GRASP with path relinking for the 3-index assignment problem

Mauricio G. C. Resende

mgcr@research.att.com

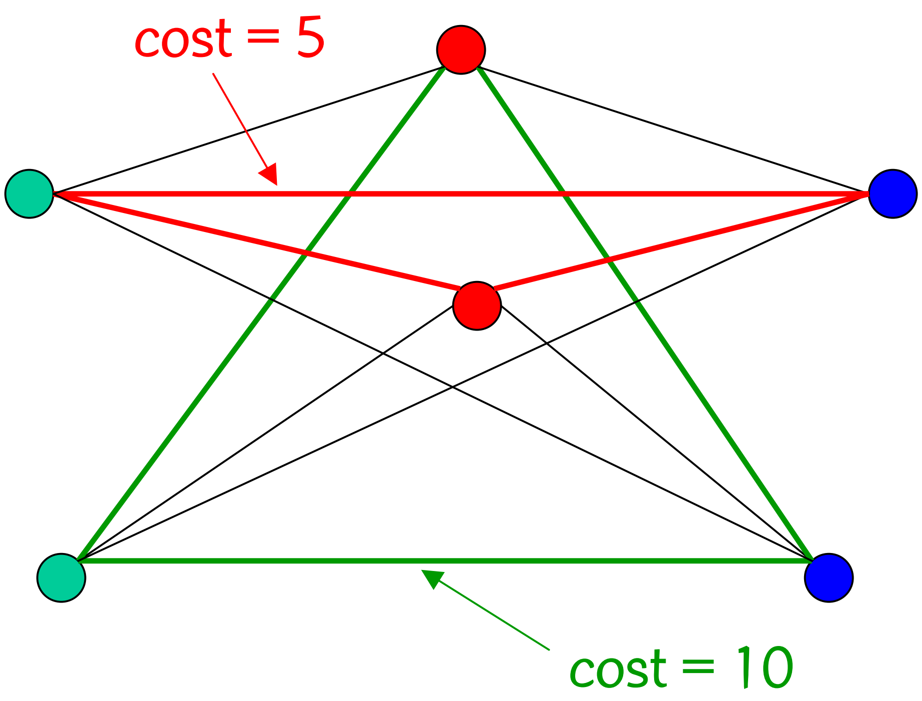
www.research.att.com/~mgcr

Algorithms & Optimization Research Department

Information Sciences Research Center / AT&T Labs Research

Joint work with R.M. Aiex, P.M. Pardalos, & G. Toraldo

3-index assignment (AP3)



Complete tripartite graph: Each triangle made up of three distinctly colored nodes has a cost.

AP3: Find a set of triangles such that each node appears in exactly one triangle and the sum of the costs of the triangles is minimized.

3-index assignment (AP3)

- Let I, J, and K be disjoint sets of size n.
- Consider the complete tripartite graph:
  $K_{n,n,n} = (I \cup J \cup K, (I \times J) \cup (I \times K) \cup (J \times K))$
- If each triangle $(i, j, k) \in I \times J \times K$ costs $c_{i,j,k}$
- AP3 consists in finding a subset $A \subseteq I \times J \times K$ of n triangles such that every element of $I \times J \times K$ occurs in exactly one triangle of A and the cost of the chosen triangles is minimized.

3-index assignment (AP3)

- First stated by Pierskalla (1967) as a straightforward extension of the 2-dim assignment problem.
- AP3 is NP-complete (Frieze, 1983)
- Applications include:
  - Scheduling capital investments
  - Military troop assignment
  - Satellite coverage optimization
  - Production of printed circuit boards

Exact algorithms & heuristics for AP3

- Pierskalla (1967)
- Vlach (1967)
- Hansen & Kaufman (1973)
- Burkard & Fröhlich (1980)
- Balas & Saltzman (1991)
- Crama & Spieksma (1992)
- Burkard & Rudolf (1993)
- Burkard, Rudolf, & Woeginger (1996)

Summary of talk

- GRASP for AP3
  - Construction of greedy randomized solution
  - Local search
- Path relinking for AP3
- GRASP with path relinking for AP3
- Computational experience with sequential algorithms
- Parallel implementation & computation

GRASP: greedy randomized adaptive search procedure

- Multi-start meta-heuristic (Feo & R., 1989)
- Repeat:
  - Construct greedy randomized solution
  - Use local search to improve constructed solution
  - Keep track of best solutions found

GRASP for assignment problems

- QAP: Li, Pardalos, & R. (1994); Pardalos, Pitsoulis, & R. (1995); R., Pardalos, & Li (1996); Pardalos, Pitsoulis, & R. (1997); Rangel, Abreu, Boaventura-Netto, & Boeres (1998); Fleurent & Glover (1999); Pitsoulis (1999); Rangel, Abreu, & Boaventura-Netto (1999); Ahuja, Orlin, & Tiwari (2000)
- Biquadratic assignment: Mavridou, Pardalos, Pitsoulis, & R. (1998)
- Multi-dimensional assignment: Robertson (1998); Murphey, Pardalos, & Pitsoulis (1998); Pitsoulis (1999)

GRASP for assignment problems

- Intermodal trailer assignment: Feo & Gonzalez-Velarde (1995)
- Turbine balancing: Pitsoulis (1999); Pitsoulis, Pardalos, & Hearn (2001)

Greedy randomized construction for AP3

- Solution A is built by selecting n triplets, one at a time.
- Let C be the set of candidate triplets (initially the set of all triplets)
- $c_* = \min \{c_{i,j,k} \mid (i,j,k) \in C\}$; $c^* = \max \{c_{i,j,k} \mid (i,j,k) \in C\}$
- $C' = \{ (i,j,k) \in C \mid c_{i,j,k} \leq c_* + \alpha (c^* - c_*) \}$

$(\alpha$ random, $0 \leq \alpha \leq 1)$

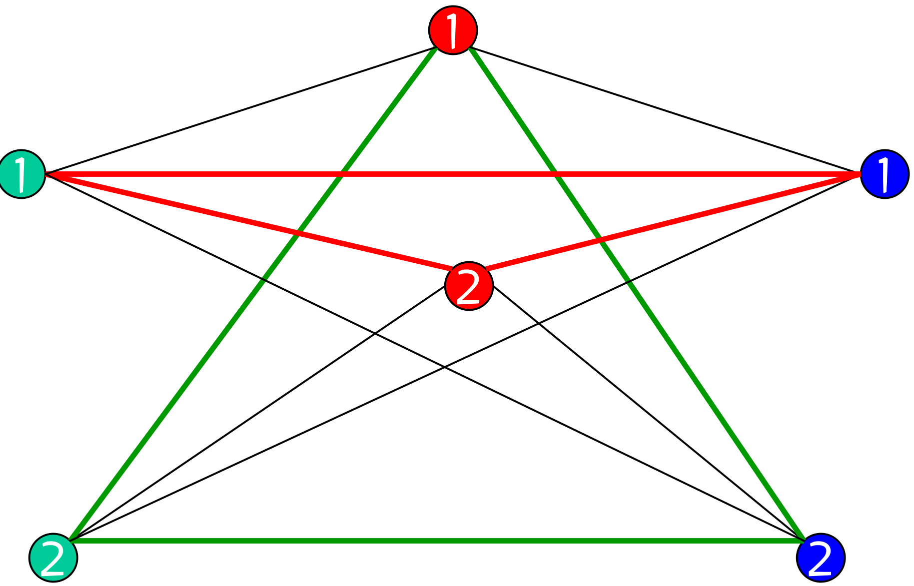
Greedy randomized construction for AP3

- $A = \varnothing$
- Repeat $n-1$ times:
  - Build restricted candidate list C'
  - Choose $(i,j,k) \in C'$ at random
  - $A = A \cup (i,j,k)$
  - Update candidate list C
- $A = A \cup C$

Data structure uses 4 doubly linked lists.

Local search for AP3

- Permutation representation of AP3 solution.



$(p, q) = (\{2,1\}, \{1,2\})$

Solution space consists of all $(n!)^2$ possible combinations of permutations.

Local search for AP3

- Difference between 2 permutations s and s':

$$\delta(s,s') = \{ i \mid s(i) \neq s'(i) \}$$

- Distance between them:

$$d(s,s') = |\delta(s,s')|$$

- The neighborhood used in our local search:

$$N_2(p,q) = \{ p', q' \mid d(p,p') + d(q,q') = 2 \}$$

Local search for AP3

(p,q) is starting solution;

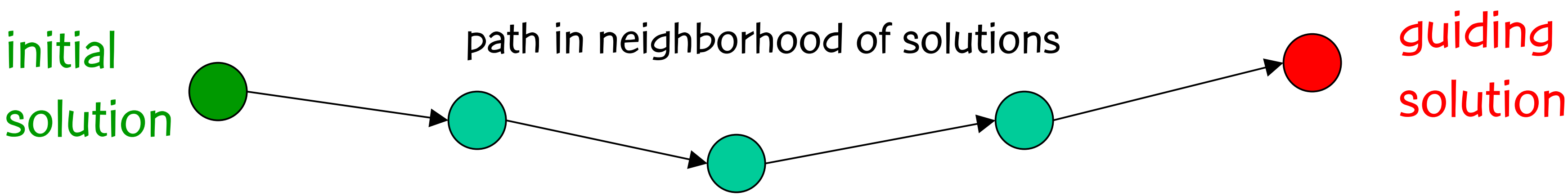
while ($\exists\, (p',q') \in N_2(p,q) \mid c(p',q') < c(p,q)$){

$(p,q) = (p',q')$;

