METAHEURÍSTICA GRASP COM REFINAMENTO POR BUSCA LOCAL PARA O FLOWSHOP PERMUTACIONAL

Alberto F. K. Neto

Otimização Combinatória — INF05010 — 2019/2

AGENDA

1. Introdução
2. Definição do Problema
3. Método heurístico de resolução
4. Experimentos computacionais
5. Conclusões

Flowshop permutacional

- Tópico de pesquisa recorrente em Otimização Combinatória
- Grande interesse acadêmico e aplicado
- Fácil obter soluções factíveis
- Difícil de provar uma solução ótima

Sobre o trabalho desenvolvido

- Formulação inteira mista de Tseng et al. (2004)
- Solução construtiva com GRASP
- Melhoramento por Busca Local
- Comparação de desempenho

Dados do problema

- N tarefas
- M máquinas
- $T_{ri} \geqslant 0$ tempo de processamento $(1 \leqslant i \leqslant N \ ; \ 1 \leqslant r \leqslant M)$

Objetivo: Minimizar tempo de processamento final da máquina M

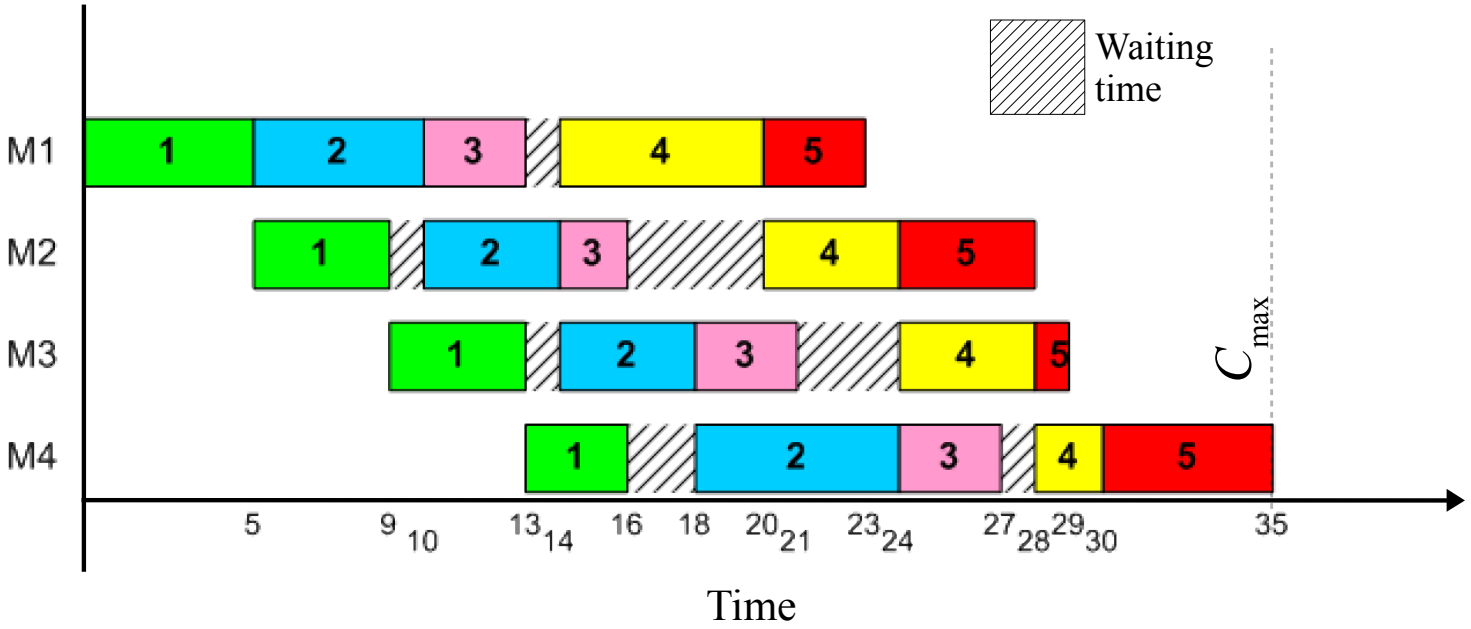
Solução: Ordem de execução das tarefas

Restrições

- Mesma ordem em todas as máquinas
- Processamento completo na máquina anterior antes de prosseguir

