DISTRIBUTED OPTIMIZATION SYSTEM

Trabalho Prático – Sistemas Distribuídos e Paralelos

Daniel Kneipp

3 de Julho de 2015

Universidade Federal de Viçosa – *Campus Florestal*

ÍNDICE

MOTIVAÇÃO

MOTIVAÇÃO

- Algumas heurísticas possuem uma natureza probabilística;
- Várias execuções de uma mesma heurística com os mesmos parâmetros de entrada,podem retornar resultados diferentes.
- A ideia é executar instâncias de softwares de otimização paralelamente em um *cluster* e obter o melhor resultado de uma maneira eficiente.
  - Forma de obter o melhor resultado é inspirado em algoritmos de eleição de líderes.

DISTRIBUTED OPTIMIZATION SYSTEM

O que é:

Sistema que gerencia execuções de *softwares* de otimização paralelamente em um ambiente distribuído.

Como foi desenvolvido:

- Utilizou-se a linguagem de programação *JavaScript*;
- API de *Sockets* oferecida pela plataforma *Node.js* [Joy15].

MODELO DE ARQUITETURA

Estrutura geral:

- Cada processo m_k é obrigatoriamente um servidor e é instanciado um objeto cliente para cada servidor que m_k se conecta;
- Todos os processos possuem as mesmas capacidades (*peer-to-peer*).

Estrutura de um processo:

- Essa configuração de 1 servidor e $1 \cdots n$ clientes foi encapsulada no que é chamado de nó.

MODELO DE INTERAÇÃO

Nó:

- Coordenada a interação entre o módulo servidor e o módulo composto pelos clientes de um mesmo processo;
- Inicia operações de envio e tratamento de mensagens recebidas pelos clientes e servidor.

MODELO DE INTERAÇÃO

Cliente:

- Recebe requisições de execução e retorna a solução (não necessariamente a obtida na execução local) para o servidor conectado.

Servidor:

- Envia requisição de execução para os clientes conectados.
- Recebe mensagens de resultado de execução e mensagens que dizem respeito ao algoritmo de eleição de líderes.

MODELO DE FALHA

- Altamente dependente da topologia da rede.
  - Neste caso, se está utilizando uma topologia de rede completa.

DOS suporta:

- Falha de um processo no meio da execução da heurística ou no processo de obtenção do melhor resultado do *cluster*;
- O sistema continua em operação mesmo com apenas um processo funcionando.

MODELO DE SEGURANÇA

- Não possui.
  - Possibilidade para trabalhos futuros.

ALGORITMO DE ELEIÇÃO LÍDERES

- Utilizado para obter o melhor resultado no *cluster*;
- Foi inspirado em [Vil+05] em que se define uma rede virtual que tem a topologia de anel.

Restrição:

Este algoritmo exige que se tenha uma rede onde todos os processos estejam conectados à todos os outros.

ALGORITMO DE ELEIÇÃO LÍDERES

Passos:

1. A partir de um gatilho, o processo m_r envia para todo processo m_k seu processo seguinte m_{k+1} na rede, com $1 \leq k \leq n$;
2. m_r envia uma *flag* para m_1 fazendo com que este envie seu *id* (resultado obtido na execução da heurística) para m_2 sem esperar qualquer *id*;
3. m_k envia para m_{k+1} o melhor *id* (menor, se o problema de otimização for de minimização) entre os *ids* de m_k e m_{k-1};
4. Quando $m_k = m_r$, m_r apresenta o melhor resultado entre m_r e m_{k-1}.

TESTES E RESULTADOS

Minimizar a função conhecida como Rastrigin [Pro15] modificada por [Hed15].

Função objetivo:

$$f(\bar{x}) = 0.2 + x_1^2 + x_2^2 - 0.1cos(6.0\pi x_1) - 0.1cos(6.0\pi x_2), \quad (1)$$
$$\bar{x} = \{x_1, x_2\}, \quad x_i = \{k \in \mathbb{R} : -5 \leq k \leq 5\}$$