}

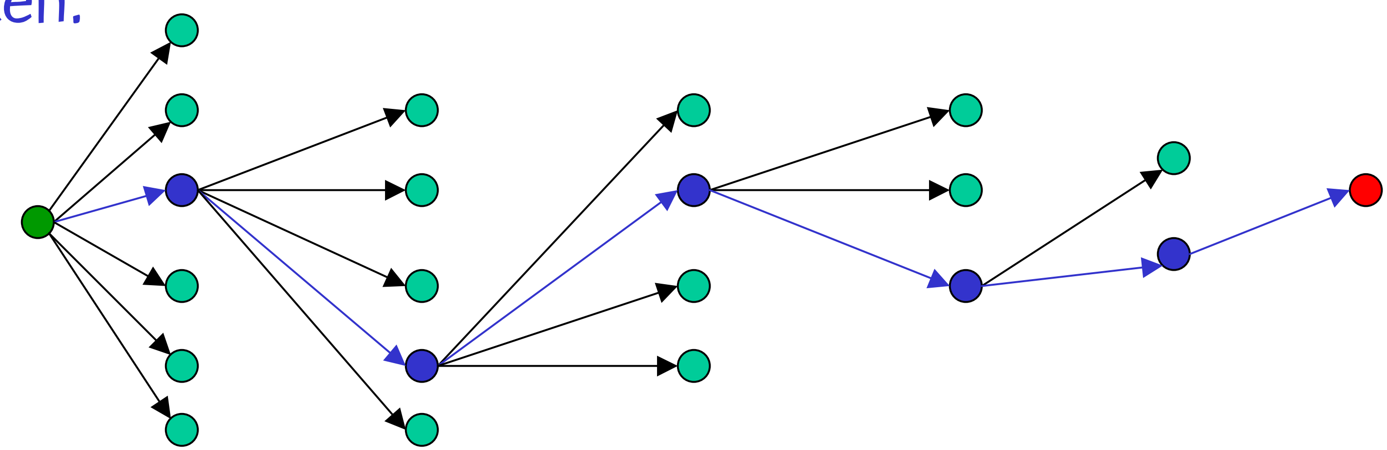
Path relinking

- Introduced in context of tabu search in Glover & Laguna (1997):
  - Approach to integrate intensification & diversification in search.
- Consists in exploring trajectories that connect high quality solutions.



Path relinking

- Path is generated by selecting moves that introduce in the initial solution attributes of the guiding solution.
- At each step, all moves that incorporate attributes of the guiding solution are analyzed and best move is taken.



AT&T

Path relinking in GRASP

- Introduced by Laguna & Martí (1999)
- Maintain an elite set of solutions found during GRASP iterations.
- After each GRASP iteration (construction & local search):
  - Select an elite solution at random: guiding solution.
  - Use GRASP solution as initial solution.
  - Do path relinking between these two solutions.

Path relinking for AP3

- Path relinking is done between
  - Initial solution

    $$S = \{ (1, j_1^S, k_1^S), (2, j_2^S, k_2^S), \ldots, (n, j_n^S, k_n^S) \}$$

  - Guiding solution

    $$T = \{ (1, j_1^T, k_1^T), (2, j_2^T, k_2^T), \ldots, (n, j_n^T, k_n^T) \}$$

AT&T

Path relinking for AP3

- Symmetric difference between S and T:

$$\delta J = \{ i = 1,\ldots,n \mid j_i^S \neq j_i^T \}$$

$$\delta K = \{ i = 1,\ldots,n \mid k_i^S \neq k_i^T \}$$

- while ($|\delta J| + |\delta K| > 0$) {

    evaluate moves corresponding to δJ and δK

    make best move

    update symmetric difference

}

Path relinking moves

- Guided by δJ : for all $i \in \delta J$, let q be such that $j_q^T = j_i^S$

  Triplets $\{(i, j_i^S, k_i^S), (q, j_q^S, k_q^S)\}$ are replaced by

  triplets $\{(i, j_q^S, k_i^S), (q, j_i^S, k_q^S)\}$

- Guided by δK: for all $i \in \delta K$, let q be such that $k_q^T = k_i^S$

  Triplets $\{(i, j_i^S, k_i^S), (q, j_q^S, k_q^S)\}$ are replaced by

  triplets $\{(i, j_i^S, k_q^S), (q, j_q^S, k_i^S)\}$

Path relinking: Elite set

- P is set of elite solutions
- Each iteration of first $|P|$ GRASP iterations adds one solution to P.
- After that: solution x is promoted to P if:
  - x is better than best solution in P.
  - x is not better than best solution in P, but is better than worst and it is sufficiently different from all solutions in P.

Path relinking: Solution dissimilarity

- Initial solution

$$S = \{ (1, j_1^S, k_1^S), (2, j_2^S, k_2^S), \ldots, (n, j_n^S, k_n^S) \}$$

- Guiding solution

$$T = \{ (1, j_1^T, k_1^T), (2, j_2^T, k_2^T), \ldots, (n, j_n^T, k_n^T) \}$$

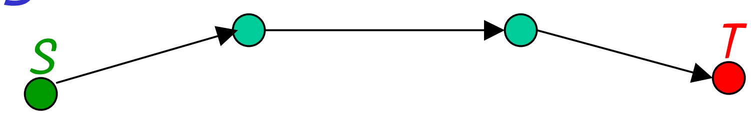
- Dissimilarity: $\Delta(S, T)$ = count of non-matching triplet indices.
- Solutions are sufficiently different if $\Delta(S, T) > n$

Path relinking: Intensification & post-optimization

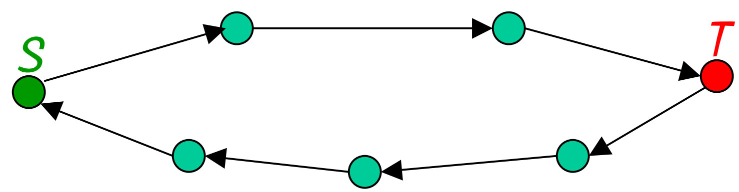
- Elite set intensification (periodically or as post-optimization phase):
  - Apply path relinking between all pairs of elite set solutions.
  - Update elite set, if necessary, and repeat until no change occurs.
- If done as post-optimization:
  - Apply local search to each elite set solution.
  - Repeat if necessary.

Path relinking: Variants

- How targets are chosen:
  - Select a subset of targets $\underline{P} \subseteq P$ from elite set.
  - We test $|\underline{P}| = 1$ and $|\underline{P}| = |P|$.
- Direction of path relinking:
  - Forward: from S to T.



  - Forward and back: from S to T, then from T to S.



Computational experiments

- Test problems (358 instances):
  - Balas & Saltzman: Integer costs $c_{i,j,k}$ randomly generated in uniform interval [0,100]. Five instances of sizes n = 12,14,16,18, 20, 22, 24, and 26.
  - Crama & Spieksma: Edge (i,j) of $K_{n,n,n}$ has cost $d_{i,j}$ and triplet (i,j,k) has cost $c_{i,j,k} = d_{i,j} + d_{i,k} + d_{k,j}$. Three types of instances use different schemes to generate the costs $d_{i,j}$. Each type has three instances of sizes n = 33 and 66.
  - Burkard, Rudolf, & Woeginger: $c_{i,j,k} = \alpha_i * \beta_j * \gamma_k$, where α_i, β_j, and γ_k are uniformly distributed in [0,10]. One hundred instances of sizes n = 12, 14, and 16.

Computational experiments: Algorithm variants

- GRASP: pure GRASP with no path relinking
- GPR(RAND): Adds to GRASP 2-way PR between initiating & randomly selected guiding solution.
- GPR(ALL): Adds to GRASP 2-way PR between initiating & all elite solutions.
- GPR(RAND,POST): Adds to GPR(RAND) a post-optimization PR phase.
- GPR(ALL,POST): Adds to GPR(ALL) a post-optimization PR phase.

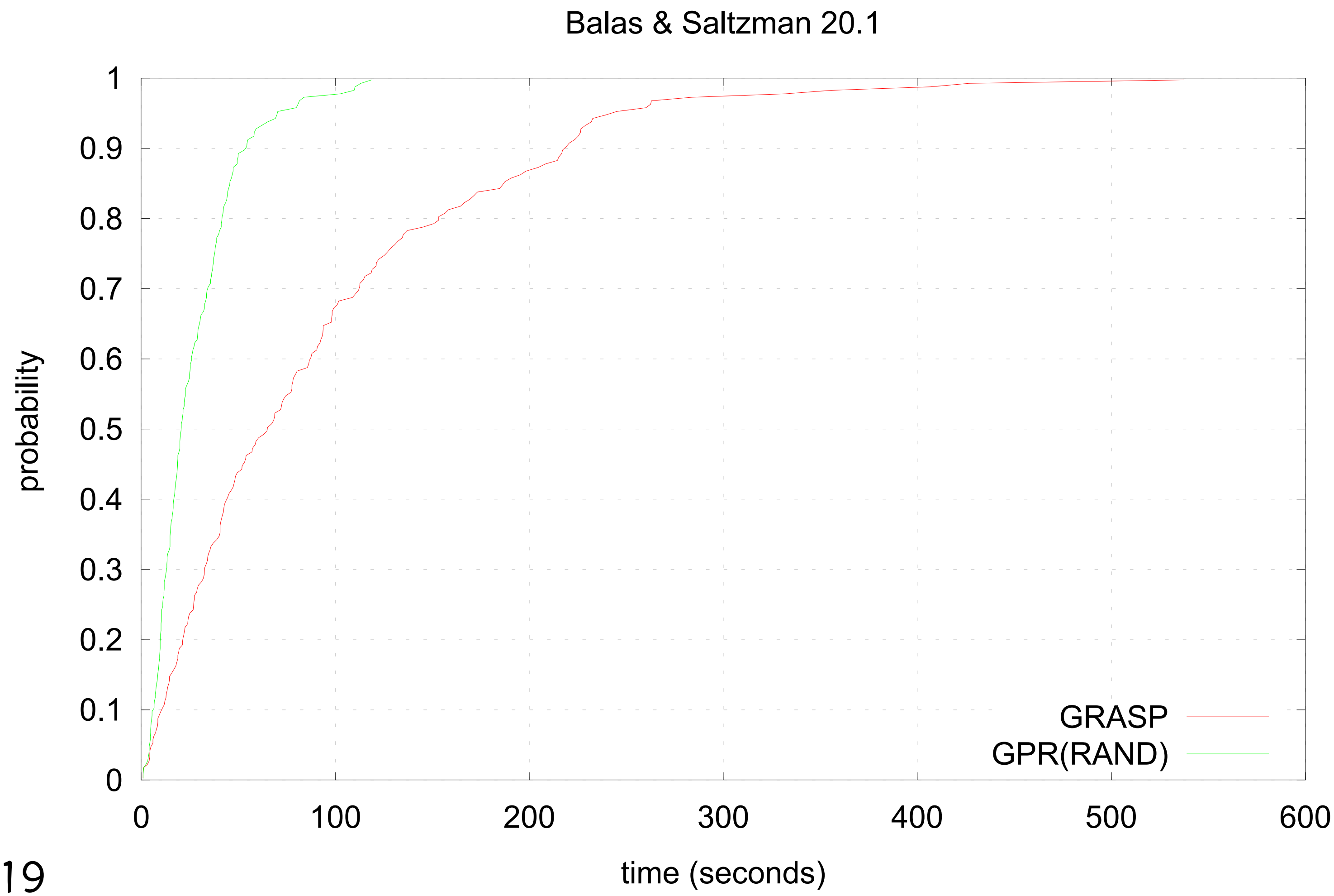
Computational experiments: Algorithm variants

- GPR(RAND,POST,INT): Adds an intensification phase to GPR(RAND,POST). Intensification is done in fixed intervals.
- GPR(ALL,POST,INT): Adds an intensification phase to GPR(ALL,POST). Intensification is done in fixed intervals.

