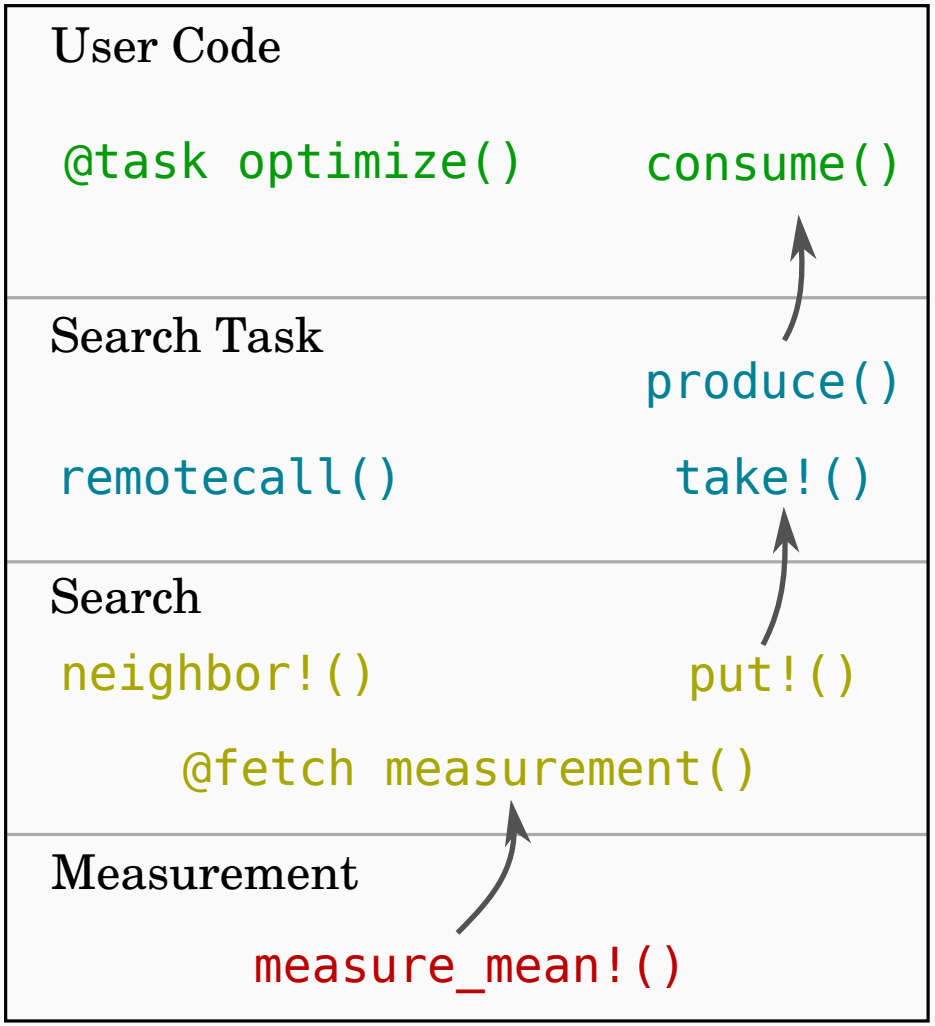
COMPARAÇÕES: FLUXO DE EXECUÇÃO

Obtenção de um resultado no OpenTuner, simplificadamente:

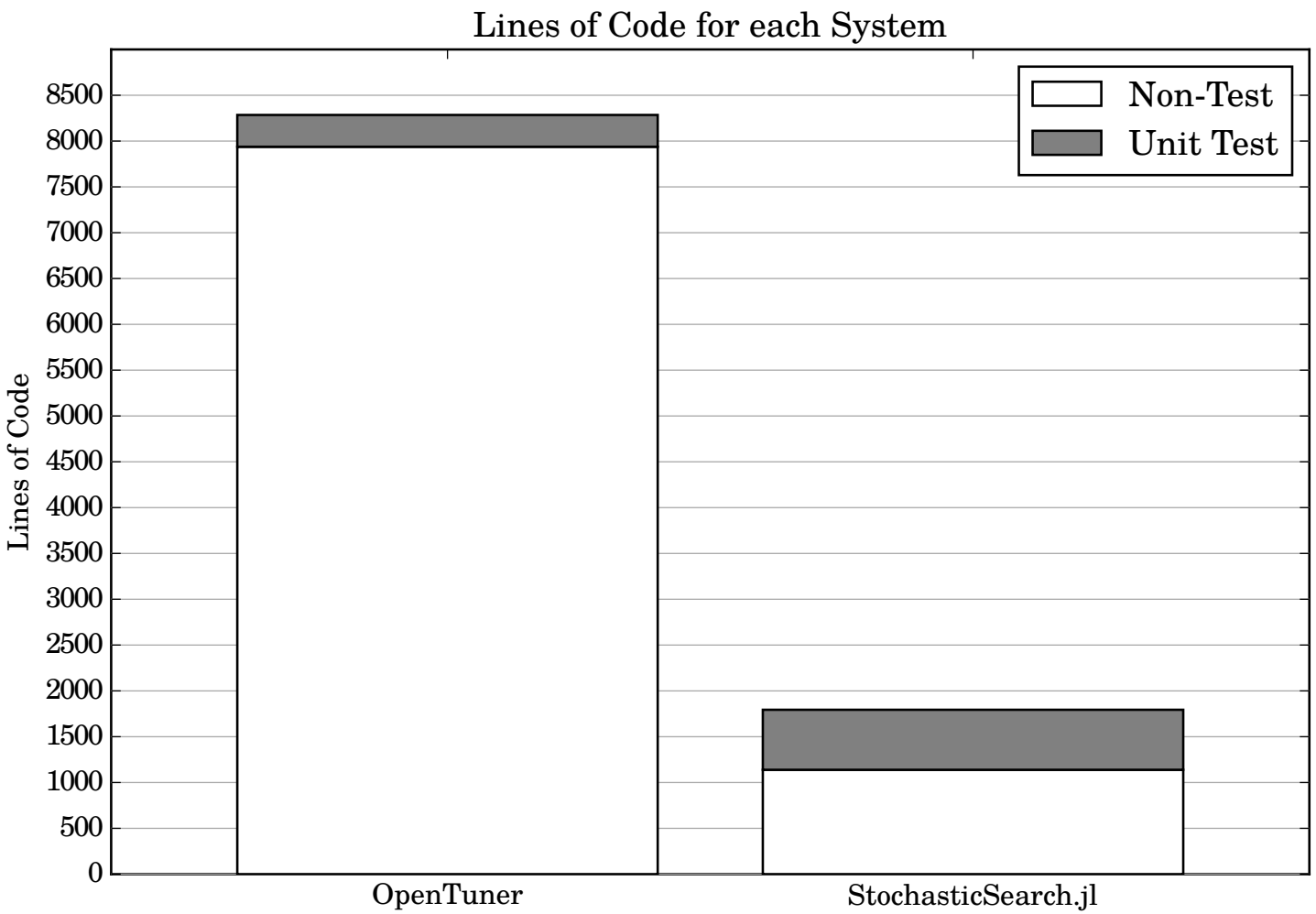


COMPARAÇÕES: FLUXO DE EXECUÇÃO

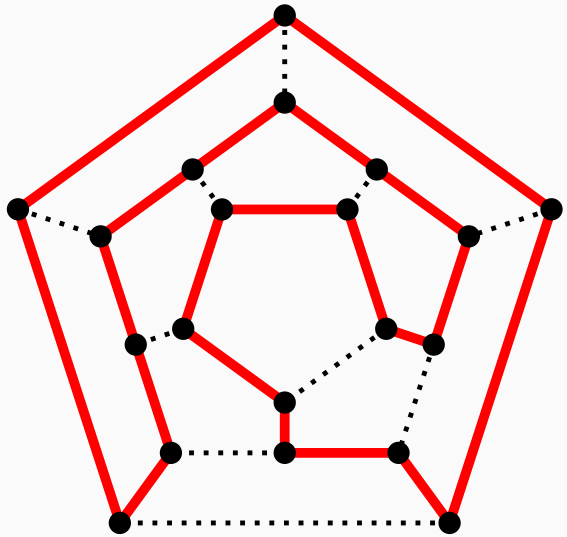
Obtenção de um resultado no StochasticSearch.jl, simplificadamente:



COMPARAÇÕES: ESFORÇO DE IMPLEMENTAÇÃO



COMPARAÇÕES: TRAVELLING SALESPERSON PROBLEM



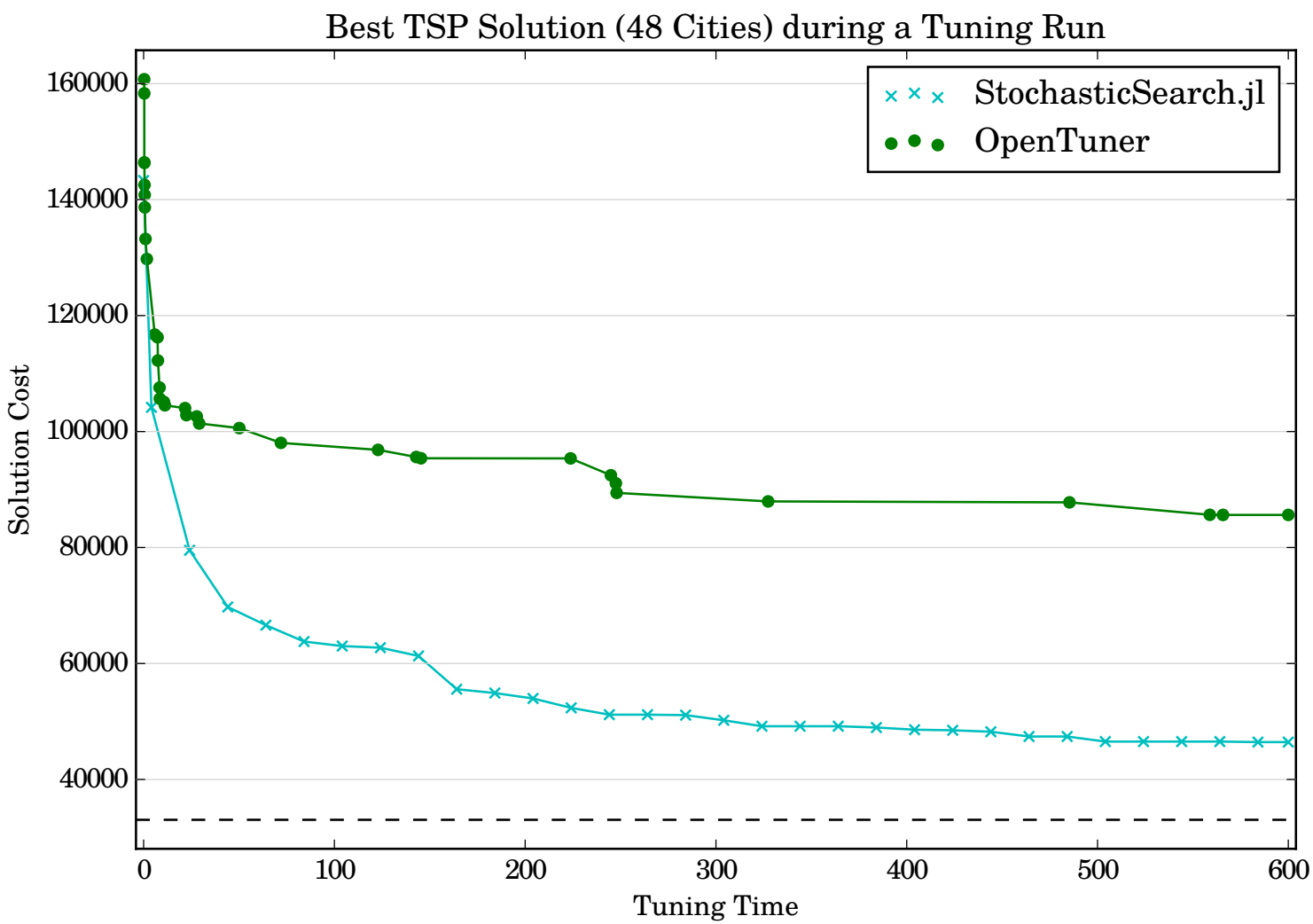
Comparação do desempenho de autotuners que resolvem instâncias do TSP, implementados com o OpenTuner e com o StochasticSearch.jl.

As soluções são representadas por permutações das cidades.

