Prof. Lucas de Souza Batista - DEE/EE/UFMG

Otimização de Redes

Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP

- Proposto por T.A. Feo e M.G.C. Resende (1989):
  - *Probabilistic algorithm (GRASP) for difficult set covering problems*

- Algumas ideias do GRASP foram relatadas em trabalhos anteriores:
  - random multistart local search (Lin & Kernighan, 1973)
  - semi-greedy heuristics (Hart & Shogan, 1987)

GRASP

- Representa uma metaheurística multistart para problemas de otimização combinatória;

- Cada iteração consiste de duas etapas:
  - heurística construtiva;
  - busca local.

GRASP

- Na etapa construtiva,
  - constrói-se uma solução viável;
  - um mecanismo de reparo pode ser necessário.

- Na etapa de busca local,
  - a vizinhança da solução corrente é investigada até encontrar um ótimo local.

Pseudocódigo do GRASP

```
Algoritmo: GRASP (Max_Iterations, Seed)
    Read_Input();
    for k = 1,..., Max_Iterations do
        Solution ← Greedy_Randomized_Construction(Seed);
        if Solution is not feasible then
            Solution ← Repair(Solution);
        end
        Solution ← Local_Search(Solution);
        Update_Solution(Solution, Best_Solution);
    end
    return Best_Solution;
end GRASP
```

Fase Construtiva do GRASP

- Na etapa construtiva,
  - defini-se o conjunto de elementos candidatos que podem ser incorporados na solução parcial;
  - todos esses elementos são avaliados de acordo com uma função gulosa (*greedy function*);
  - uma parcela dos melhores elementos compõem uma RCL (*restricted candidate list*);
  - o elemento adicionado à solução parcial é selecionado aleatoriamente desta lista;
  - esse processo é repetido até construir a solução completa.

Fase Construtiva do GRASP

```
Algoritmo: Greedy_Randomized_Construction(Seed)
    Solution ← ∅;
    Initialize the set of candidate elements;
    Evaluate the incremental costs of the candidate elements;
    while there exists at least one candidate element do
        Build the restricted candidate list (RCL);
        Select an element s from the RCL at random;
        Solution ← Solution ∪ {s};
        Update the set of candidate elements;
        Reevaluate the incremental costs;
    end
    return Solution;
end Greedy_Randomized_Construction
```

Fase de Refinamento do GRASP

- A fase construtiva não garante um ótimo local!!!
- Na etapa de refinamento,
  - a solução construída é melhorada iterativamente até um ótimo local;
  - a vizinhança pode ser explorada usando *best improvement* ou *first improvement*;
  - muitas aplicações mostram que as duas técnicas conduzem à mesma solução final;
  - entretanto, *first improvement* requer um menor custo.

Fase de Refinamento do GRASP

***Algoritmo**: Local_Search (Solution)*
    while** Solution in not locally optimal **do
        ***Find** $s' \in N(Solution)$ **with** $f(s') < f(Solution)$;*
        Solution $\leftarrow s'$;
    end
    ***return** Solution;*
***end** Local_Search*

Construção da Lista Restrita de Candidatos

- Assuma $c(e)$ o custo incremental associado à inclusão do elemento e à solução parcial;
- Assuma c^{min} e c^{max} o menor e o maior custos incrementais, respectivamente;
- A lista RCL é composta pelos melhores elementos, i.e., $c(e) \in [c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$;
- $\alpha = 0$: construção puramente gulosa;
- $\alpha = 1$: construção puramente aleatória.

Construção da Lista Restrita de Candidatos

***Algoritmo**: Greedy_Randomized_Construction (α, Seed)*

Solution ← ∅;

Initialize the candidate set: C ← E;

Evaluate the incremental cost c(e) for all e ∈ C;

while** C ≠ ∅ **do

$c^{min} \leftarrow min\{c(e) \mid e \in C\}$;

$c^{max} \leftarrow max\{c(e) \mid e \in C\}$;

$RCL \leftarrow \{e \in C \mid c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\}$;

Select an element s from the RCL at random;

Solution ← Solution ∪ {s};

Update the candidate set C;

