

# AUTOTUNING USANDO BUSCA ESTOCÁSTICA LOCAL E PARALELISMO NA LINGUAGEM JULIA

---

Pedro Bruel

[phrb@ime.usp.br](mailto:phrb@ime.usp.br)

Alfredo Goldman

[gold@ime.usp.br](mailto:gold@ime.usp.br)

29 de Setembro de 2015



Instituto de Matemática e Estatística  
Universidade de São Paulo

1. Introdução
2. OpenTuner
3. Busca Estocástica Local
4. StochasticSearch.jl

## MOTIVAÇÃO E CONTEXTO

---

# MOTIVAÇÃO

---

- Modelos de programação paralela e distribuída
- Arquiteturas multi-core e co-processadores

# MOTIVAÇÃO

---

- Modelos de programação paralela e distribuída
- Arquiteturas multi-core e co-processadores
- Autotuning com programação paralela e distribuída

# MOTIVAÇÃO

---

- Modelos de programação paralela e distribuída
- Arquiteturas multi-core e co-processadores
- Autotuning com programação paralela e distribuída
- Problemas computacionalmente difíceis

# MOTIVAÇÃO

---

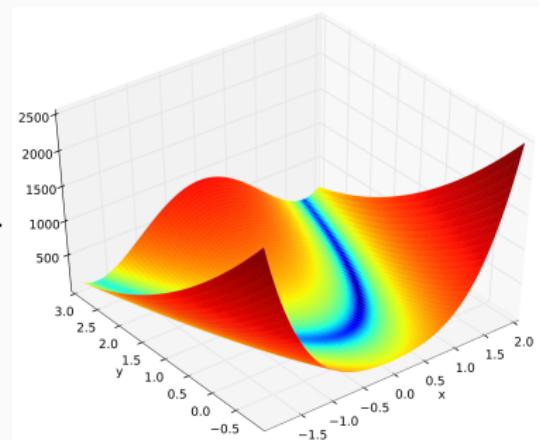
- Modelos de programação paralela e distribuída
- Arquiteturas multi-core e co-processadores
- Autotuning com programação paralela e distribuída
- Problemas computacionalmente difíceis
- Estudar o desempenho da Busca Estocástica Local