Computational experiments: Questions

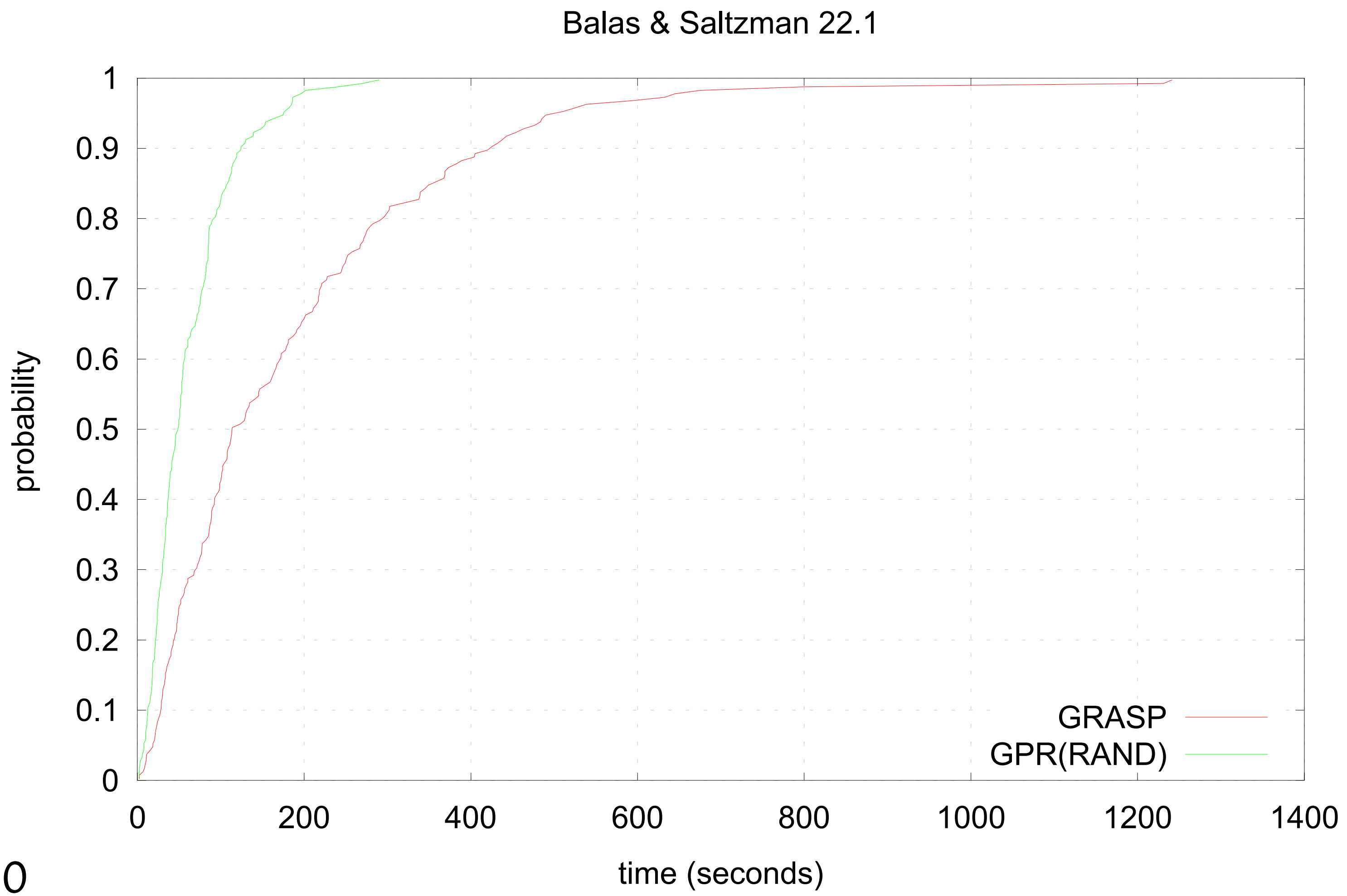
- Does PR improve performance of GRASP and what is the tradeoff in terms of CPU time?
- What are the tradeoffs between CPU time and solution quality for the different variants of GRASP with PR?
- Are random variables *time to target solution* exponentially distributed, and if so, how does a straightforward parallel implementation do?

200 independent runs
of each algorithm.



look4 = 19

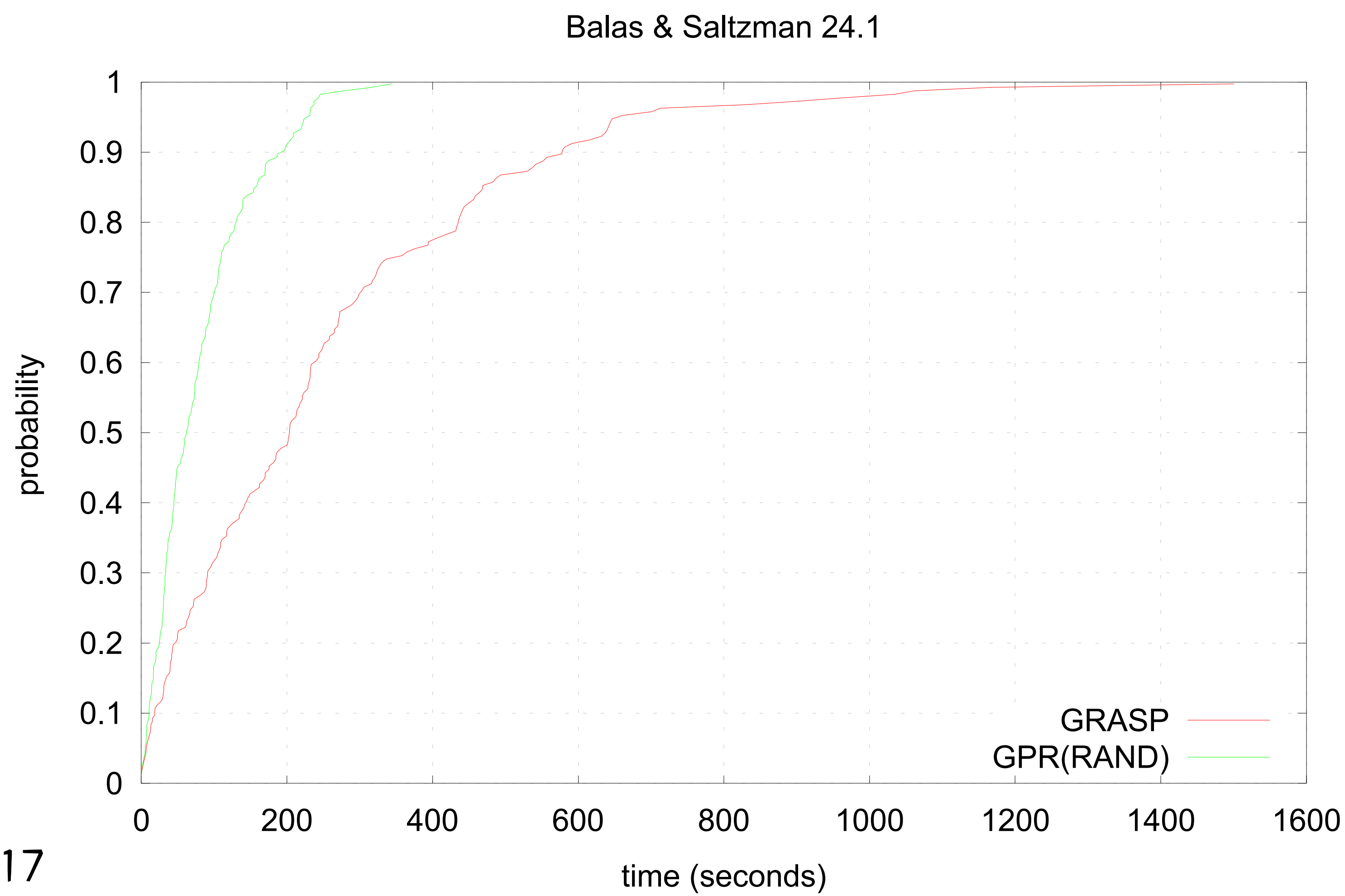
200 independent runs
of each algorithm.