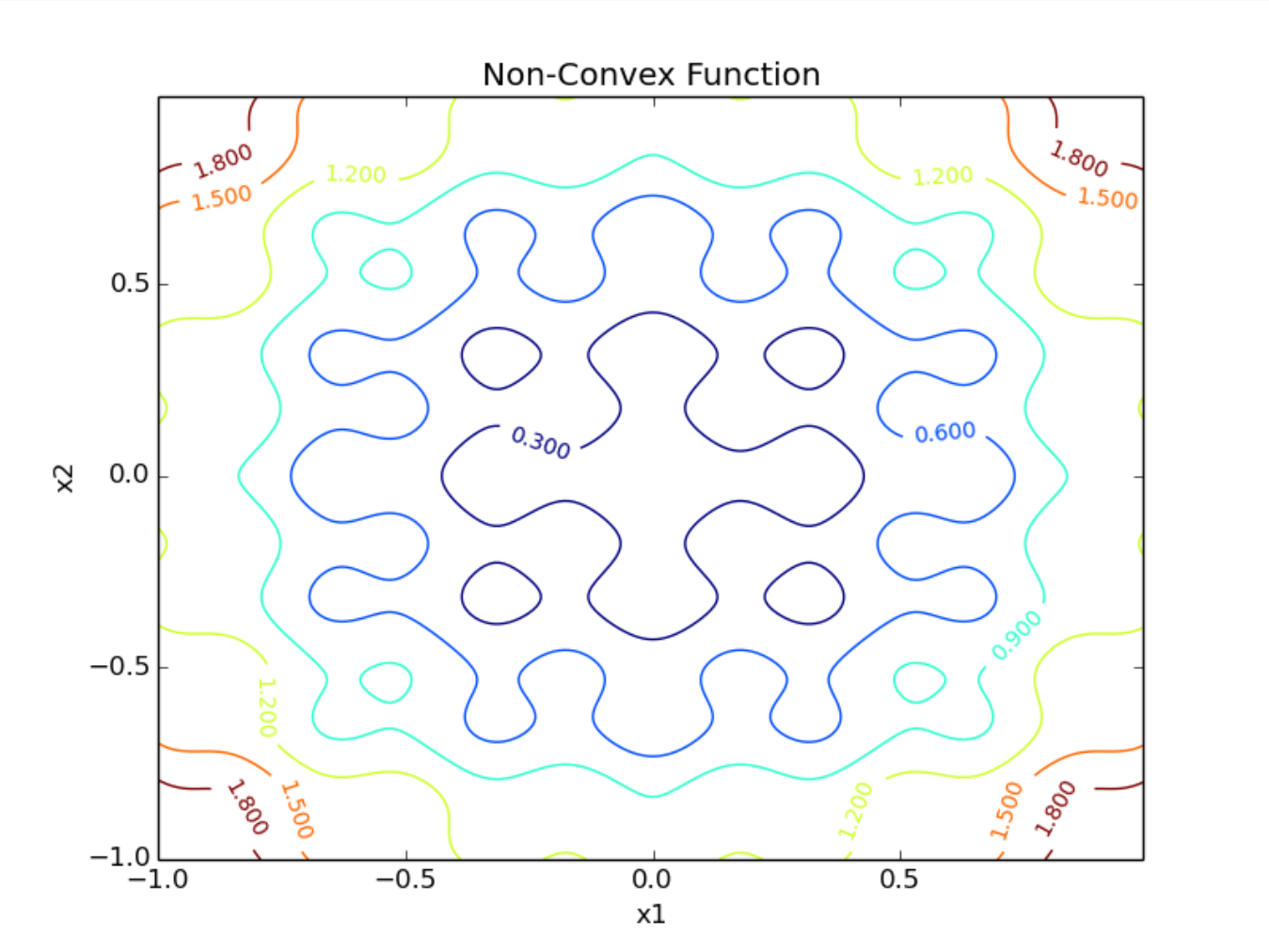

Figura 1: Representação gráfica da equação 1.

HEURÍSTICA

- Algoritmo de otimização baseado na meta-heurística [Net15] *Simulated Annealing*;
- Apenas uma solução é utilizada em todo o processo de otimização;
- Aceita piora da função objetivo (para sair de mínimos locais);
- Código em *Python* obtido a partir de [Hed15].

TESTES

- Executou-se 11 vezes o processo de otimização tanto no SA no modo autônomo quanto utilizando o DOS;
- Utilizou-se uma máquina com 4 núcleos físicos de processamento;
- Para o DOS, foram instanciados 4 processos na mesma máquina e foram criadas interfaces de rede virtuais para conecta-los.