# AUTOTUNING

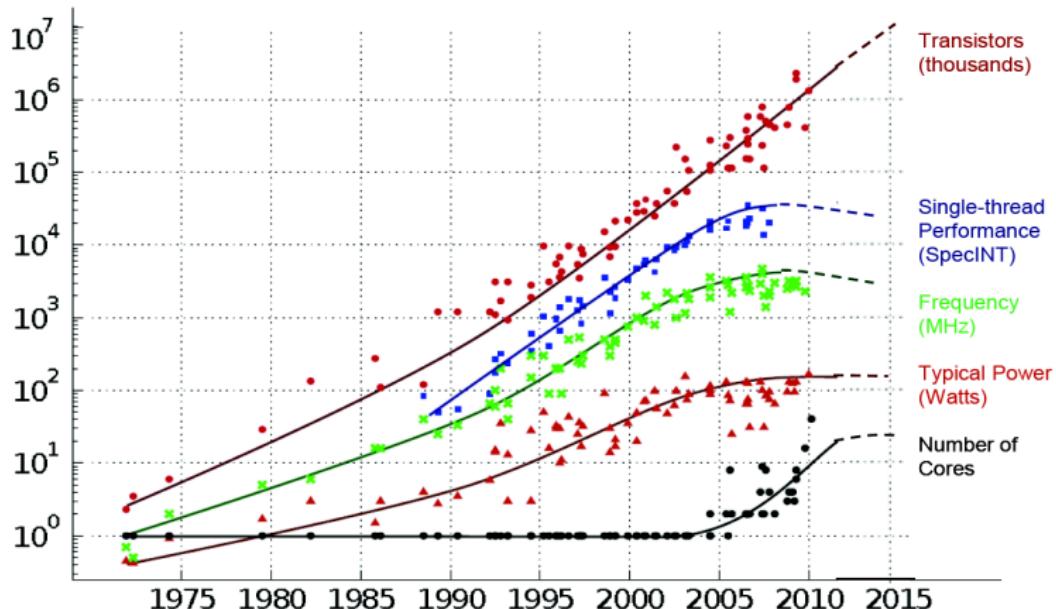
Configurações e Otimizações



Espaço de Busca

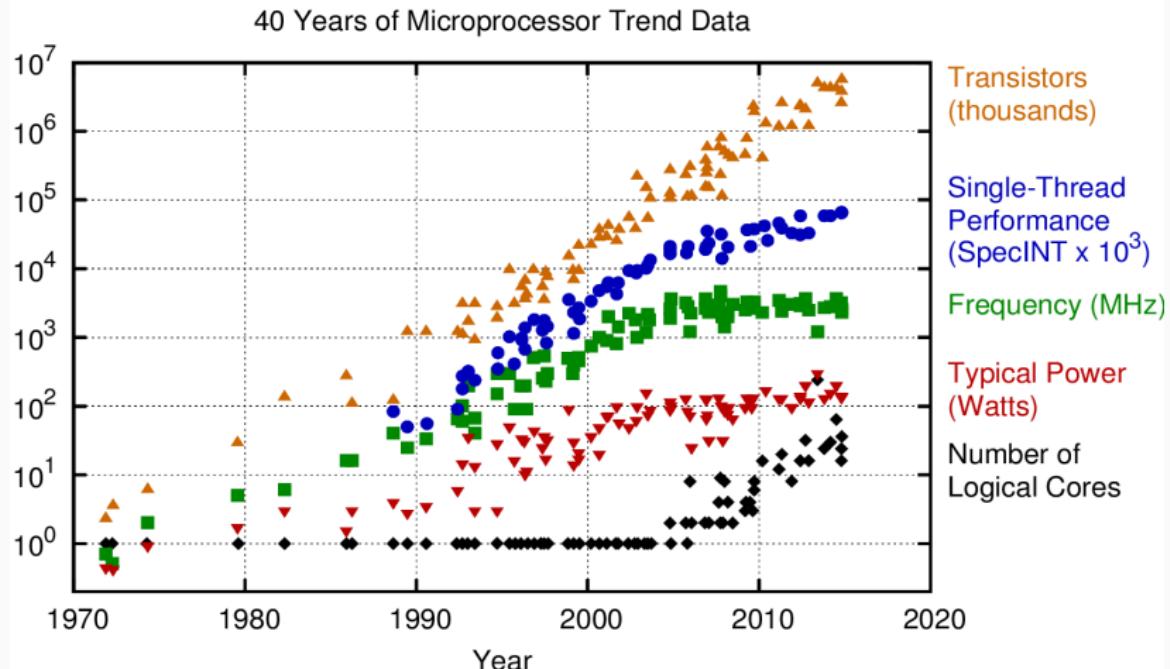


## 35 YEARS OF MICROPROCESSOR TREND DATA



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten  
Dotted line extrapolations by C. Moore

# MOTIVAÇÃO



Karl Rupp: <http://goo.gl/WLdveJ>

## PERGUNTAS DE PESQUISA

---

Visão geral:

- **RQ0:** Como e quando aplicar técnicas de autotuning a problemas difíceis e arquiteturas heterogêneas?

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?

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?
- **RQ2a:** Como utilizar programação paralela e distribuída para melhorar o desempenho de autotuners?
- **RQ2b:** Para quais domínios de problema isso é vantajoso?

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?
- **RQ2a:** Como utilizar programação paralela e distribuída para melhorar o desempenho de autotuners?
- **RQ2b:** Para quais domínios de problema isso é vantajoso?
- **RQ3:** Como aproveitar medições de desempenho obtidas em arquiteturas diferentes da arquitetura alvo?

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?

OPENTUNER

---



- Arcabouço para autotuning
- Independente de domínio

---

Código sob Licença MIT: [github.com/jansel/opentuner](https://github.com/jansel/opentuner)



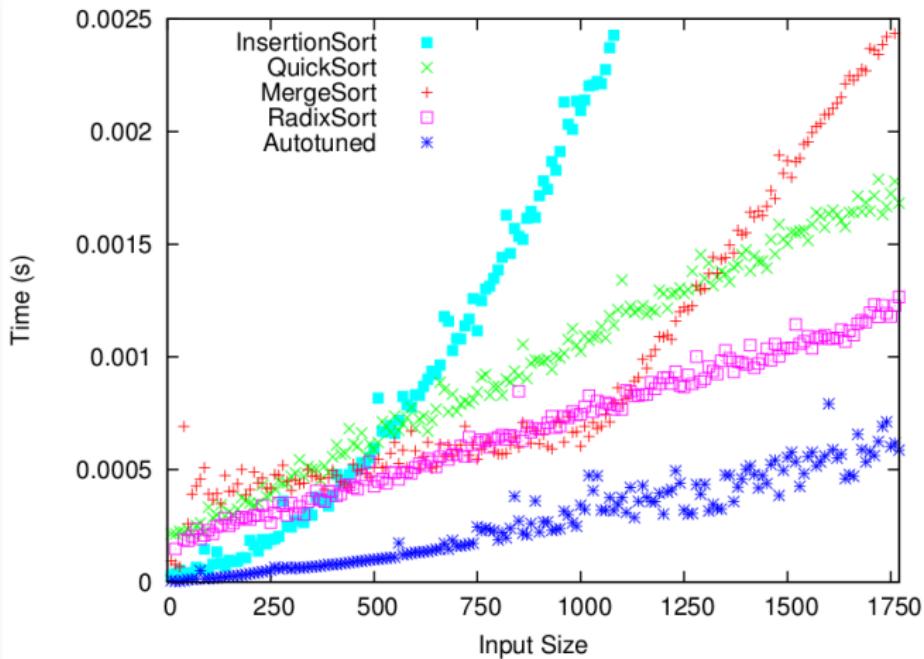
- Arcabouço para autotuning
- Independente de domínio
- Conjuntos de técnicas de busca
- Compartilhamento de resultados entre técnicas

---

Código sob Licença MIT: [github.com/jansel/opentuner](https://github.com/jansel/opentuner)



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



**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 (disponível no [GitHub](#)):

- Computação Distribuída ([Google Compute Engine](#)):

phrb/gce\_autotuning\_example

phrb/measurement\_client

phrb/gce\_interface

phrb/measurement-server

Trabalho com OpenTuner em andamento (disponível no [GitHub](#)):

- Computação Distribuída ([Google Compute Engine](#)):  
phrb/gce\_autotuning\_example  
phrb/measurement\_client  
phrb/gce\_interface  
phrb/measurement-server
- Parâmetros de compilação em GPUs ([CUDA](#)):  
phrb/gpu-autotuning