look4 = 20

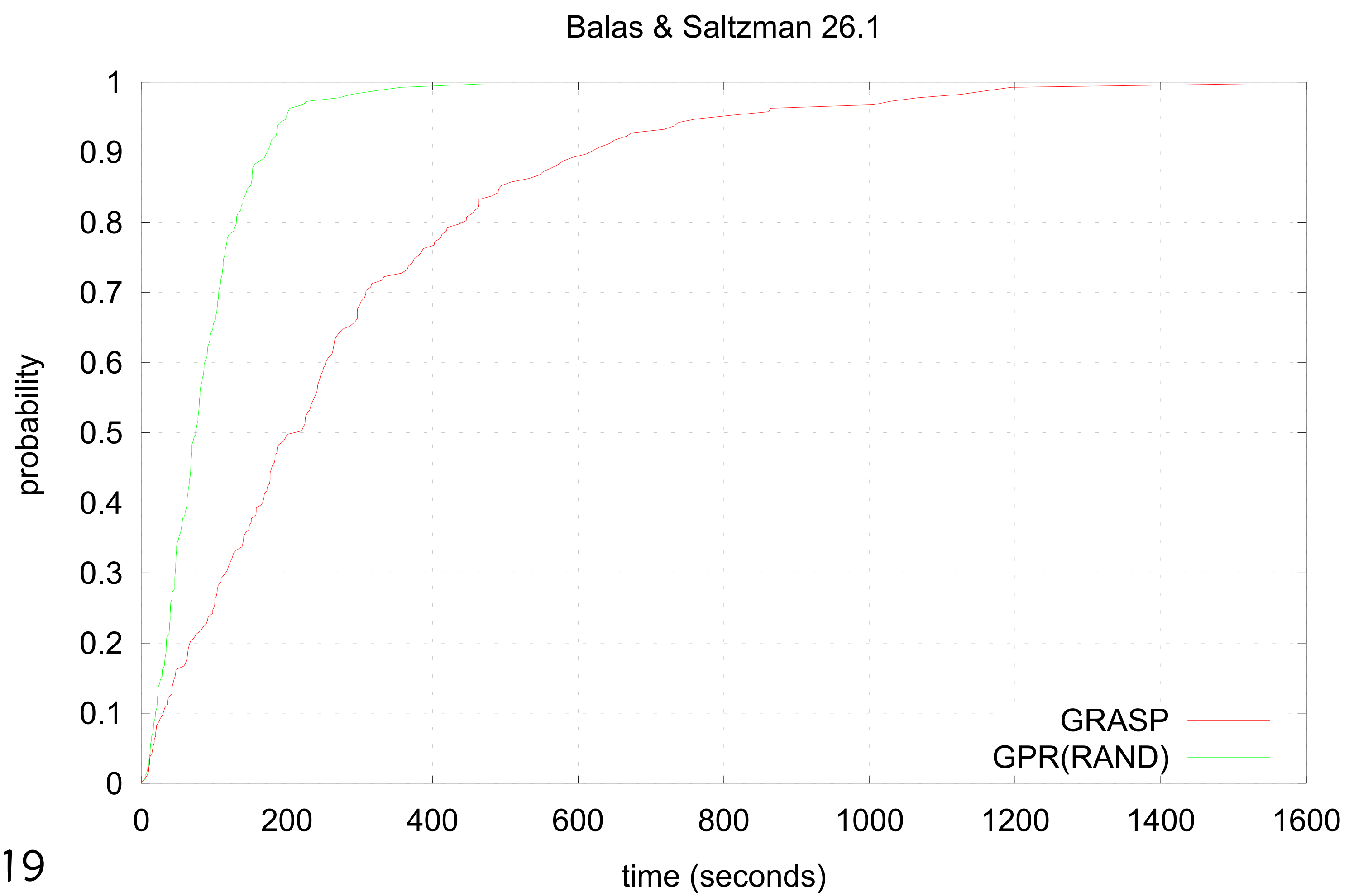
AT&T

200 independent runs
of each algorithm.



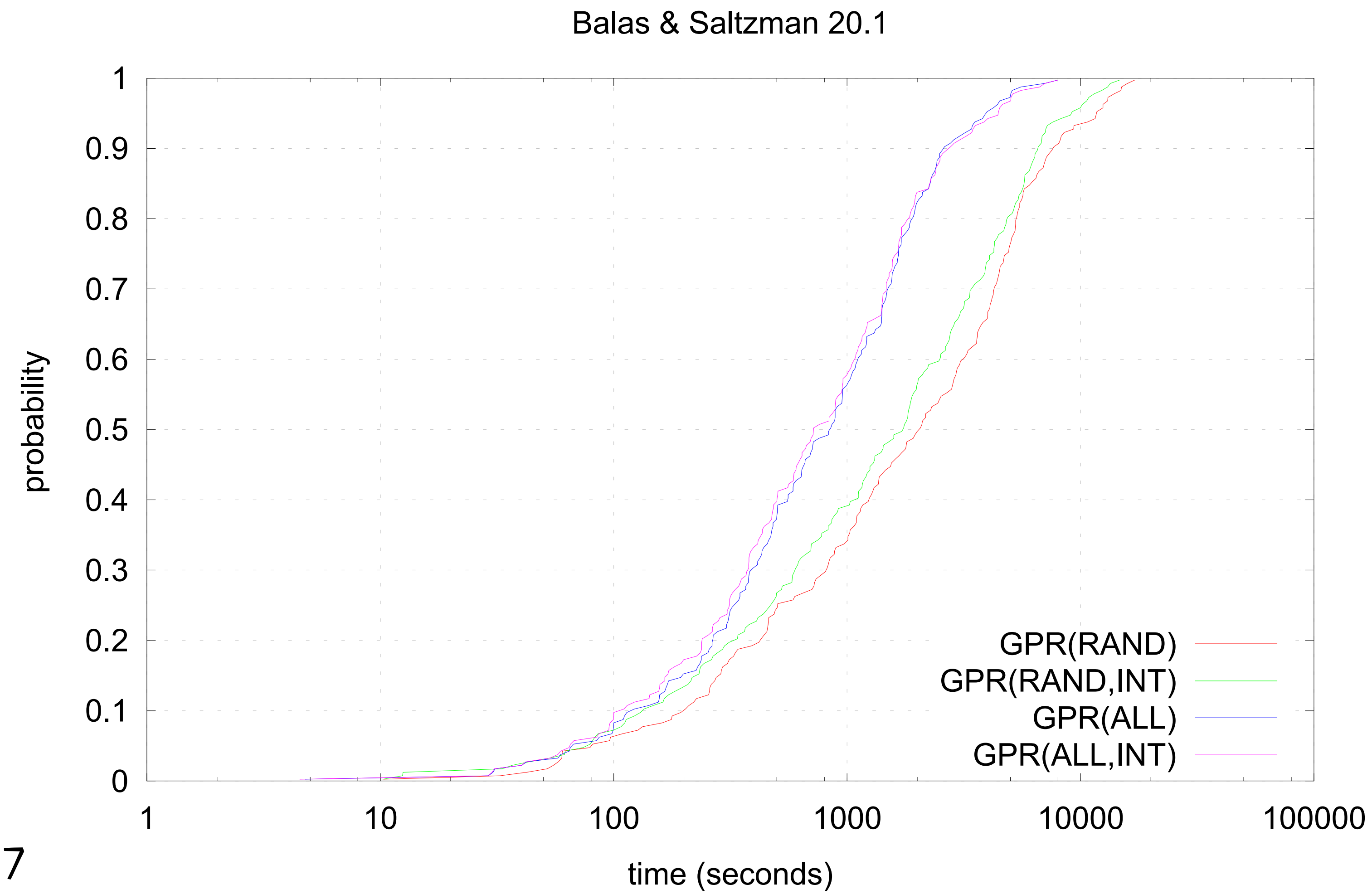
look4 = 17

200 independent runs
of each algorithm.



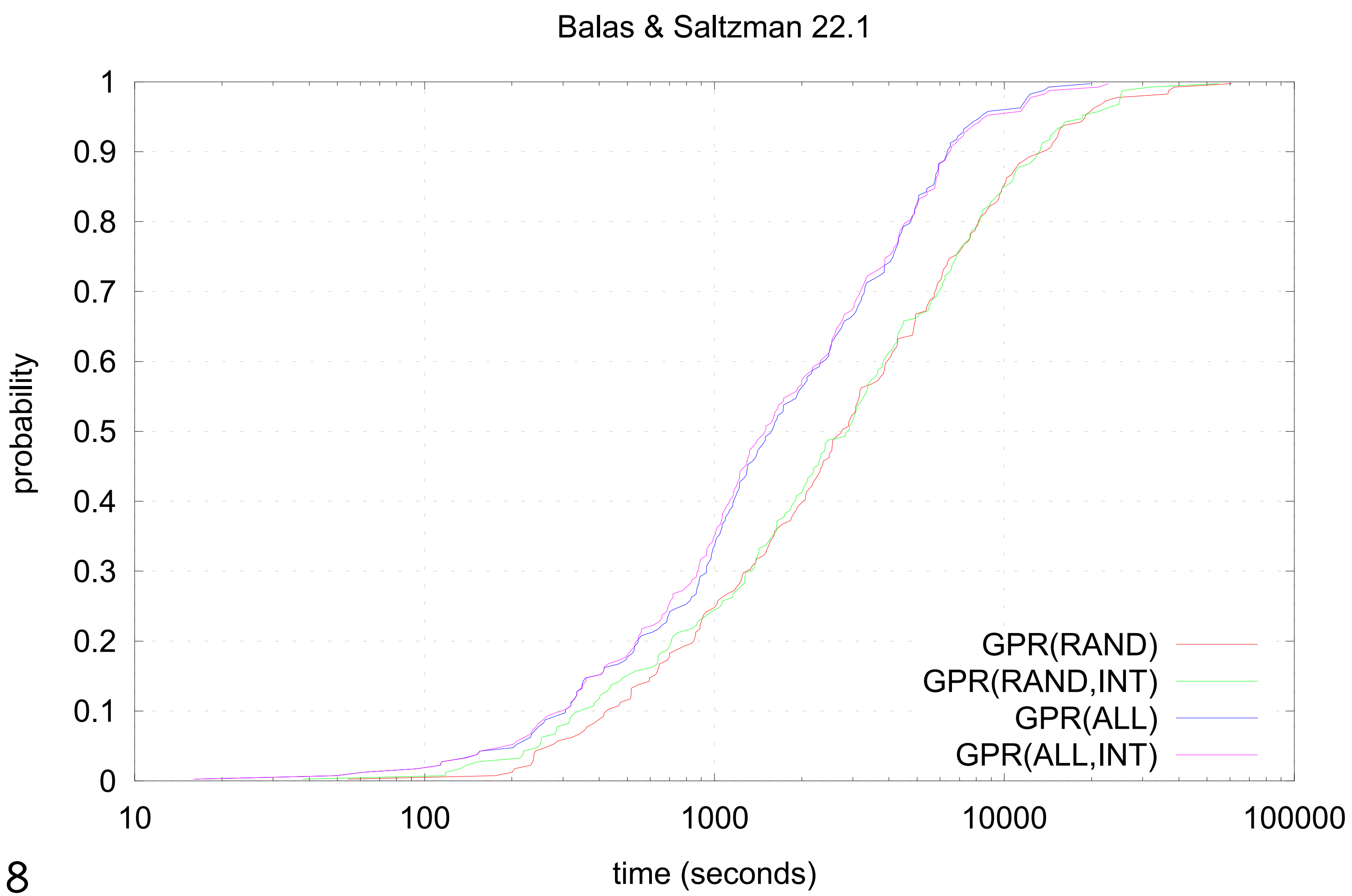
look4 = 19

200 independent runs
of each algorithm.