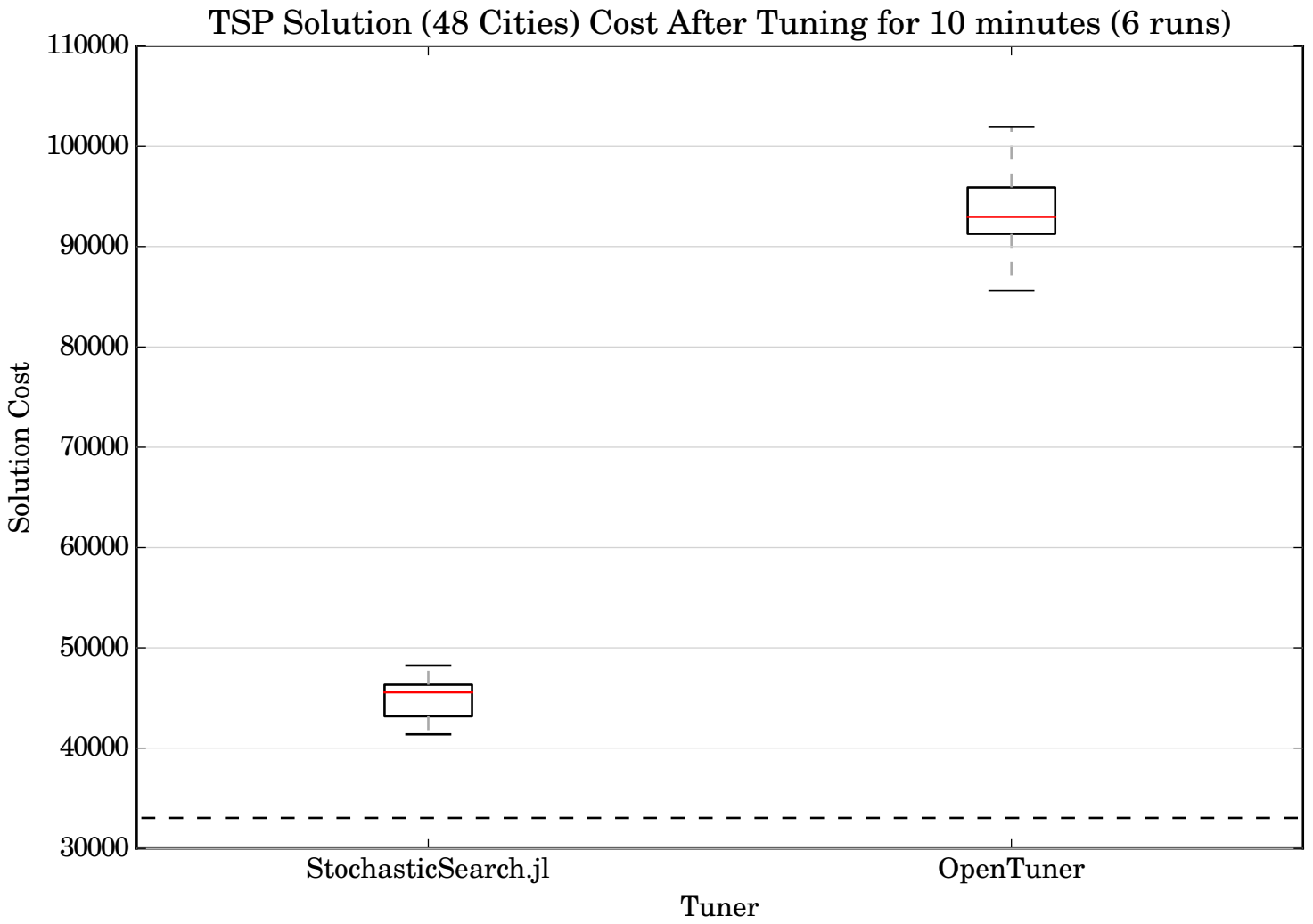
Disponível em:

https://github.com/phrb/stochasticsearch-docs

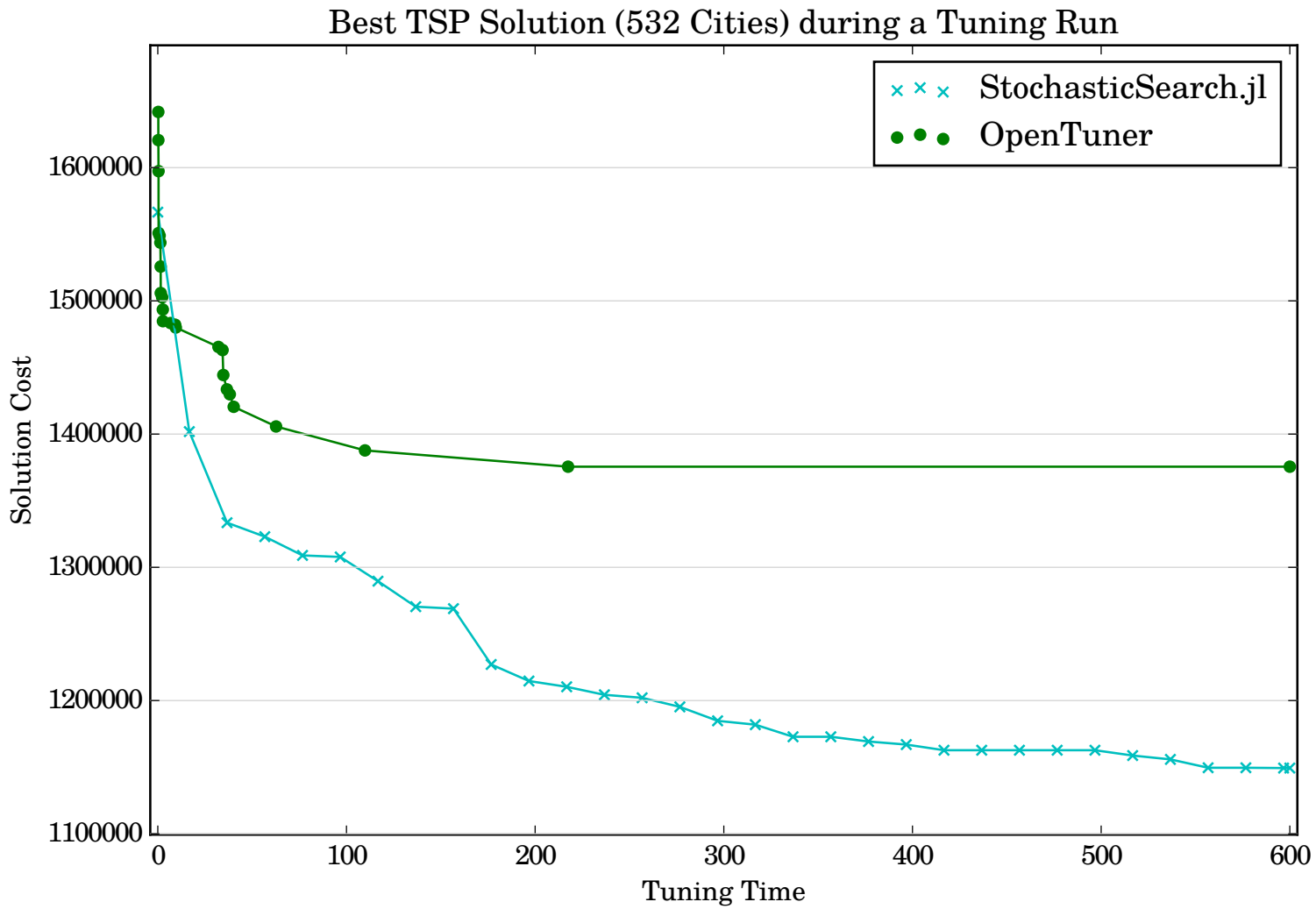
COMPARAÇÕES: DESEMPENHO TSP (48 CIDADES)