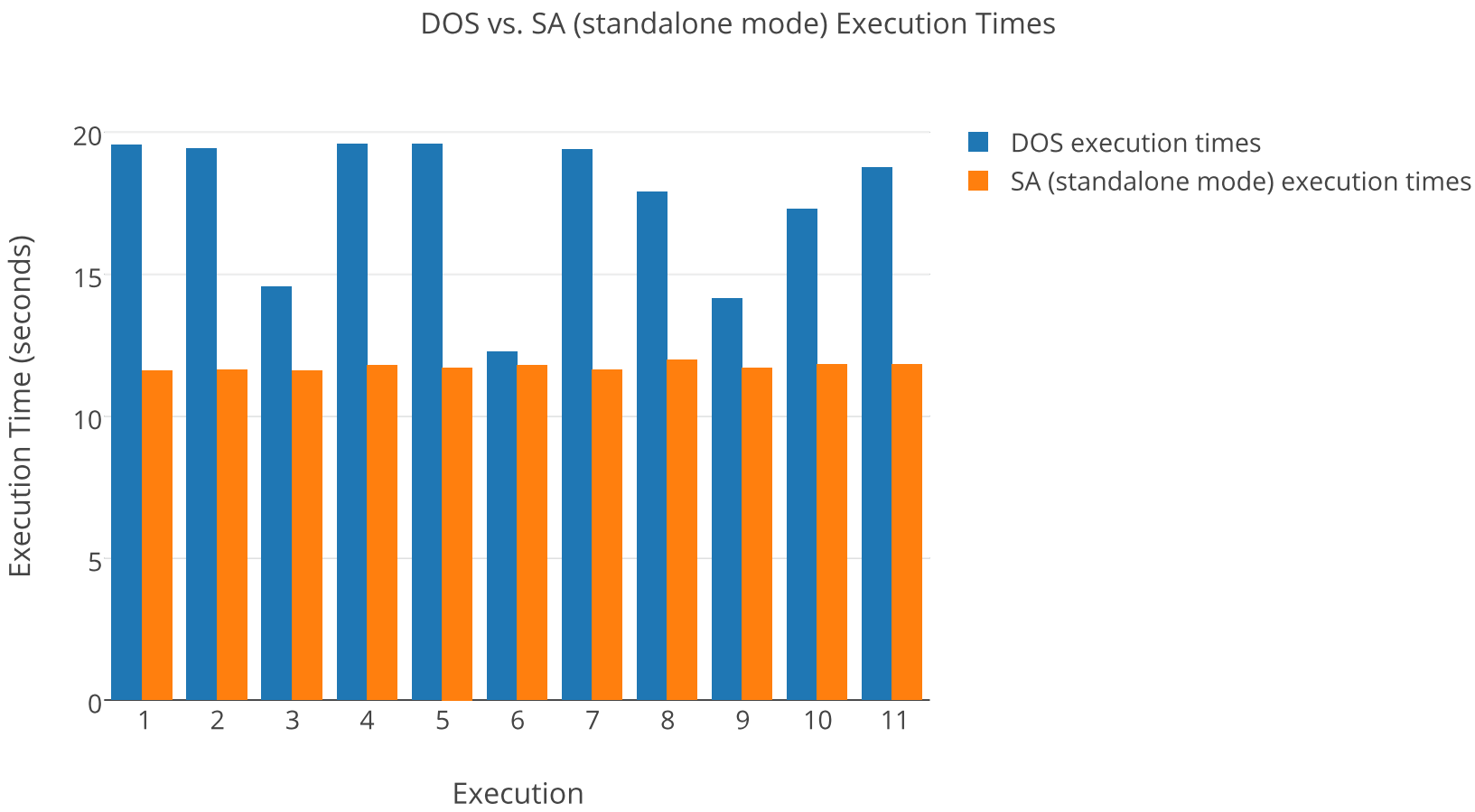

Figura 2: Comparativo de tempos de execução.

Tabela 1: Comparativo de tempos de execução (em segundos).

	SA	DOS
Média	11.7332685427	17.4981684871
Desvio Padrão	0.115128069478	2.5090479674