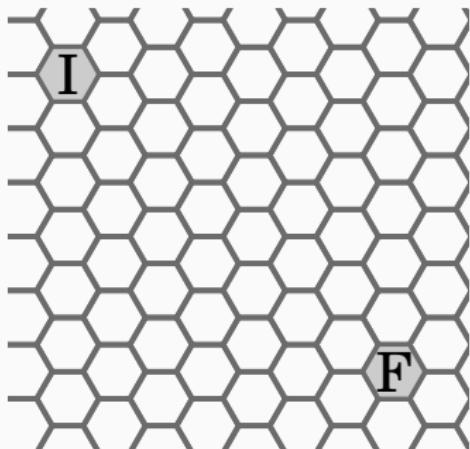
Trabalho com OpenTuner em andamento (disponível no [GitHub](#)):

- Computação Distribuída ([Google Compute Engine](#)):  
phrb/gce\_autotuning\_example  
phrb/measurement\_client  
phrb/gce\_interface  
phrb/measurement-server
- Parâmetros de compilação em GPUs ([CUDA](#)):  
phrb/gpu-autotuning
- Resolvedores de [SAT](#) e [TSP](#):  
phrb/sat-opentuner  
phrb/stochasticsearch-docs

## BUSCA ESTOCÁSTICA LOCAL

---

# BUSCA ESTOCÁSTICA LOCAL



---

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

## BUSCA ESTOCÁSTICA LOCAL

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

---

<sup>1</sup>Hutter, Frank, et al. *ParamILS: an automatic algorithm configuration framework*. Journal of Artificial Intelligence Research. 2009.

<sup>2</sup>Dubois-Lacoste, et al. *On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP*. Proceedings of the 2015 GECCO. ACM, 2015.

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

---

<sup>1</sup>Hutter, Frank, et al. *ParamILS: an automatic algorithm configuration framework*. Journal of Artificial Intelligence Research. 2009.

<sup>2</sup>Dubois-Lacoste, et al. *On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP*. Proceedings of the 2015 GECCO. ACM, 2015.

## BUSCA ESTOCÁSTICA LOCAL

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**<sup>12</sup> na solução de **alguns tipos** de instâncias de problemas computacionalmente difíceis

---

<sup>1</sup>Hutter, Frank, et al. *ParamILS: an automatic algorithm configuration framework*. Journal of Artificial Intelligence Research. 2009.

<sup>2</sup>Dubois-Lacoste, et al. *On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP*. Proceedings of the 2015 GECCO. ACM, 2015.

# COMPOSIÇÃO

---

Alguns blocos de construção:

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

# COMPOSIÇÃO

---

Alguns blocos de construção:

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

Exemplos de composição:

*RandomWalk + FirstImprovement → RandomizedIterativeImprovement*

# COMPOSIÇÃO

---

Alguns blocos de construção:

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

Exemplos de composição:

*RandomWalk + FirstImprovement → RandomizedIterativeImprovement*

*ProbabilisticImprovement(Temperatura) → SimulatedAnnealing*

Principais referências:

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

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.

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.

STOCHASTICSEARCH.JL

---

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**

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

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**



Por que em Julia?

- Linguagem de alto nível
- Programação concorrente, paralela e distribuída
- Ótimo desempenho



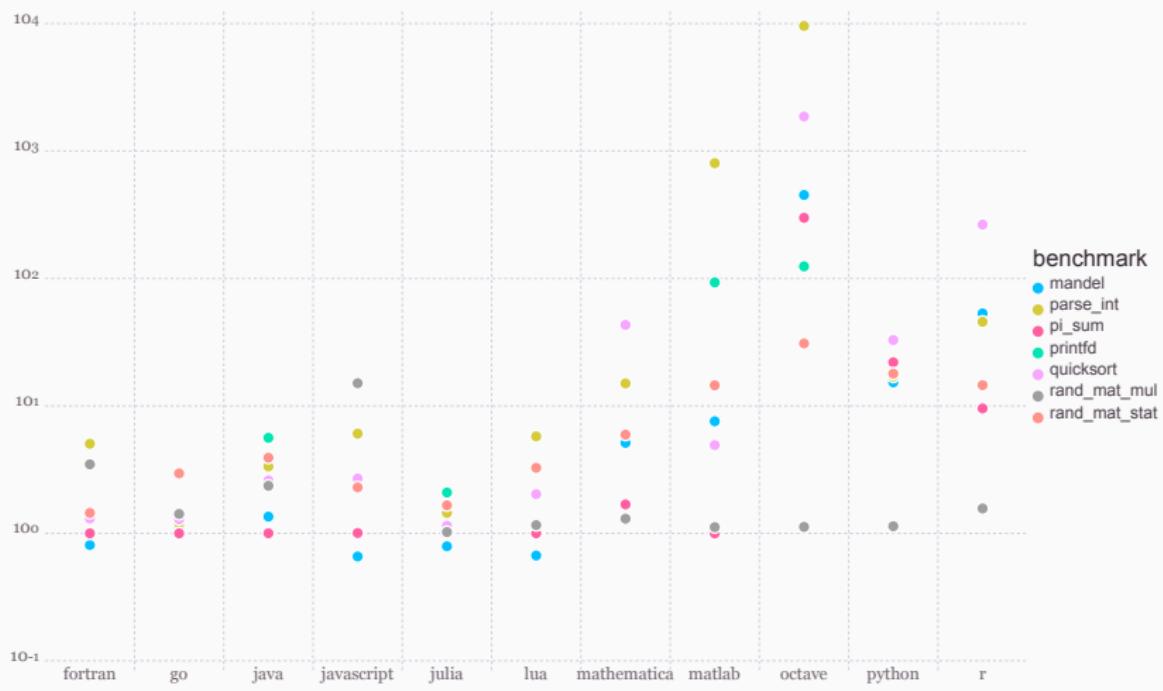
Por que em Julia?