Reevaluate the incremental cost c(e) for all e ∈ C;

end

***return** Solution;*

***end** Greedy_Randomized_Construction*

Construção da Lista Restrita de Candidatos

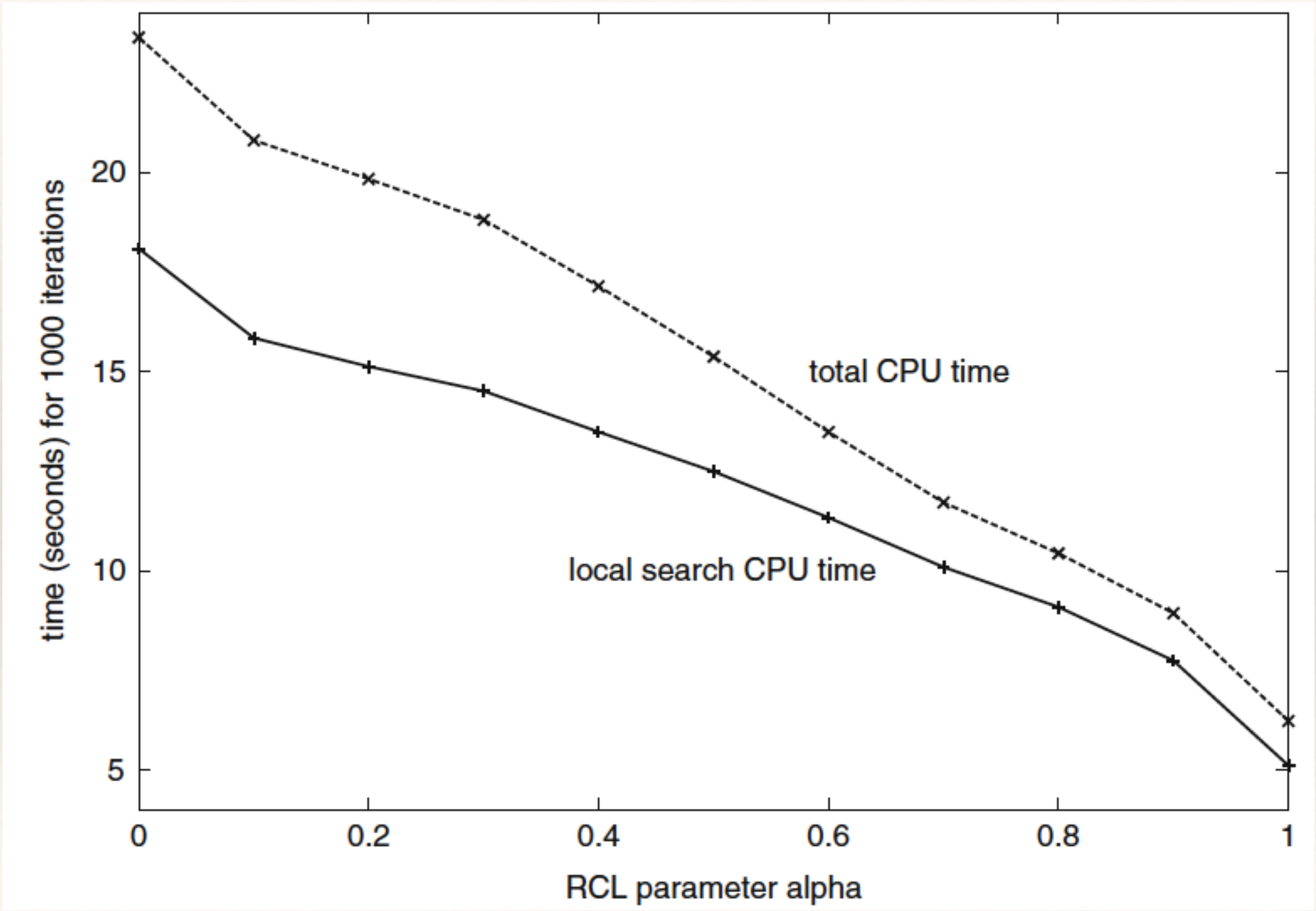
- Frequentemente, usa-se $\alpha = 0.2$:
  - produz soluções com custos relativamente baixos;
  - produz uma variedade significativa de soluções;
  - o tempo economizado na busca local a partir de uma solução inicial *boa,* pode ser usado para a realização de mais iterações do GRASP.