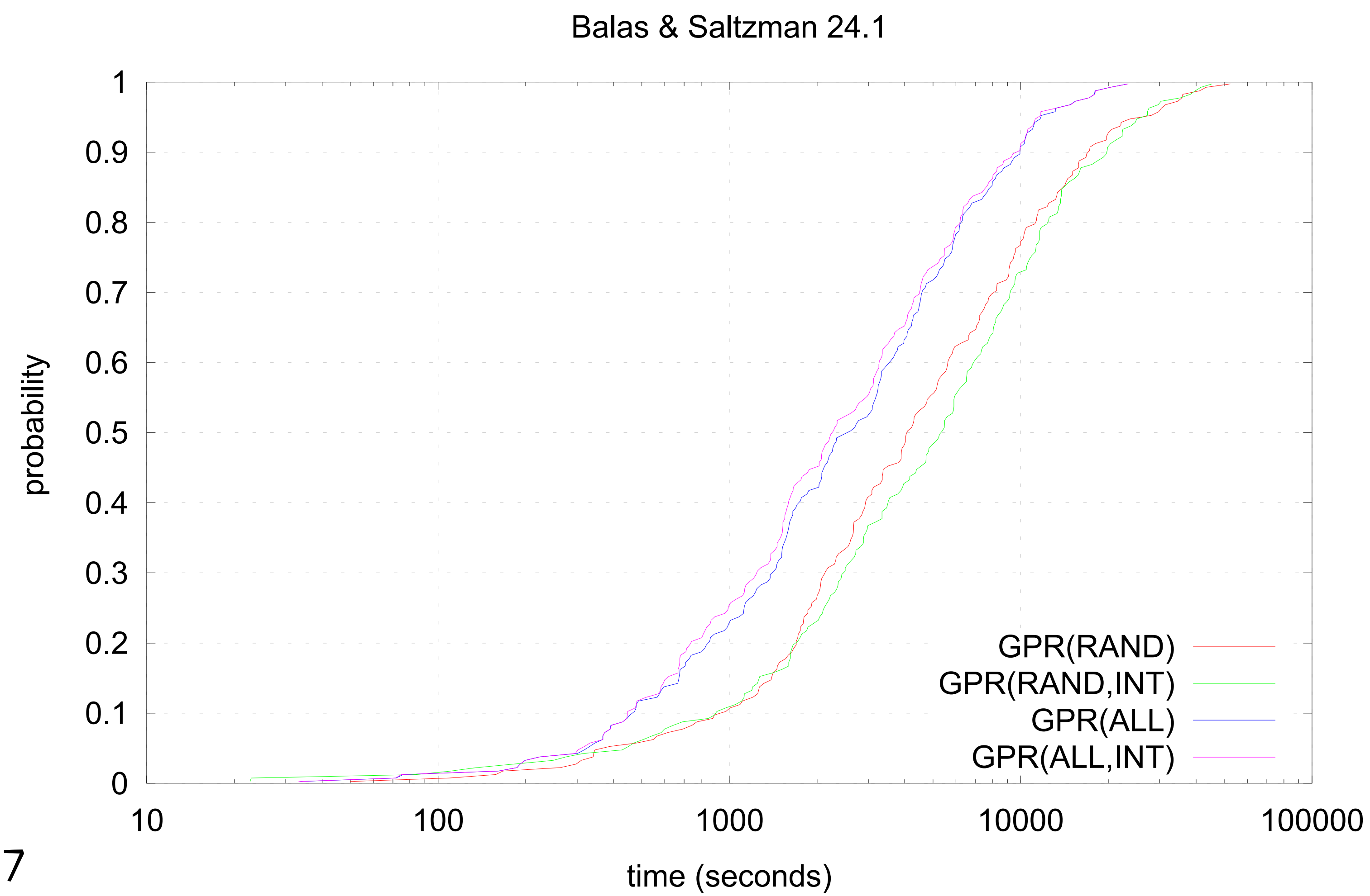
look4 = 7

200 independent runs
of each algorithm.



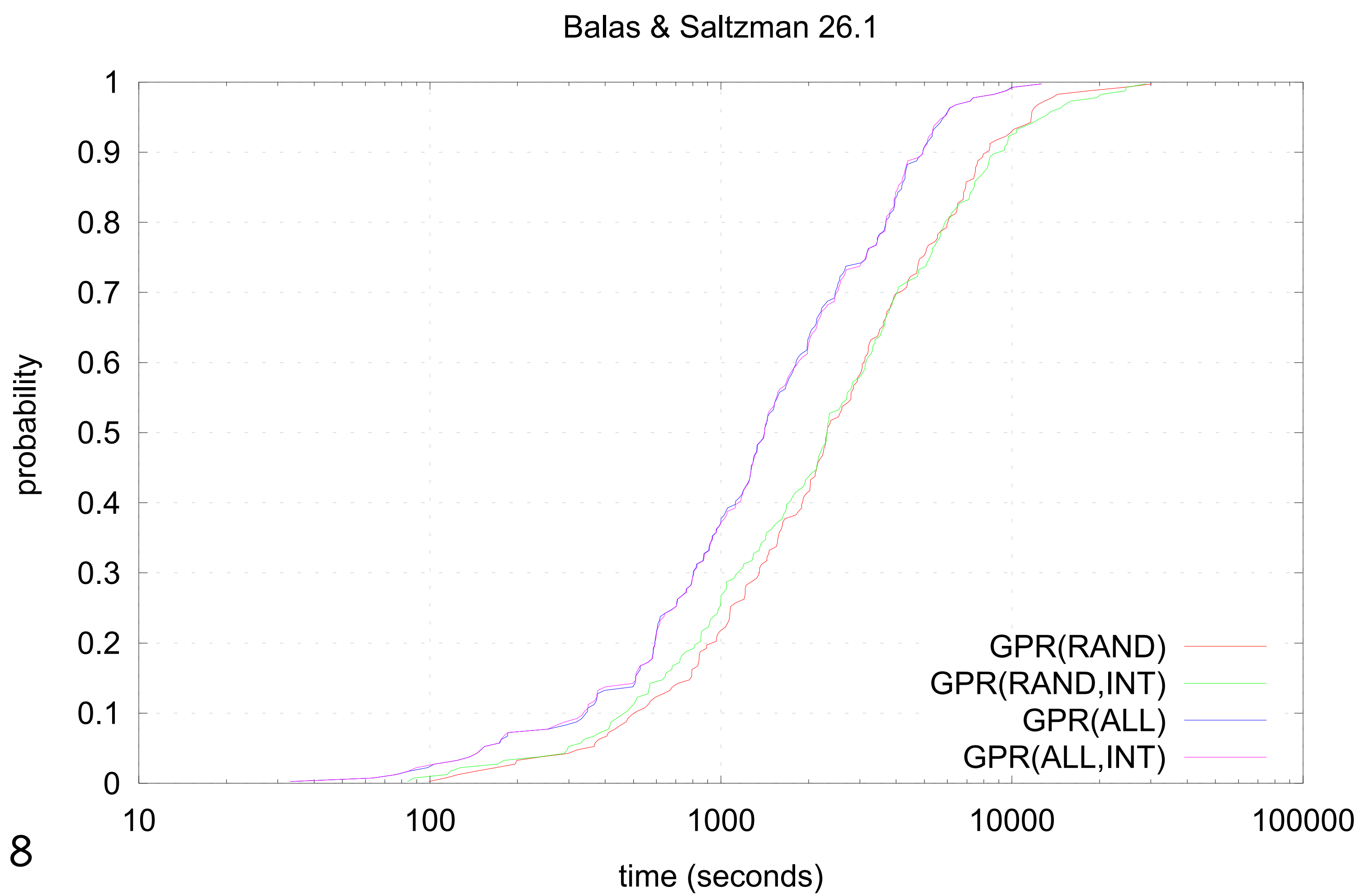
look4 = 8

200 independent runs
of each algorithm.



look4 = 7

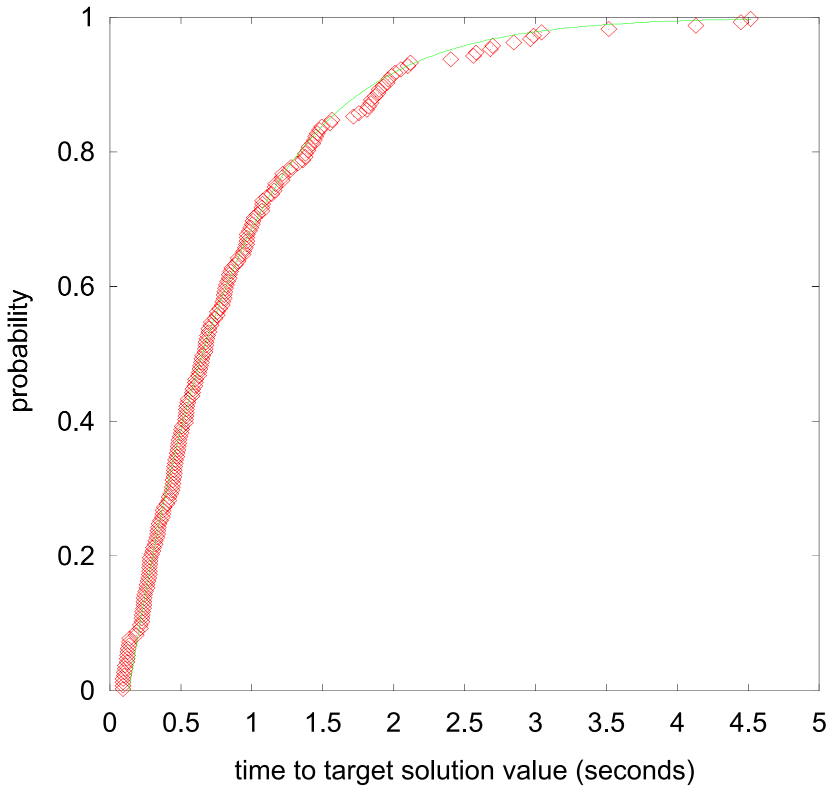
200 independent runs
of each algorithm.