Exemplo de instância

- $N = 5$
- $M = 4$
- Solução de custo $z = 35$



Variáveis de decisão

- $D_{ik} \in \{0, 1\}$: Indica se a tarefa i é processada antes da tarefa k, para $1 \leqslant i < k \leqslant N$
- $C_{ri} \geqslant 0$: Tempo em que a tarefa i termina de ser processada na máquina r, para $1 \leqslant i \leqslant N$ e $1 \leqslant r \leqslant M$
- $C_{\max}$: Tempo de processamento final da última máquina

Parâmetros

- $T_{ri} \geqslant 0$: Tempos de processamento
- P: Valor suficientemente grande

$$\text{Minimize } C_{\max} \tag{1}$$

Sujeito a:

$$C_{1i} \geqslant T_{1i} \qquad 1 \leqslant i \leqslant N \tag{2}$$

$$C_{ri} - C_{r-1,i} \geqslant T_{ri} \qquad 2 \leqslant r \leqslant M, 1 \leqslant i \leqslant N \tag{3}$$

$$C_{ri} - C_{rk} + PD_{ik} \geqslant T_{ri} \qquad 1 \leqslant r \leqslant M, 1 \leqslant i < k \leqslant N \tag{4}$$

$$C_{ri} - C_{rk} + PD_{ik} \leqslant P - T_{rk} \qquad 1 \leqslant r \leqslant M, 1 \leqslant i < k \leqslant N \tag{5}$$

$$C_{\max} \geqslant C_{Mi} \qquad 1 \leqslant i \leqslant N \tag{6}$$

$$C_{ri} \geqslant 0 \qquad 1 \leqslant r \leqslant M, 1 \leqslant i \leqslant N \tag{7}$$

$$D_{ik} \in \{0, 1\} \qquad 1 \leqslant i < k \leqslant N \tag{8}$$

Sobre o GRASP

- Proposto por Feo et al. (1994)
- Método construtivo guloso randomizado
- Parâmetro de randomização $\alpha \in [0, 1]$

Detalhes de implementação

- Lista com ordem de execução
- Tempos em estrutura de dados 2D
- Randomização controlada com sementes

Algorithm 1: Construção de solução inicial com GRASP.

```
1  Procedure GRASP(N, M, T, α)
2      pend ← lista com valores 1, 2, ..., N
3      s ← lista vazia; z ← 0
4      while pend não está vazia do
5          RCL ← lista vazia
6          for j ∈ pend do
7              z̄_j ← custo da solução parcial s com adição da tarefa j
8              adicione a tupla (j, z̄_j) em RCL
9          ordene RCL em ordem não crescente de z̄
10         tam ← tamanho da lista RCL
11         tp ← escolhe aleatoriamente um índice de [1, max{1, α · tam}]
12         atualize a solução s e custo z com os dados da tupla RCL_tp
13         remova a tarefa referente a tp de pend
14 return s
```

Estratégia randomizada

- Troca de duas tarefas aleatórias
- Sempre aceita uma melhora
- Busca local rápida e iterada

Algorithm 2: Algoritmo de Busca Local.

```
1  Procedure Swap2LS(s*, numVezes)
2      z* ← custo da solução atual
3      for i ← 1 até numVezes do
4          selecione tarefas j1 ≠ j2 aleatoriamente, com distribuição uniforme
5          s̄ ← troque a ordem de processamento de j1 ↔ j2 em s*
6          z̄ ← avalie o custo da solução s̄
7          if z̄ < z* then
8              s* ← s̄
9              z* ← z̄
10 return s*
```