Construção da Lista Restrita de Candidatos

α	Average distance	Average moves	Local search time (s)	Total time (s)
0.0	12.487	12.373	18.083	23.378
0.1	10.787	10.709	15.842	20.801
0.2	10.242	10.166	15.127	19.830
0.3	9.777	9.721	14.511	18.806
0.4	9.003	8.957	13.489	17.139
0.5	8.241	8.189	12.494	15.375
0.6	7.389	7.341	11.338	13.482
0.7	6.452	6.436	10.098	11.720
0.8	5.667	5.643	9.094	10.441
0.9	4.697	4.691	7.753	8.941
1.0	2.733	2.733	5.118	6.235

- $\alpha = 0$: construção puramente aleatória;
- $\alpha = 1$: construção puramente gulosa.

Construção da Lista Restrita de Candidatos



Construção da Lista Restrita de Candidatos

- Diferentes estratégias para definição de α foram propostas:
  - (R) auto-adaptação de α;
  - (E) escolha aleatória de α a partir de uma *pdf* uniforme;
  - (H) escolha aleatória de α a partir de uma *pdf* não-uniforme decrescente;
  - (F) valor de α fixo.

Construção da Lista Restrita de Candidatos

		R		*E*		*H*		*F*	
Problem	Instances	Hits	Time	Hits	Time	Hits	Time	Hits	Time
P-1	36	34	579.0	35	358.2	32	612.6	24	642.8
P-2	7	7	1346.8	6	1352.0	6	668.2	5	500.7
P-3	44	22	2463.7	23	2492.6	16	1740.9	11	1625.2
P-4	37	28	6363.1	21	7292.9	24	6326.5	19	5972.0
Total	124	91		85		78		59	

Construção da Lista Restrita de Candidatos

- Estratégia (R) encontrou com maior frequência os melhores resultados, mas ao custo de maiores tempos computacionais;
- Estratégia (F) apresentou os menores tempos computacionais, mas raramente encontrou a melhor solução;
- Estratégia (E) ilustra a efetividade da variação do parâmetro α, mesmo que de forma aleatória.

Mecanismos Alternativos de Construção

- *Random Plus Greedy* (RPG)
  - Ao invés de combinar *determinismo* e *aleatoriedade* em cada passo da construção, o mecanismo RPG aplica aleatoriedade durante as primeiras p fases de construção;
- O algoritmo completa a solução com uma ou mais fases gulosas de construção.
- O valor p permite controlar o balanço entre a aleatoriedade e o determinismo da construção.

Mecanismos Alternativos de Construção

- *Reactive GRASP*
  - O parâmetro α é selecionado aleatoriamente a partir de uma lista discreta a cada iteração;
  - Suponha uma lista de valores $\Psi = \{\alpha_1, \ldots, \alpha_m\}$, em que a probabilidade inicial de escolha de α_i seja $p_i = 1/m$;
  - Essas probabilidades de seleção são reavaliadas periodicamente, $p_i = q_i / \Sigma_j q_j$, com $q_i = z^*/A_i$;
  - z^* é a solução incumbente e A_i é o valor médio de todas as soluções encontradas usando α_i.

Mecanismos Alternativos de Construção

- Inúmeras outras técnicas:
  - *Sampled Greedy Construction*
  - *Cost Perturbations*
  - *Bias Functions*
  - *Intelligent Construction: Memory and Learning*
  - *Proximate Optimality Principles*

Path-Relinking

- Proposto por Glover (1996) no contexto de Tabu Search;
- Tende a melhorar tanto a qualidade da solução quanto os tempos de processamento;
- Path-relinking é usualmente aplicado entre duas soluções: *solução inicial (s)* e *solução guia (g);*
- Um ou mais caminhos que conectam estas soluções são explorados à procura de melhores soluções;
- Uma busca local é aplicada à melhor solução encontrada em cada caminho.