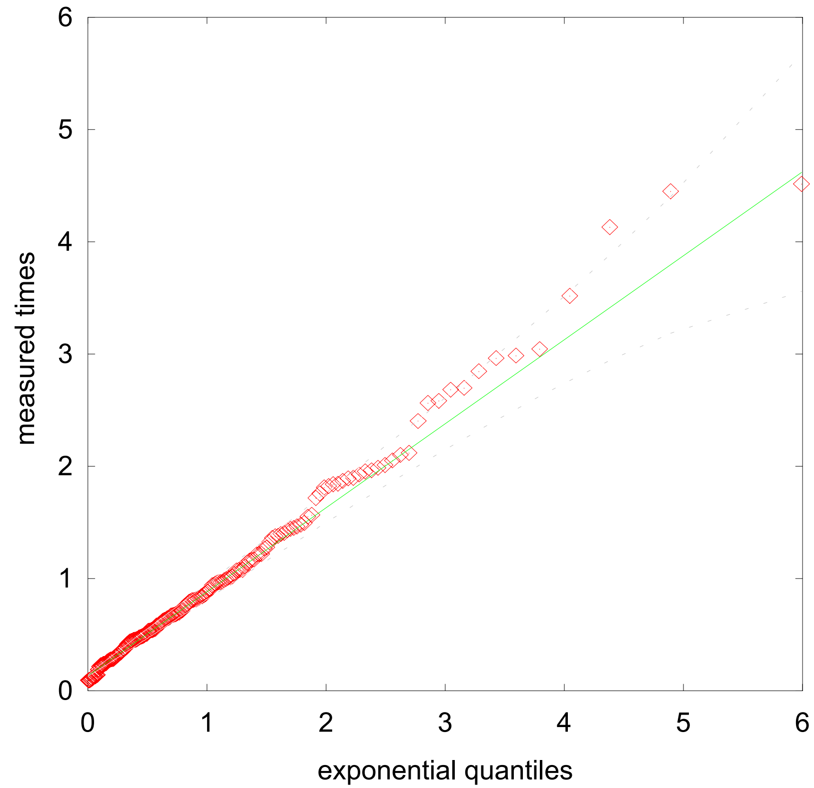


look4 = 8

Computational experiments: General remarks

- Extensive computational experiments were done.
- GRASP with path relinking was shown to improve performance of pure GRASP
  - Finds solution faster.
  - Finds better solutions in fixed number of iterations.
- In general, variants requiring more work per iteration were shown to find solutions of a given quality in less time than variants doing less work per iteration.
- New GRASP with path relinking improved upon all previously described heuristics.





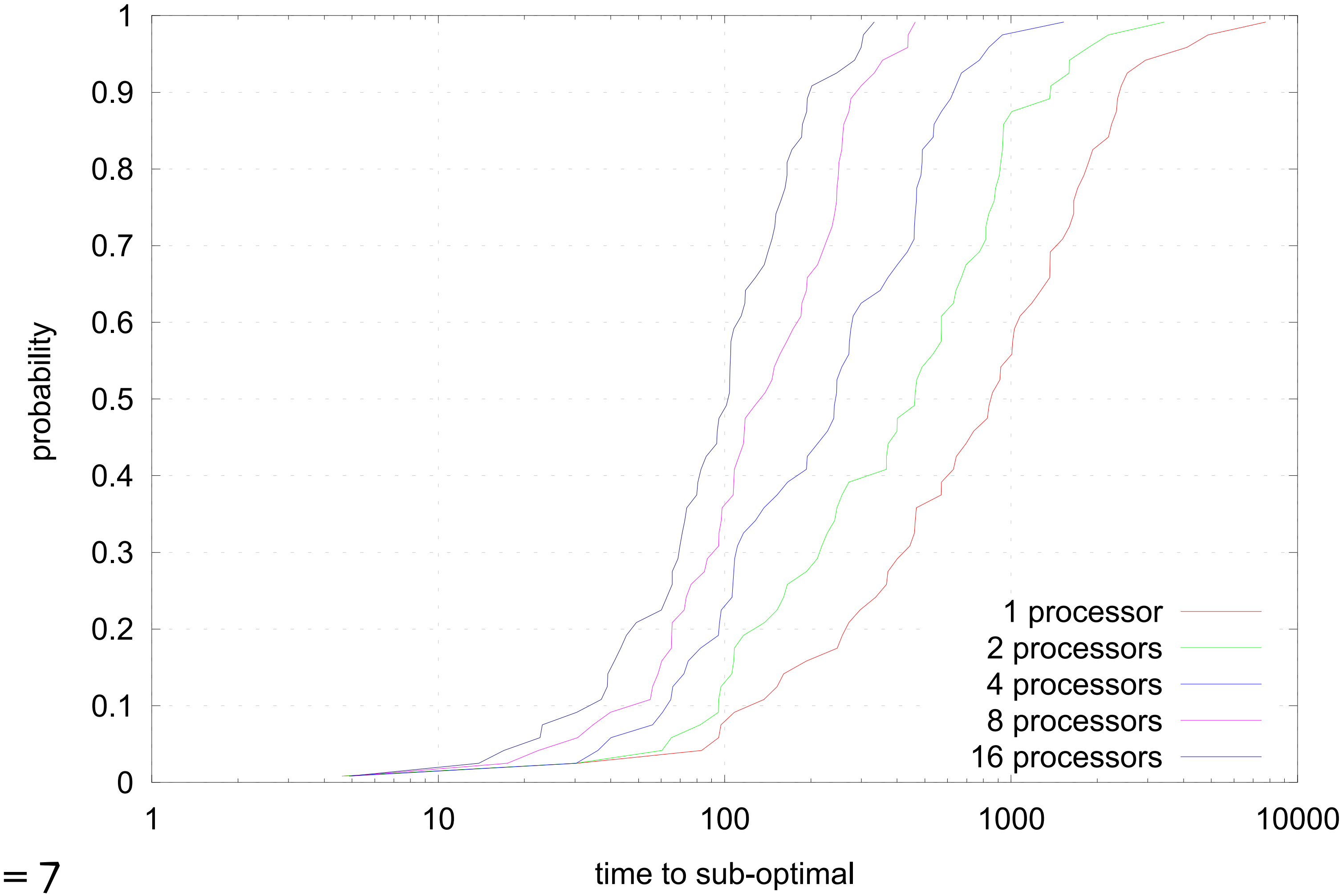
Use standard graphical methodology described in Aiex, R., & Ribeiro (2000) to study if random variable *time to target solution value* fits a two-parameter exponential distribution.

Since it does, one should expect approximate linear speedup in a straightforward parallel implementation.

60 independent runs of each algorithm.

MPI implementation.

Balas & Saltzman 20.1

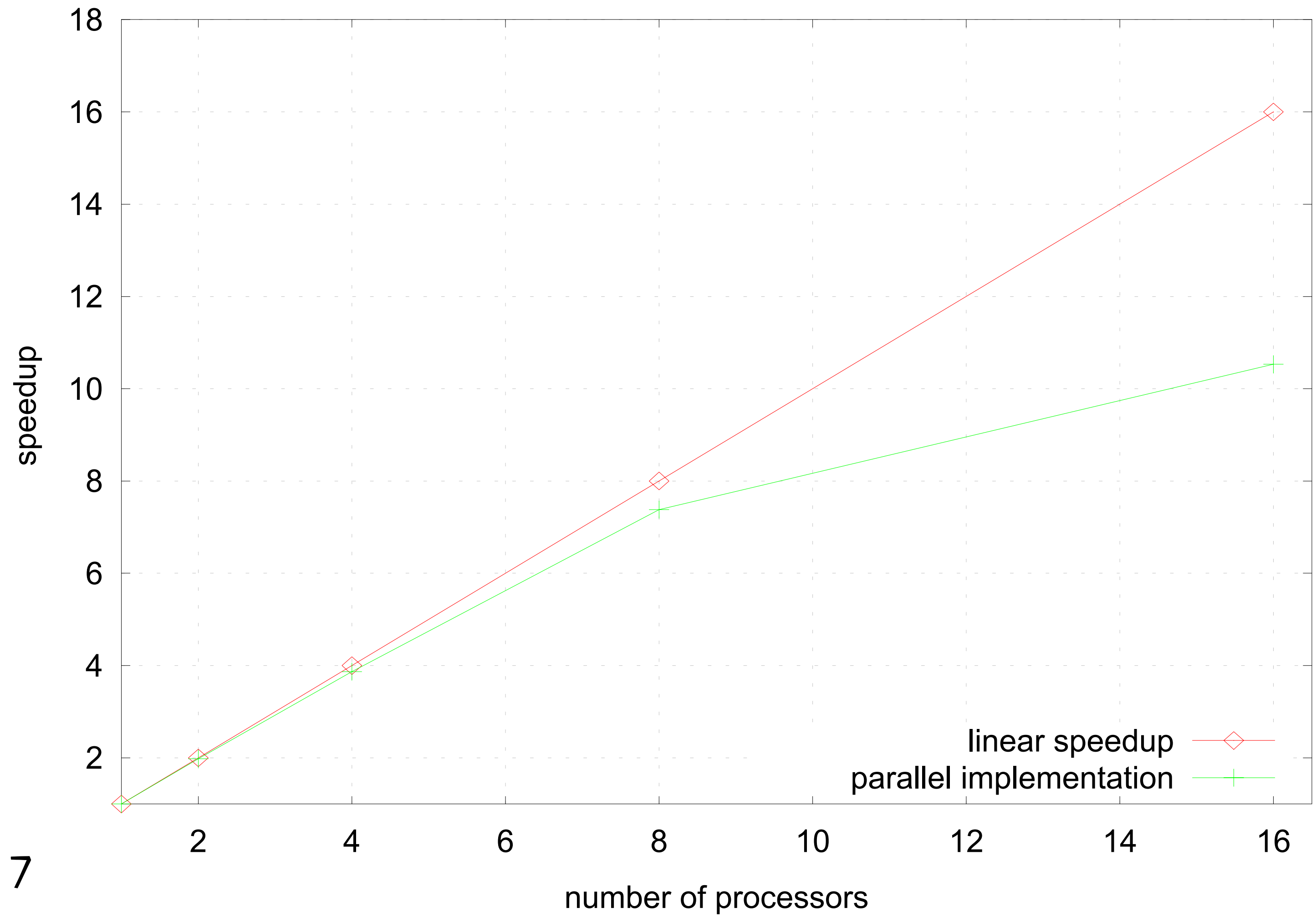


look4 = 7

Average speedup of 60 independent runs.