Algorithm 3: Heurística GRASP com Busca Local.

```
1 Procedure GRASP_LS(N, M, T, α)
2     s ← GRASP(N, M, T, α)
3     for iter ← 1 até MAX_ITER do
4         Swap2LS(s, ⌈N/100⌉)
5         Swap2LS(s, ⌈iter/1000⌉)
6         Swap2LS(s, randomInt(1,N))
7 return s
```

Hardware e software

- Intel 3612QM @ 2.10GHz, RAM 8GB
- GLPK 4.65
- Heurística em Python 3.7.4
- Arch Linux (kernel linux-5.3.8)

Experimentos realizados

- Solver por até 1h
- Heurística por até 2140 iterações
- $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
- 10 replicações por (instância,α)

Instância	BKS	Valor relaxação	Obj. solução inteira	GAP_{BKS} (%)
VFR10_15_1	1307	880,0	1307[1]	0,0
VFR10_10_3	1592	687,0	1873	56,9
VFR_20_20_1	2270	1391,0	2573	42,6
VFR60_5_10	3663	382,0	3878	89,3
VFR100_60_1	9395	TL	–	∞
VFR500_40_1	28548	TL	–	∞
VFR500_60_3	31125	TL	–	∞
VFR600_20_1	31433	TL	–	∞
VFR700_20_10	36417	TL	–	∞

Figura: Resultado obtido por meio do GLPK.

[1] Após 1244,7 segundos de processamento.

Instância	BKS	Sol. GRASP		Sol. GRASP+BL		
		F.O.	Desvio (%)	F.O.	Desvio (%)	Tempo (seg.)
VFR10_15_1	1307	1424 ± 0	8,95	$1339,6 \pm 18,319$	2,49	$1,5$
VFR20_10_3	1592	2017 ± 0	26,70	$1687,5 \pm 29,304$	6	$2,1$
VFR20_20_1	2270	2715 ± 0	19,60	$2360,1 \pm 33,478$	3,97	$3,9$
VFR60_5_10	3663	3849 ± 0	5,08	$3668,4 \pm 7,291$	0,15	$3,2$
VFR60_10_3	3423	4357 ± 0	27,29	$3632,6 \pm 62,45$	6,12	$6,0$
VFR100_60_1	9395	11247 ± 0	19,71	$10008,8 \pm 47,123$	6,53	$57,7$
VFR500_40_1	28548	33119 ± 0	16,01	$30640,6 \pm 67,832$	7,33	$200,4$
VFR500_60_3	31125	36930 ± 0	18,65	$33539,6 \pm 106,966$	7,76	$298,5$
VFR600_20_1	31433	35473 ± 0	12,85	$32904,4 \pm 69,306$	4,68	$118,4$
VFR700_20_10	36417	40916 ± 0	12,35	$37857,4 \pm 114,996$	3,96	$140,6$

Figura: Resultados médios da heurística para $\boldsymbol{\alpha = 0}$.