- Linguagem de **alto nível**
- Programação **concorrente, paralela e distribuída**
- Ótimo **desempenho**
- É um brinquedo **novo**
- Comunidade bastante **ativa** (**v0.4.x-stable, v0.5-dev**)

---

Código sob Licença MIT: [github.com/JuliaLang/julia](https://github.com/JuliaLang/julia)

## Comparações de desempenho relativo à linguagem C:



# PROGRAMAÇÃO PARALELA E DISTRI- BUÍDA EM JULIA

---

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

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

- Diferente do MPI, o usuário controla explicitamente **apenas um processo**

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

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

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

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
- **Threading**: Pull request #13410

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Um **ClusterManager** é responsável por:

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

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

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()`

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Chamadas de funções:

```
remotecall(id, func, args...)
```

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Chamadas de funções:

```
remotecall(id, func, args...)
```

```
put!(RemoteRef, value)
```

```
take!(RemoteRef)
```

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Chamadas de funções:

```
remotecall(id, func, args...)
```

```
put!(RemoteRef, value)
```

```
take!(RemoteRef)
```

```
fetch(RemoteRef)
```

```
wait(RemoteRef)
```

```
remotecall_fetch(id, func, args...)
```

```
remotecall_wait(id, func, args...)
```

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Chamadas de funções:

```
remotecall(id, func, args...)
```

```
put!(RemoteRef, value)
```

```
take!(RemoteRef)
```

```
fetch(RemoteRef)
```

```
wait(RemoteRef)
```

```
remotecall_fetch(id, func, args...)
```

```
remotecall_wait(id, func, args...)
```

```
Task(func)
```

```
produce(value)
```

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

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Usando macros:

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

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Usando macros:

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

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

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)  
[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

# PROGRAMAÇÃO PARALELA E DISTRIBUÍDA EM JULIA

Usando macros:

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

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

```
@async()
@sync()
```

```
@parallel()
@everywhere()
```

---

[docs.julialang.org/en/stable/stlib/parallel](https://docs.julialang.org/en/stable/stlib/parallel)

[docs.julialang.org/en/stable/manual/parallel-computing](https://docs.julialang.org/en/stable/manual/parallel-computing)

## DETALHES DE IMPLEMENTAÇÃO

---



StochasticSearch.jl

- Pacote para Busca Estocástica Local
- Independente de domínio
- Conjuntos de técnicas de busca



StochasticSearch.jl

- Pacote para Busca Estocástica Local
- Independente de domínio
- Conjuntos de técnicas de busca
- Técnicas de execução independente
- Técnicas de busca modulares



# 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



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:  
[github.com/phrb/StochasticSearch.jl](https://github.com/phrb/StochasticSearch.jl)
- Disponível no **repositório oficial**:  
julia> Pkg.add("StochasticSearch")

- Código sob Licença MIT:  
[github.com/phrb/StochasticSearch.jl](https://github.com/phrb/StochasticSearch.jl)
- Disponível no **repositório oficial**:  
`julia> Pkg.add("StochastichSearch")`
- Testes de Unidade: **97%** de cobertura:  
[coveralls.io/github/phrb/StochasticSearch.jl](https://coveralls.io/github/phrb/StochasticSearch.jl)
- Integração Contínua (**TravisCI**):  
[travis-ci.org/phrb/StochasticSearch.jl](https://travis-ci.org/phrb/StochasticSearch.jl)

# BUSCA ESTOCÁSTICA LOCAL: COMPOSIÇÃO

---

Blocos de construção:

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

# BUSCA ESTOCÁSTICA LOCAL: COMPOSIÇÃO

---

Blocos de construção:

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

## BUSCA ESTOCÁSTICA LOCAL: COMPOSIÇÃO

---

Técnicas de busca usando os blocos:

DONE:

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

# BUSCA ESTOCÁSTICA LOCAL: COMPOSIÇÃO

---

Técnicas de busca usando os blocos:

DONE:

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

TODO:

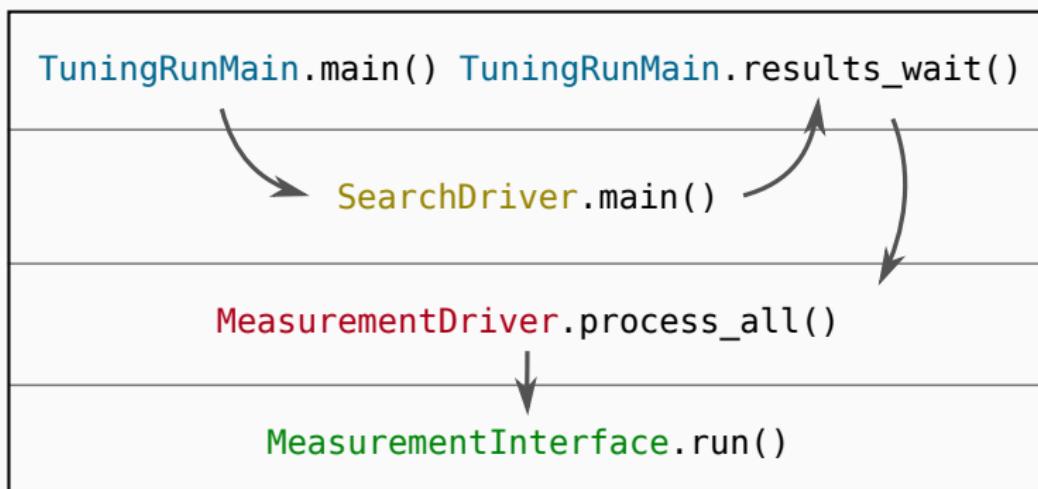
- Randomized Best Improvement
- Iterative Probabilistic Improvement
- Iterated Local Search
- Tabu Search
- Ant Colony Optimization
- Particle Swarm Optimization
- ...

## COMPARAÇÕES COM O OPENTUNER

---

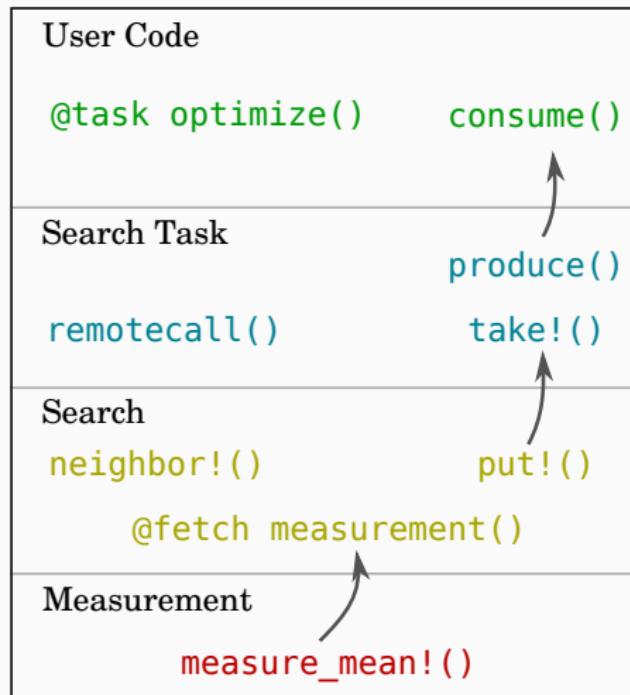
## COMPARAÇÕES COM O OPENTUNER: FLUXO DE EXECUÇÃO

Obtenção de um resultado no OpenTuner, simplificadamente:

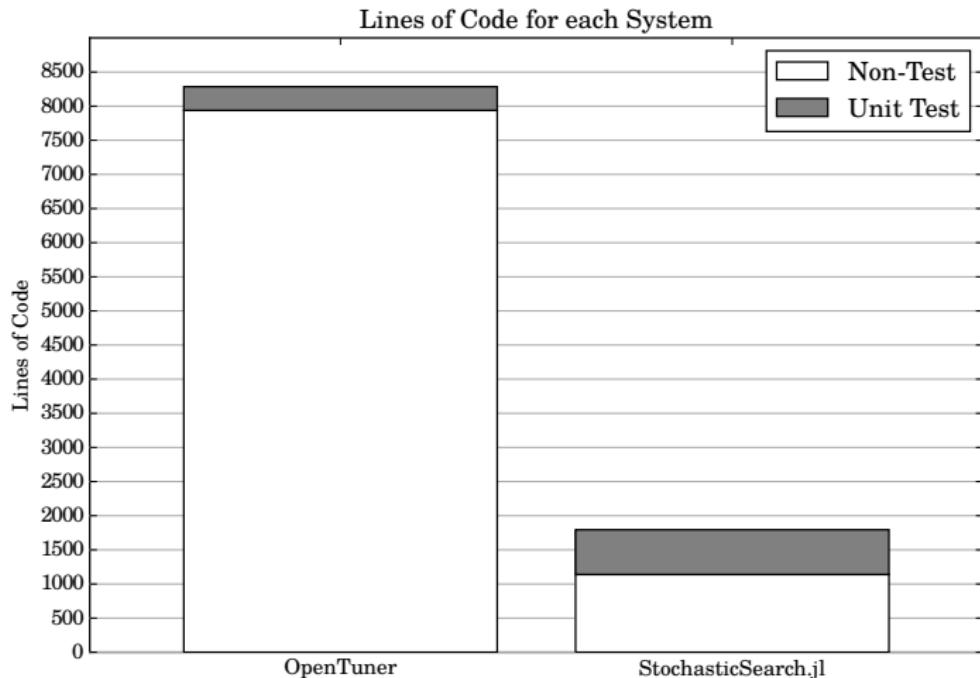


## COMPARAÇÕES COM O OPENTUNER: FLUXO DE EXECUÇÃO

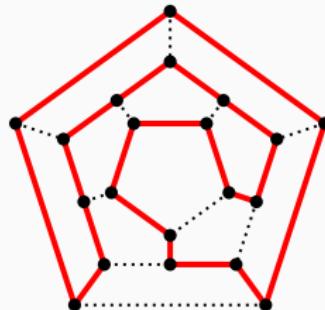
Obtenção de um resultado no StochasticSearch.jl,  
simplificadamente:



# COMPARAÇÕES COM O OPENTUNER: ESFORÇO DE IMPLEMENTAÇÃO



## COMPARAÇÕES: TRAVELLING SALESPERSON PROBLEM

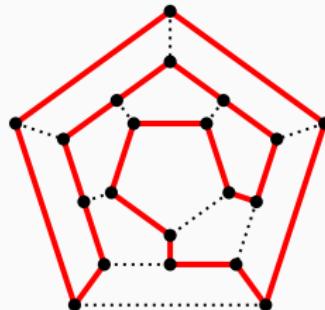


Comparação do desempenho de **autotuners** que resolvem instâncias<sup>1</sup> do TSP, implementados com o OpenTuner e com o StochasticSearch.jl.

---

<sup>1</sup>TSPLIB: <http://goo.gl/8ZZXi2>

## COMPARAÇÕES: TRAVELLING SALESPERSON PROBLEM



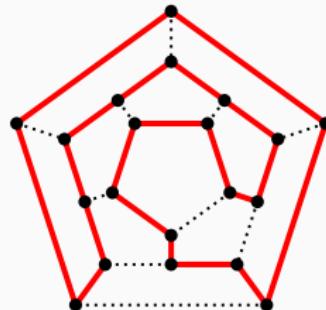
Comparação do desempenho de **autotuners** que resolvem instâncias<sup>1</sup> do TSP, implementados com o OpenTuner e com o StochasticSearch.jl.

As soluções são representadas por **permutações** das cidades.

---

<sup>1</sup>TSPLIB: <http://goo.gl/8ZZXi2>

## COMPARAÇÕES: TRAVELLING SALESPERSON PROBLEM



Comparação do desempenho de **autotuners** que resolvem instâncias<sup>1</sup> do TSP, implementados com o OpenTuner e com o StochasticSearch.jl.

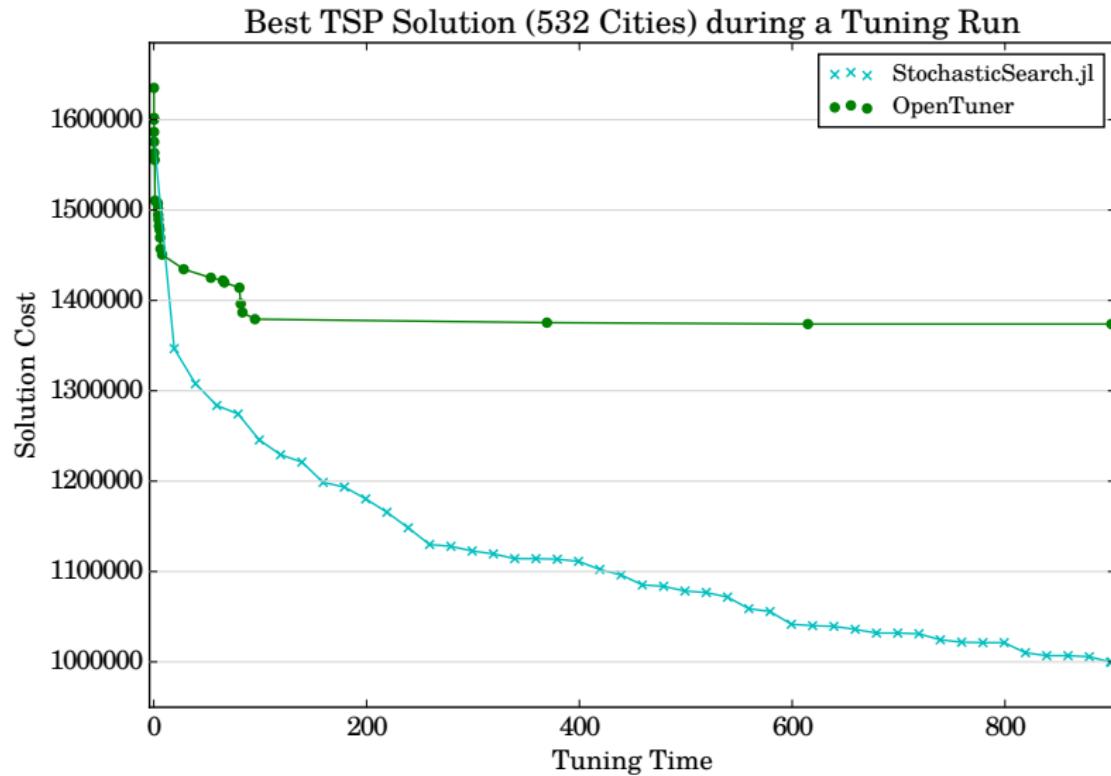
As soluções são representadas por **permutações** das cidades.