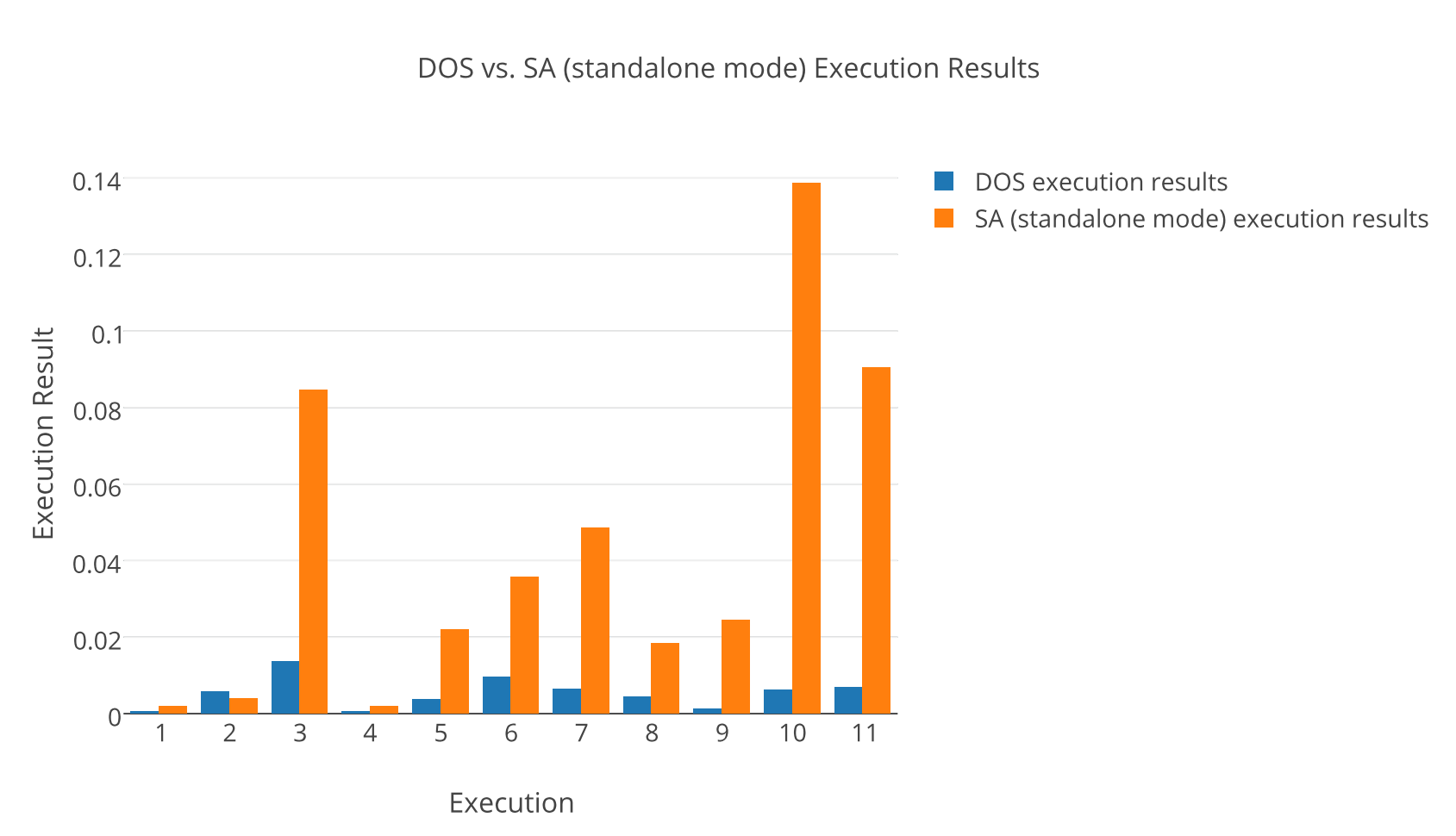

Figura 3: Comparativo de resultados obtidos.

Tabela 2: Comparativo de resultados obtidos.

	SA	DOS
Média	0.0427484902005	0.00538080908429
Desvio Padrão	0.042122092619	0.00378422466426

Demonstração

CONCLUSÃO

CONCLUSÃO

- Utilização do DOS traz excelentes resultados dependendo do problema;
  - É mais provável de se ter resultados melhores com várias execuções de uma heurística que possui natureza probabilística.
- Mas consome muito processamento e energia já que se executa várias instancias da heurística paralelamente e não apenas uma.

Perguntas?

REFERÊNCIAS

[Hed15] John D. Hedengren. *Simulated Annealing Tutorial*. 20 de jun. de 2015. URL: http://apmonitor.com/me575/index.php/Main/SimulatedAnnealing.

[Joy15] Joyent, Inc. *node.js*. 15 de abr. de 2015. URL: https://nodejs.org/.

[Net15] Metaheuristics Network. *Metaheuristics Network web site*. 11 de abr. de 2015. URL: http://www.metaheuristics.net/index.php?main=1.

[Pro15] Go Test Problems. *TEST FUNCTIONS: Rastrigin Function*. 20 de jun. de 2015. URL: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2607.htm.

REFERÊNCIAS

[Vil+05] J. Villadangos et al. “Efficient leader election in complete networks”. Em: *Parallel, Distributed and Network-Based Processing, 2005. PDP 2005. 13th Euromicro Conference on*. Fev. de 2005, pp. 136–143. DOI: 10.1109/EMPDP.2005.21.