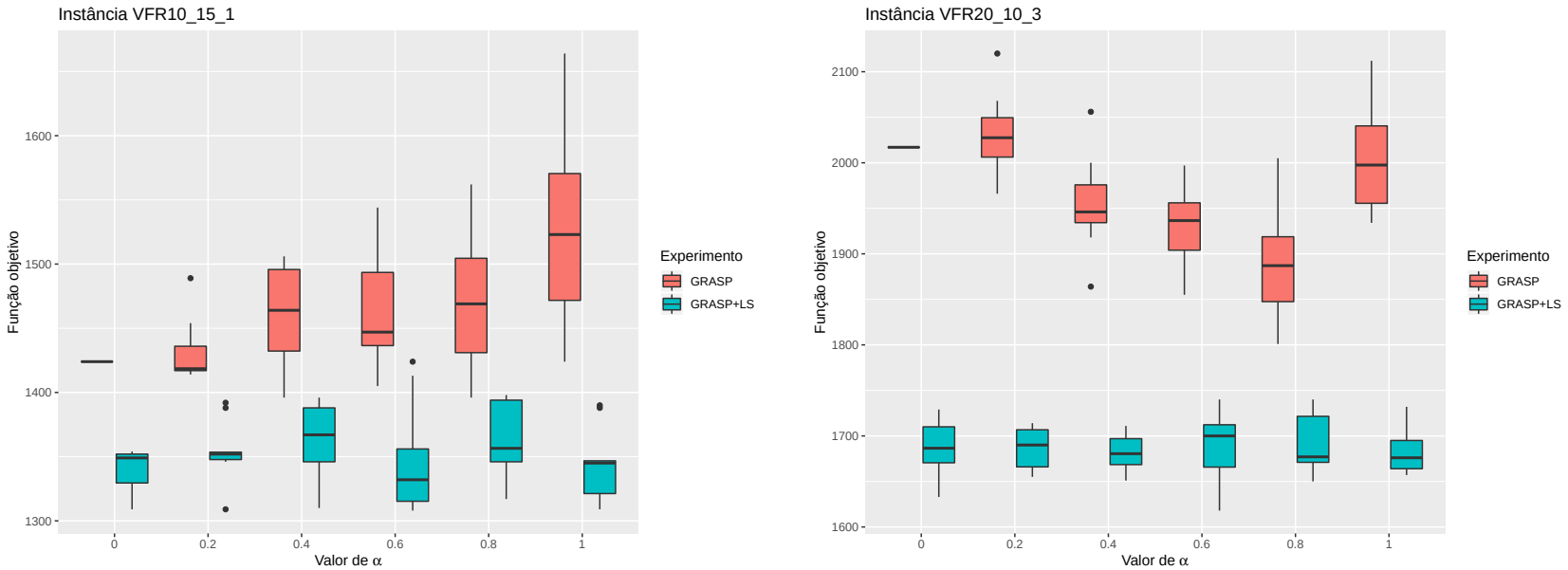


Figura: Boxplot relacionando valor médio da função objetivo para as diversas instâncias de testes, com vários valores α e 10 replicações por caso de teste.