Disponível em: [github.com/phrb/stochasticsearch-docs](https://github.com/phrb/stochasticsearch-docs)

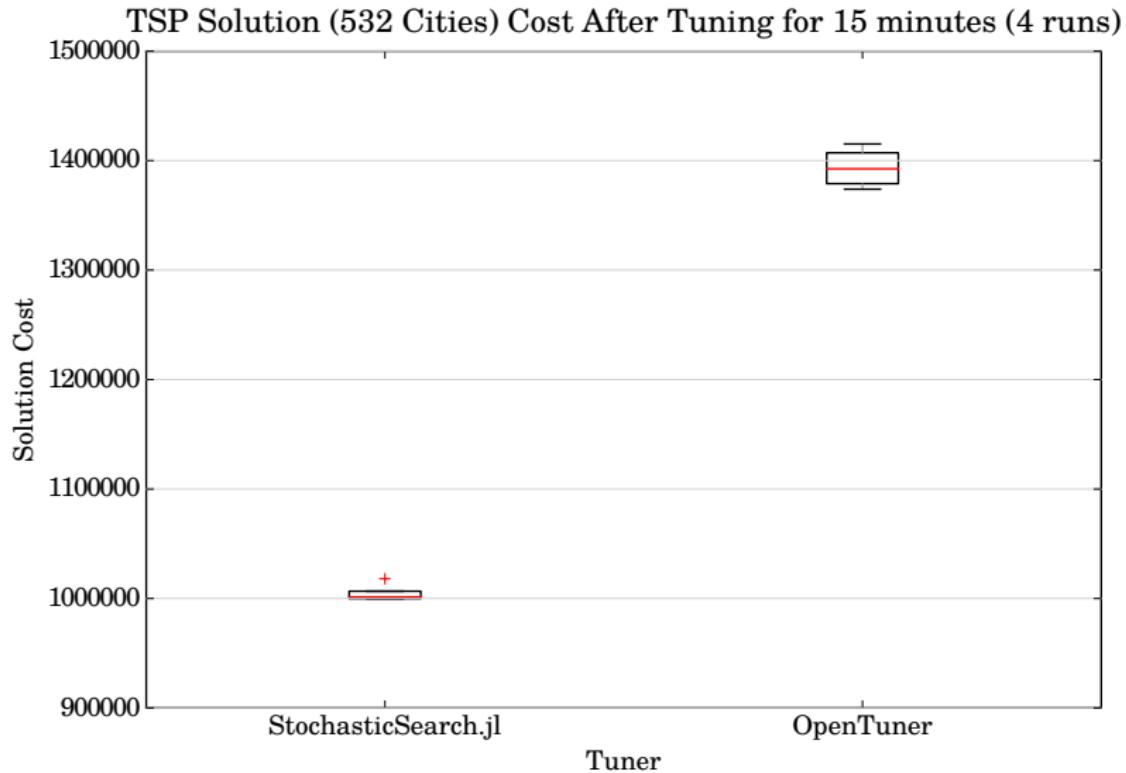
---

<sup>1</sup>TSPLIB: <http://goo.gl/8ZZXi2>

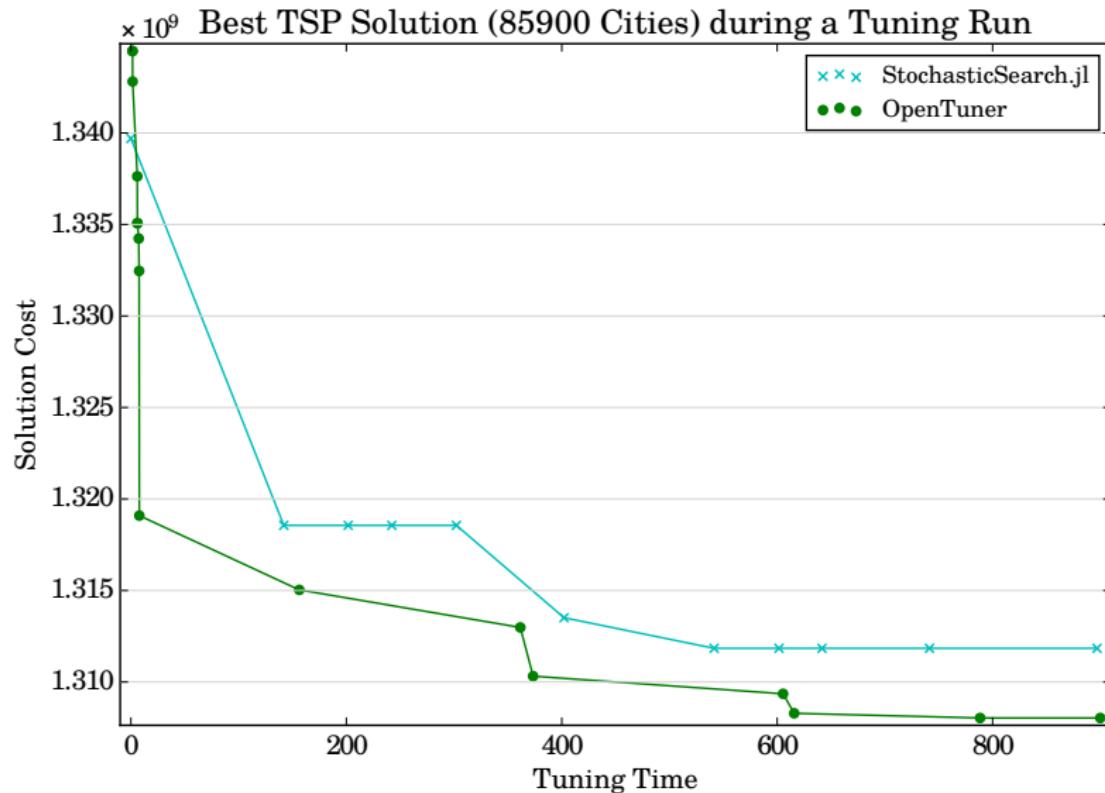
# COMPARAÇÕES: DESEMPENHO TSP (532 CIDADES)