MPI implementation.

Balas & Saltzman 20.1

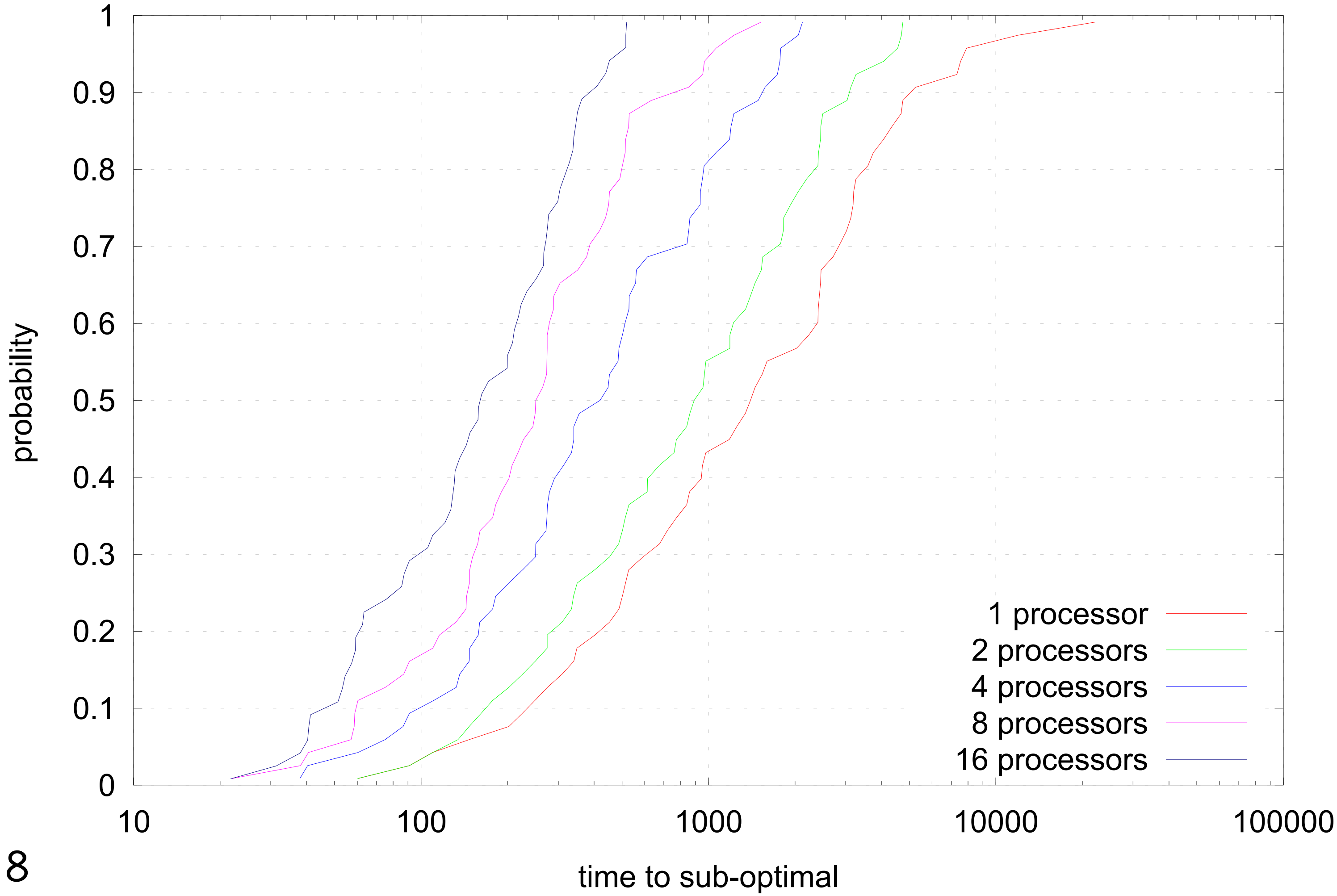


look4 = 7

60 independent runs of each algorithm.

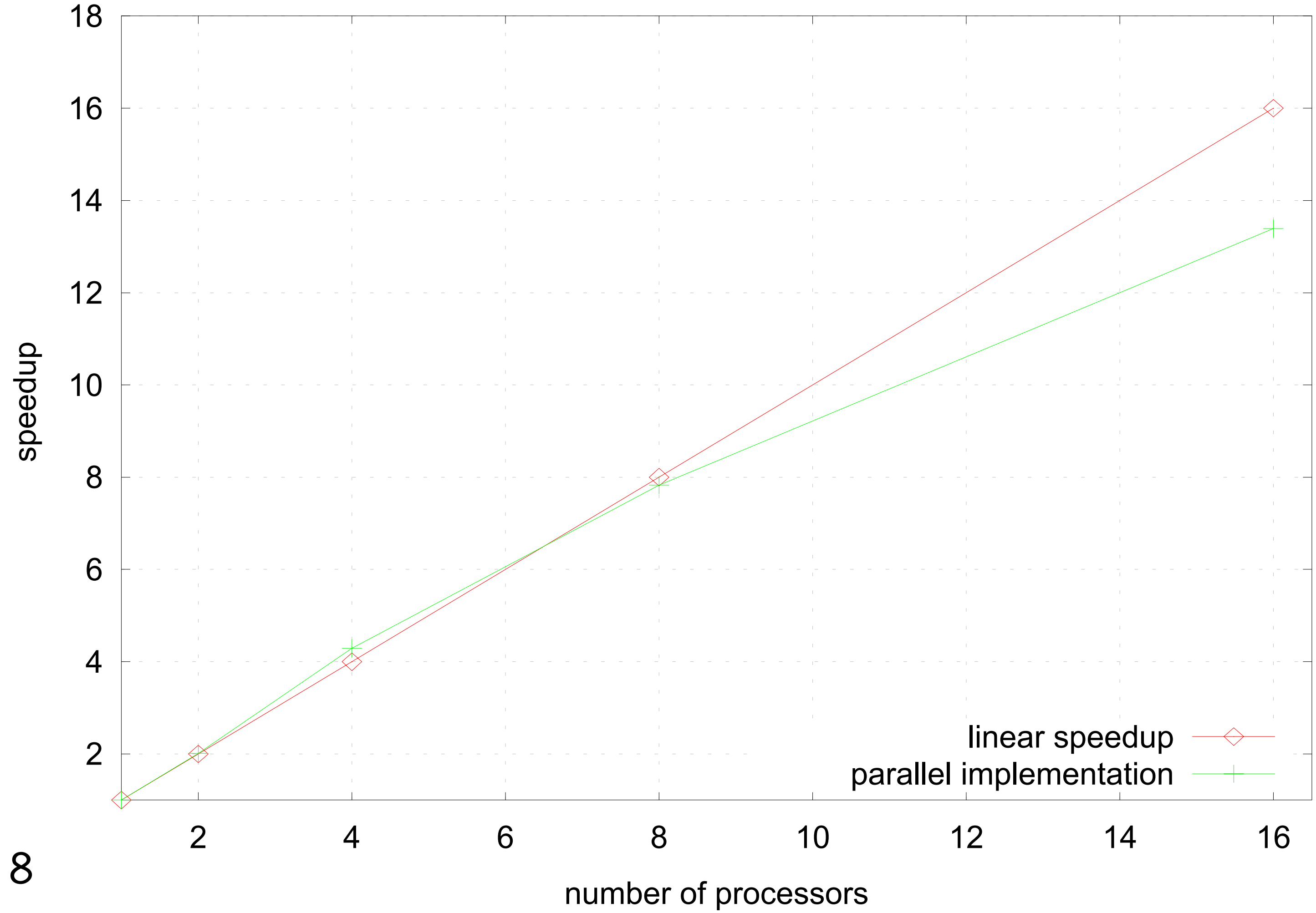
MPI implementation.