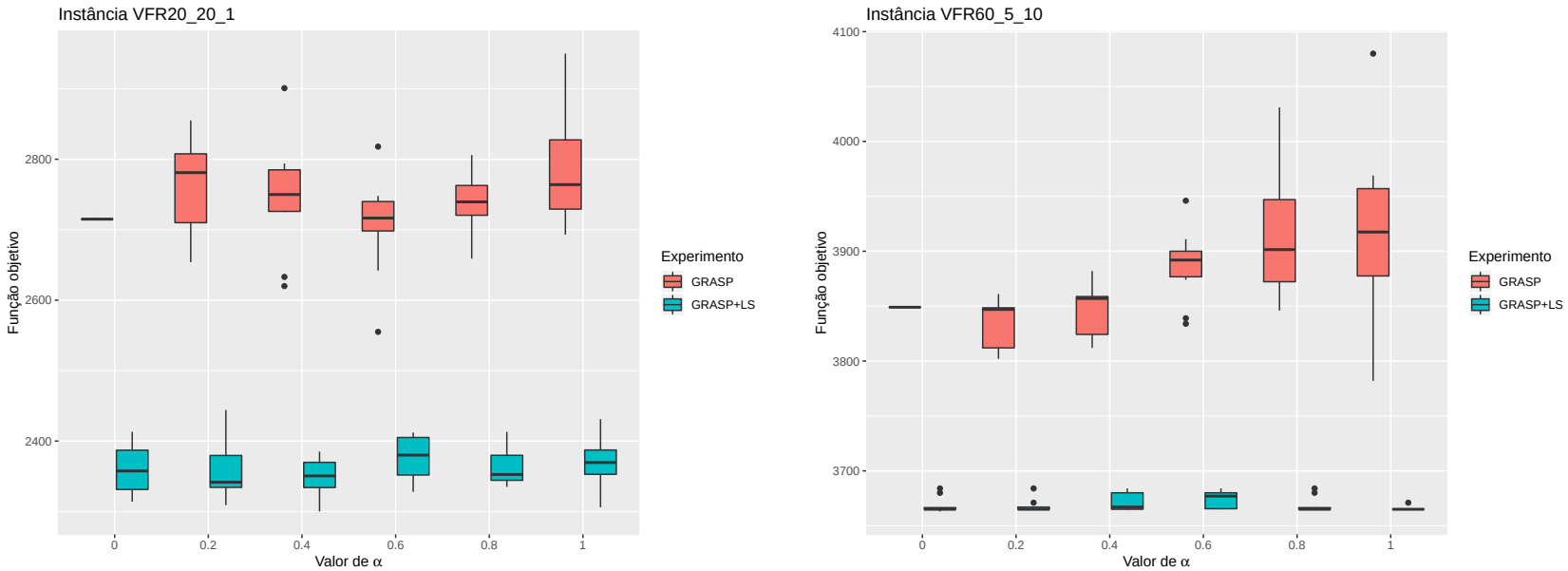


Figura: Boxplot relacionando valor médio da função objetivo para as diversas instâncias de testes, com vários valores α e 10 replicações por caso de teste.