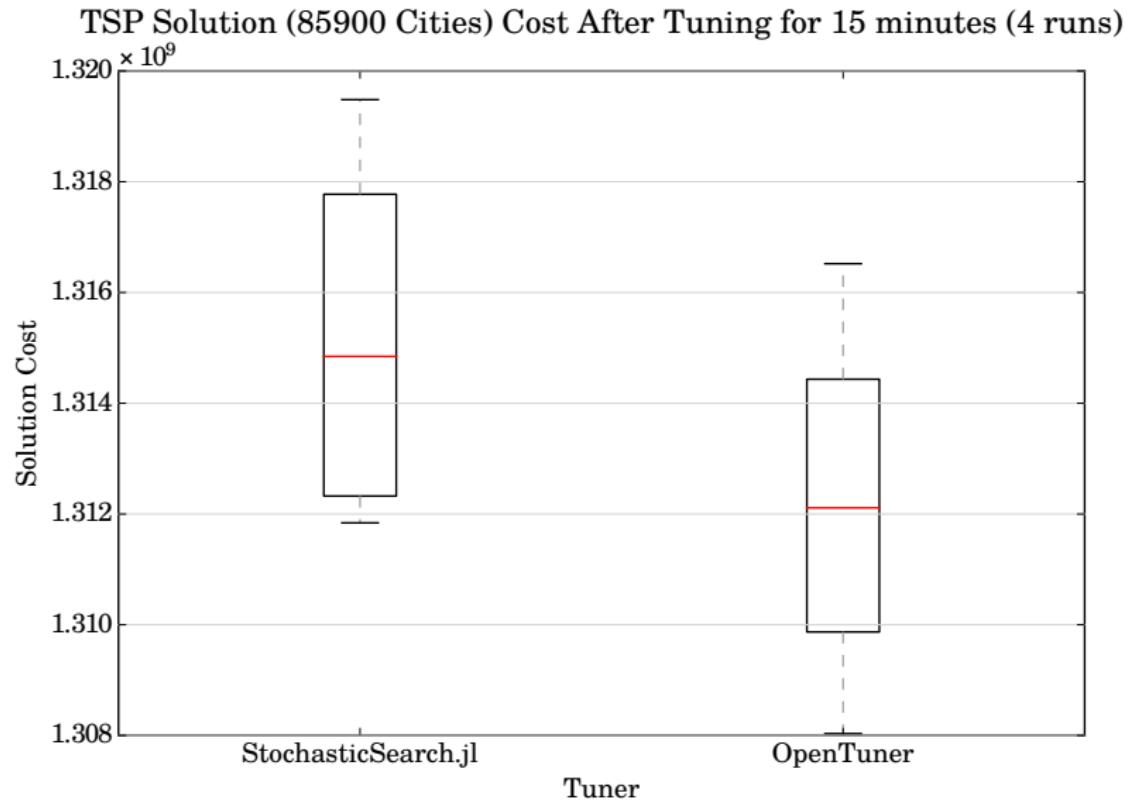
## COMPARAÇÕES: DESEMPENHO TSP (532 CIDADES)



# COMPARAÇÕES: DESEMPENHO TSP (85900 CIDADES)



# COMPARAÇÕES: DESEMPENHO TSP (85900 CIDADES)



## TRABALHOS FUTUROS

---

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no [OpenTuner](#)

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no OpenTuner
- Adicionar, remover e modificar técnicas de busca e parâmetros em tempo de execução

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no OpenTuner
- Adicionar, remover e modificar técnicas de busca e parâmetros em tempo de execução
- Implementar ClusterManagers para GPUs e Nuvem

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no **OpenTuner**
- Adicionar, remover e modificar técnicas de busca e parâmetros em tempo de execução
- Implementar **ClusterManagers** para **GPUs** e **Nuvem**
- Parâmetros de compilação **CUDA**
- Resolvedores **SAT**

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no OpenTuner
- Adicionar, remover e modificar técnicas de busca e parâmetros em tempo de execução
- Implementar ClusterManagers para GPUs e Nuvem
- Parâmetros de compilação CUDA
- Resolvedores SAT
- Experimentos com domínios diferentes
- Experimentos com execução distribuída
- Usar as threads nativas em Julia

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no OpenTuner
- Adicionar, remover e modificar técnicas de busca e parâmetros em tempo de execução
- Implementar ClusterManagers para GPUs e Nuvem
- Parâmetros de compilação CUDA
- Resolvedores SAT
- Experimentos com domínios diferentes
- Experimentos com execução distribuída
- Usar as threads nativas em Julia
- Aumentar a cobertura de testes

## TRABALHOS FUTUROS

---

- Implementar Busca Estocástica Local no OpenTuner
- Adicionar, remover e modificar técnicas de busca e parâmetros em tempo de execução
- Implementar ClusterManagers para GPUs e Nuvem
- Parâmetros de compilação CUDA
- Resolvedores SAT
- Experimentos com domínios diferentes
- Experimentos com execução distribuída
- Usar as threads nativas em Julia
- 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?

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?

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](mailto:phrb@ime.usp.br)

Alfredo Goldman

[gold@ime.usp.br](mailto:gold@ime.usp.br)

29 de Setembro de 2015



Instituto de Matemática e Estatística  
Universidade de São Paulo