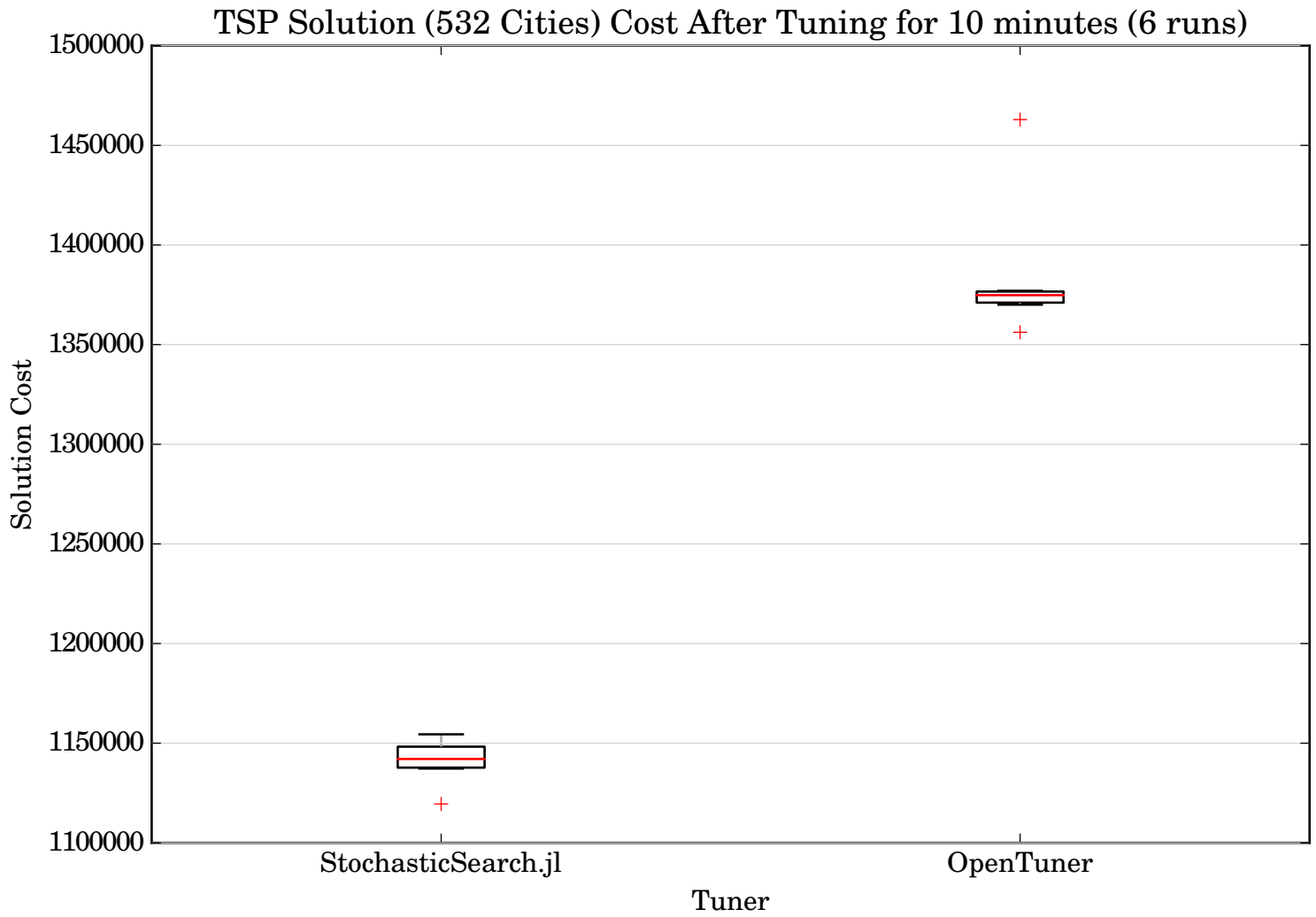
COMPARAÇÕES: DESEMPENHO TSP (48 CIDADES)



COMPARAÇÕES: DESEMPENHO TSP (532 CIDADES)



COMPARAÇÕES: DESEMPENHO TSP (532 CIDADES)



TRABALHOS FUTUROS

- Implementar Busca Estocástica Local no OpenTuner
- Implementar ClusterManagers para GPUs e Nuvem
- Parâmetros de compilação CUDA
- Resolvedores SAT
- Experimentos com domínios diferentes
- Experimentos com execução distribuída
- Aumentar a cobertura de testes
- Implementar técnicas de busca mais “poderosas”

Responder às perguntas:

- RQ1: Técnicas de busca estocástica apresentam alguma vantagem em relação a outras técnicas de autotuning?
- RQ2a: Como utilizar programação paralela e distribuída para melhorar o desempenho de autotuners?
- RQ2b: Para quais domínios de problema isso é vantajoso?

Obrigado!

AUTOTUNING USANDO BUSCA ESTOCÁSTICA LOCAL E PARALELISMO NA LINGUAGEM JULIA

Pedro Bruel
phrb@ime.usp.br

Alfredo Goldman
gold@ime.usp.br

29 de Setembro de 2015

Instituto de Matemática e Estatística
Universidade de São Paulo