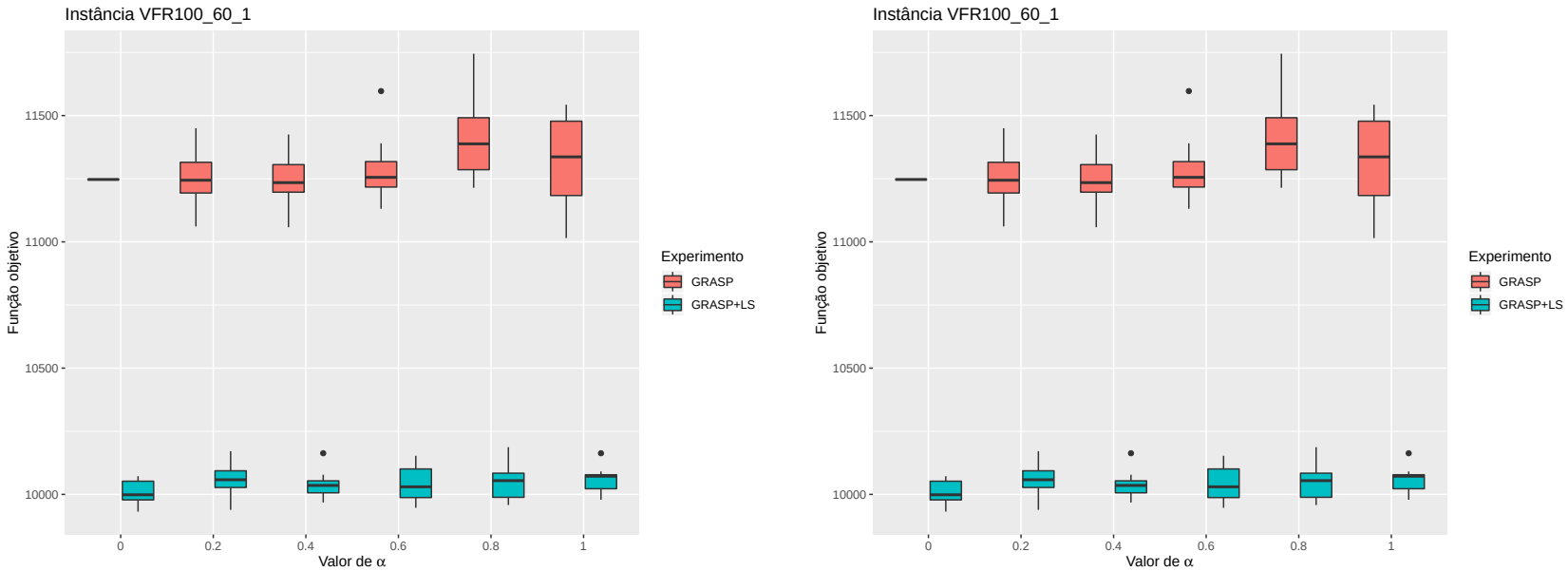


Figura: Boxplot relacionando valor médio da função objetivo para as diversas instâncias de testes, com vários valores α e 10 replicações por caso de teste.

EFEITO DA BUSCA LOCAL

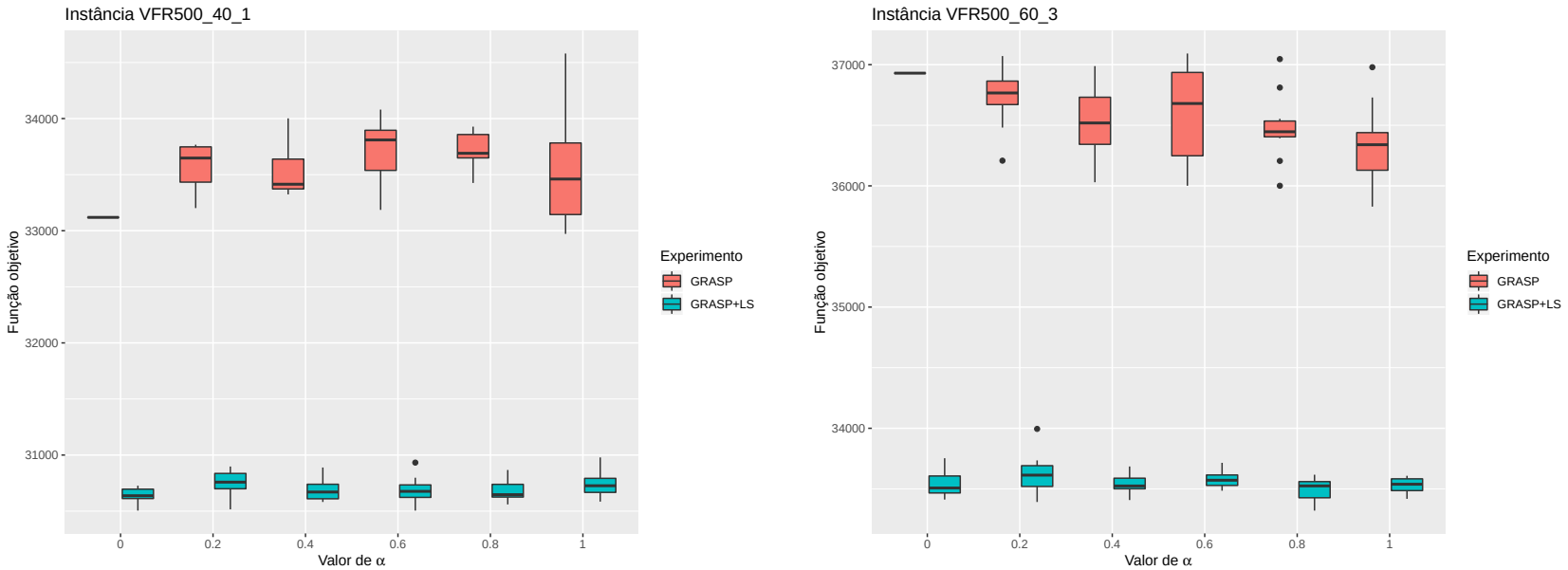


Figura: Boxplot relacionando valor médio da função objetivo para as diversas instâncias de testes, com vários valores α e 10 replicações por caso de teste.