Path-Relinking

Procedimento Geral:

1. Determina a solução inicial *s* e a solução guia *g*;
2. Determina o conjunto de componentes $\Delta(s,g)$ em que *s* e *g* se diferem;
3. Este conjunto corresponde aos movimentos que precisam ser aplicados a *s* para obter *g*;

Path-Relinking

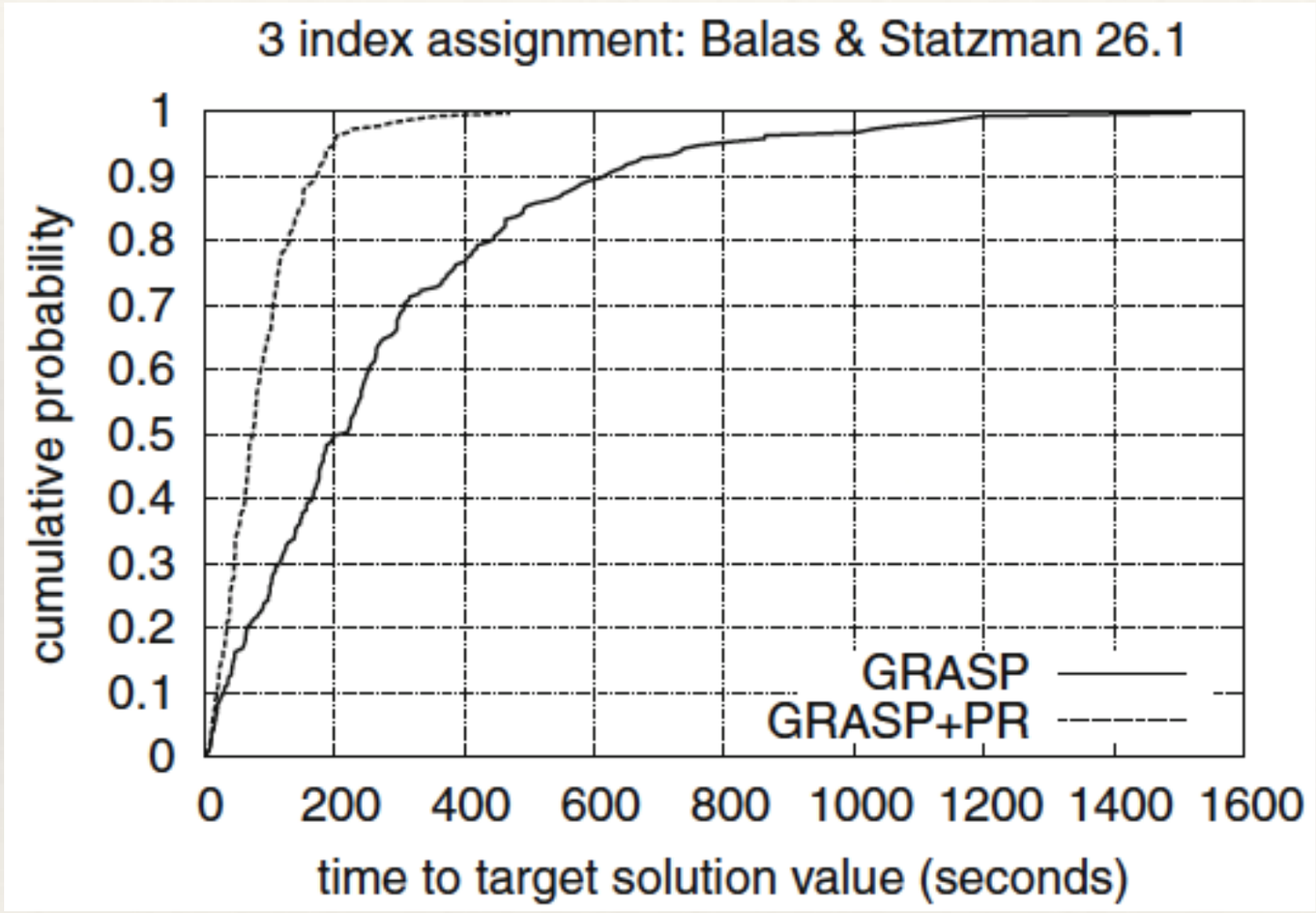
4. Começando da solução *s*, o *melhor movimento* em $\Delta(s,g)$ ainda não usado é aplicado à solução corrente, até obter a solução guia *g*;