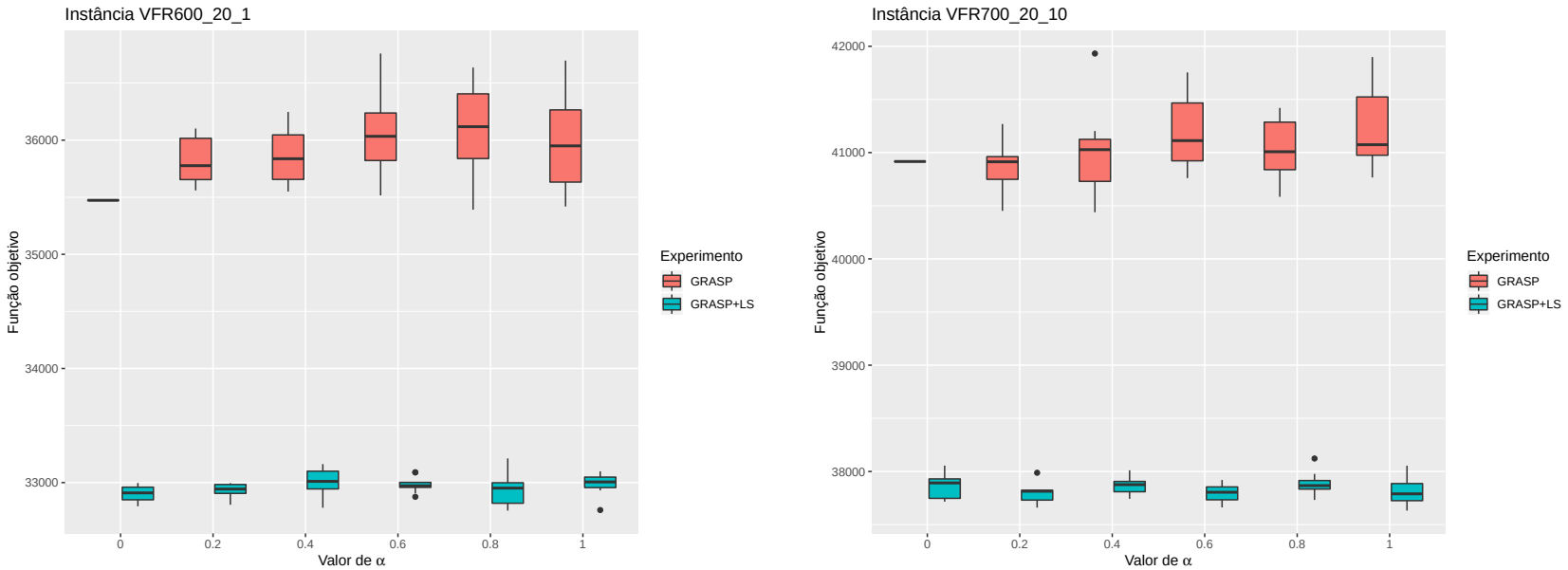


Figura: Boxplot relacionando valor médio da função objetivo para as diversas instâncias de testes, com vários valores α e 10 replicações por caso de teste.

PFSP é um problema relevante

- Abordagem exata é ineficiente
- GRASP obtém soluções iniciais rapidamente
- Randomização pouco efetiva
- Busca Local fez diferença
- Heurística foi eficaz e eficiente

Trabalhos futuros

- Cálculo mais eficiente de custo da vizinhança com `Swap2LS`
- Calibração dos parâmetros

Thomas A Feo, Mauricio GC Resende, and Stuart H Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.

Fan T Tseng, Edward F Stafford Jr, and Jatinder ND Gupta. An empirical analysis of integer programming formulations for the permutation flowshop. *Omega*, 32(4):285–293, 2004.

Obrigado!

Alberto F. K. Neto

Institute of Informatics (II)
Federal University of Rio Grande do Sul (UFRGS)
afkneto@inf.ufrgs.br