5. O *melhor movimento* é aquele que resulta na solução de melhor qualidade na vizinhança restrita;

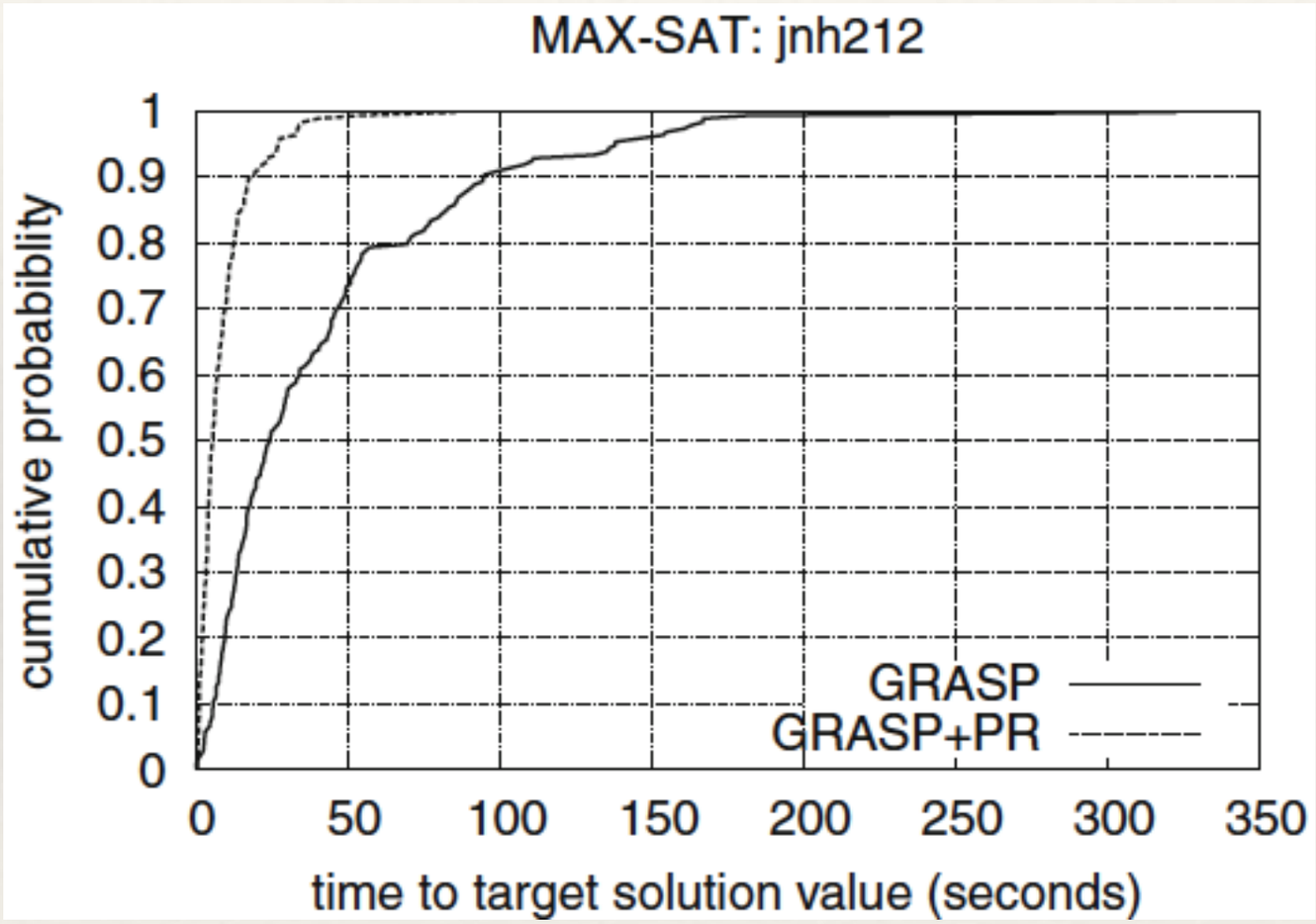
6. A melhor solução encontrada ao longo dessa trajetória é submetida à uma busca local;

7. Essa solução refinada é retornada como a solução produzida pelo algoritmo path-relinking.

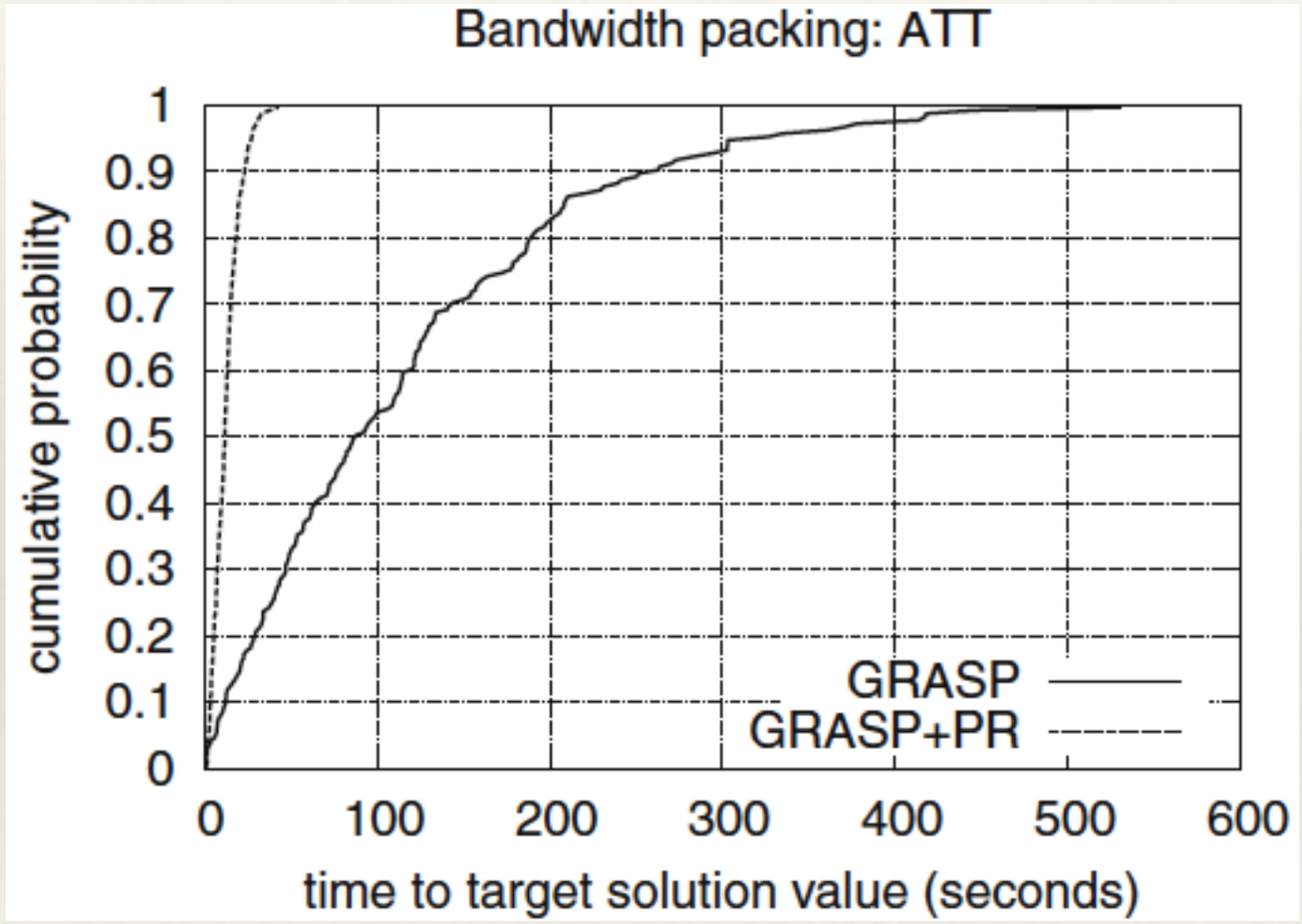
Path-Relinking



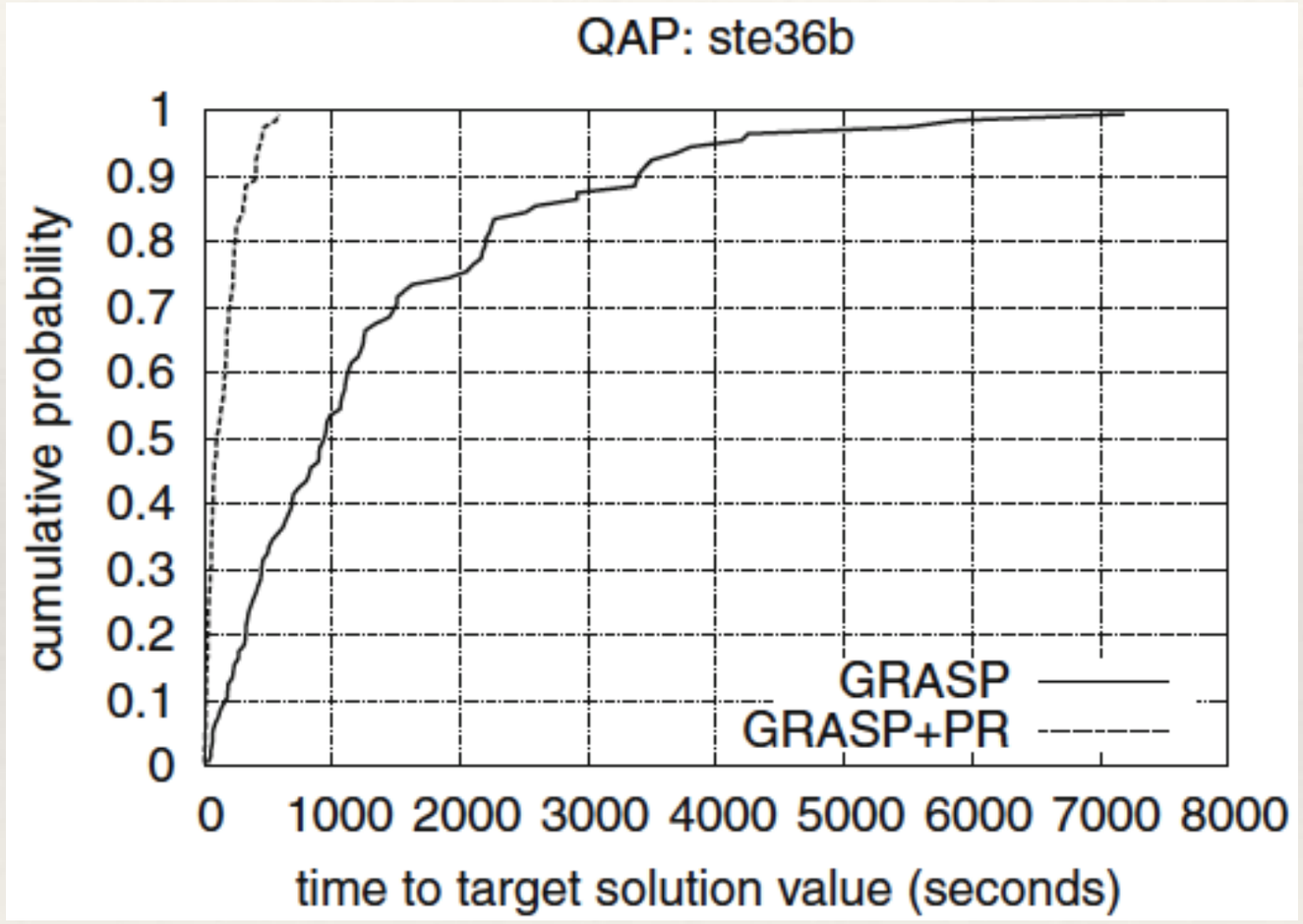
Path-Relinking



Path-Relinking



Path-Relinking



Path-Relinking

- Formas de implementação do path-relinking:
  - estratégia de intensificação, entre cada ótimo local obtido após a fase de busca local do GRASP e uma ou mais soluções da elite;
  - intensificação *a posteriori*, entre cada par de soluções da elite;
  - intensificação *a posteriori*, submetendo o conjunto de soluções elite a um processo evolucionário;
  - qualquer combinação das estratégias acima.

Path-Relinking

- Estratégias de path-relinking:
  - *forward path-relinking*
    - s (ótimo local do GRASP) e g (elite)
  - *backward path-relinking*
    - s (elite) e g (ótimo local do GRASP)
  - *back and forward path-relinking*
  - *mixed path-relinking*

Extensões do GRASP

- GRASP + estrutura de memória (Tabela Hash);
- GRASP + VNS;
- GRASP + VND;
- GRASP + Algoritmo Genético;
- GRASP + Busca Tabu;
- GRASP + ILS.

Reference

- M. Gendreau, J.-Y. Potvin (eds.), *Handbook of Metaheuristics*, Springer, 2nd ed., 2010.