Balas & Saltzman 22.1



look4 = 8

Balas & Saltzman 22.1

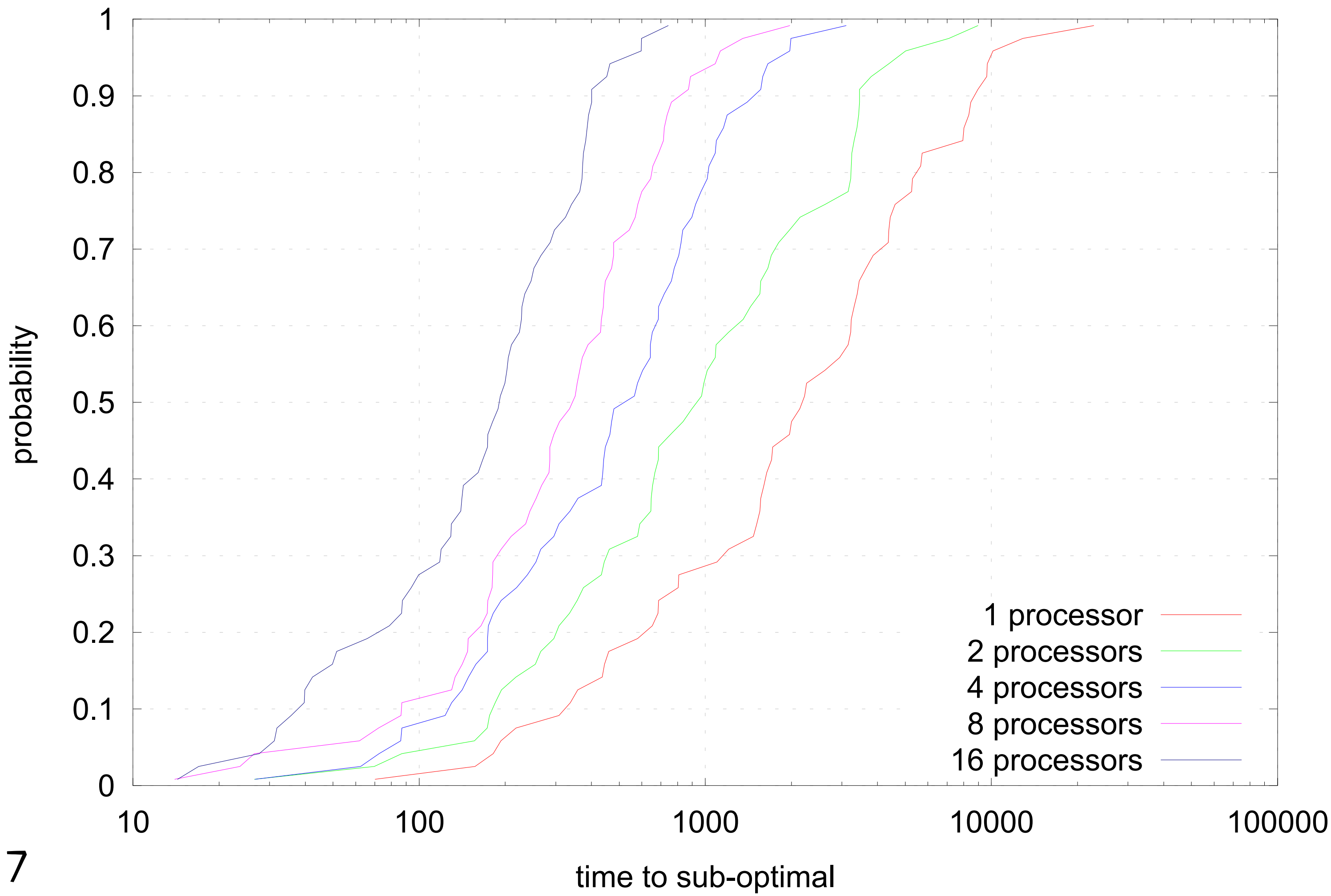


look4 = 8

60 independent runs of each algorithm.

MPI implementation.

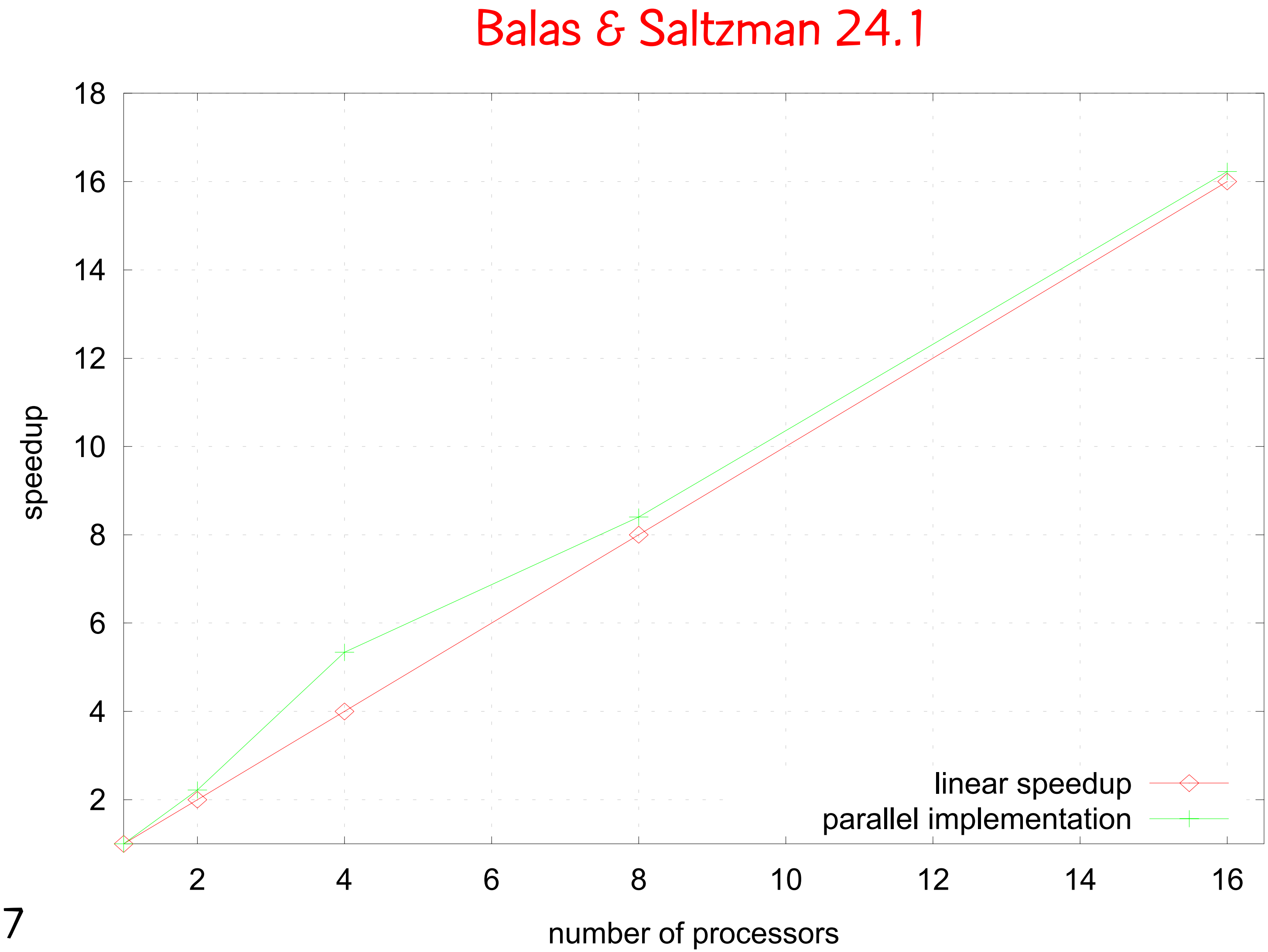
Balas & Saltzman 24.1



look4 = 7

Average speedup of 60 independent runs.

MPI implementation.

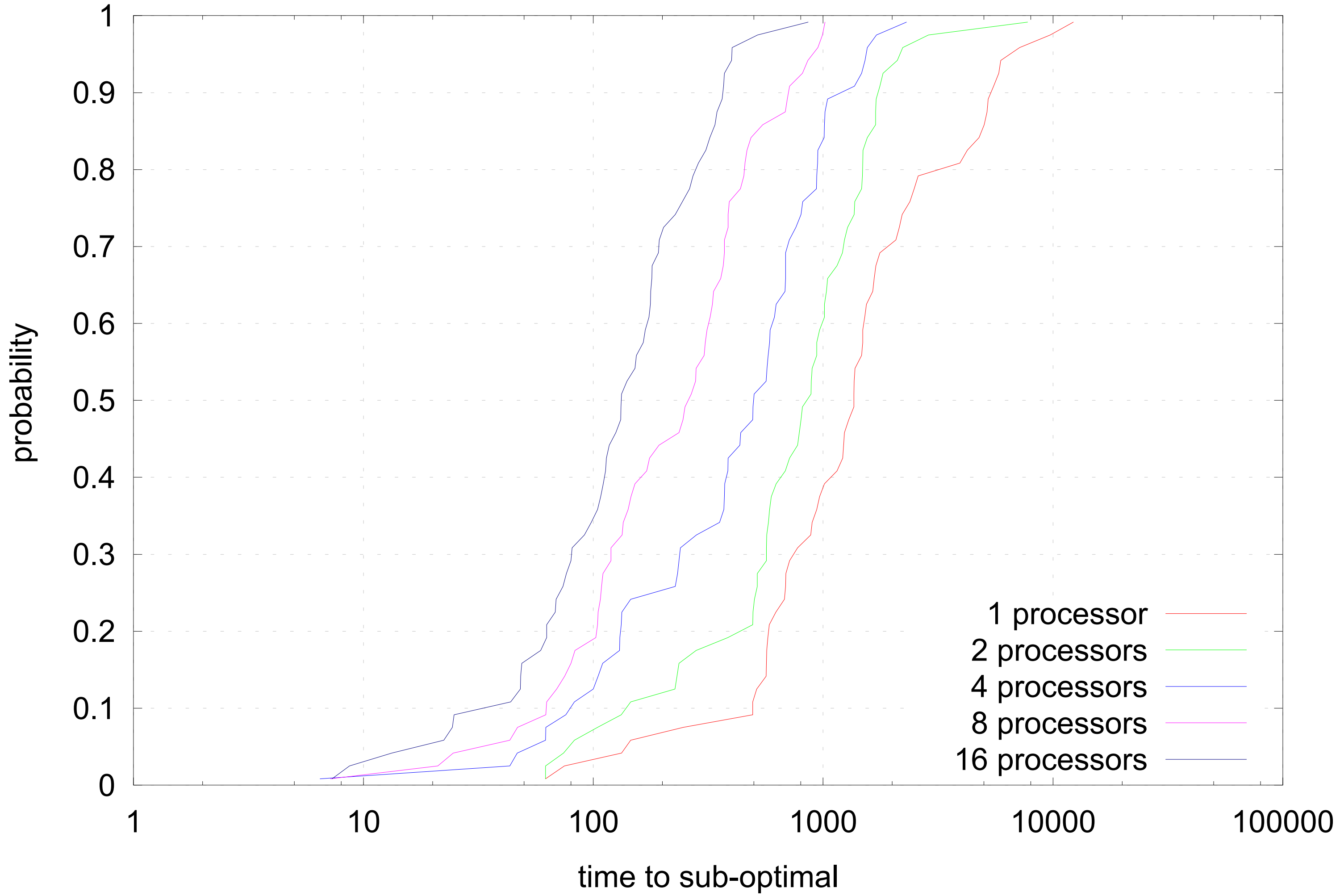


look4 = 7

60 independent runs of each algorithm.

MPI implementation.

Balas & Saltzman 26.1

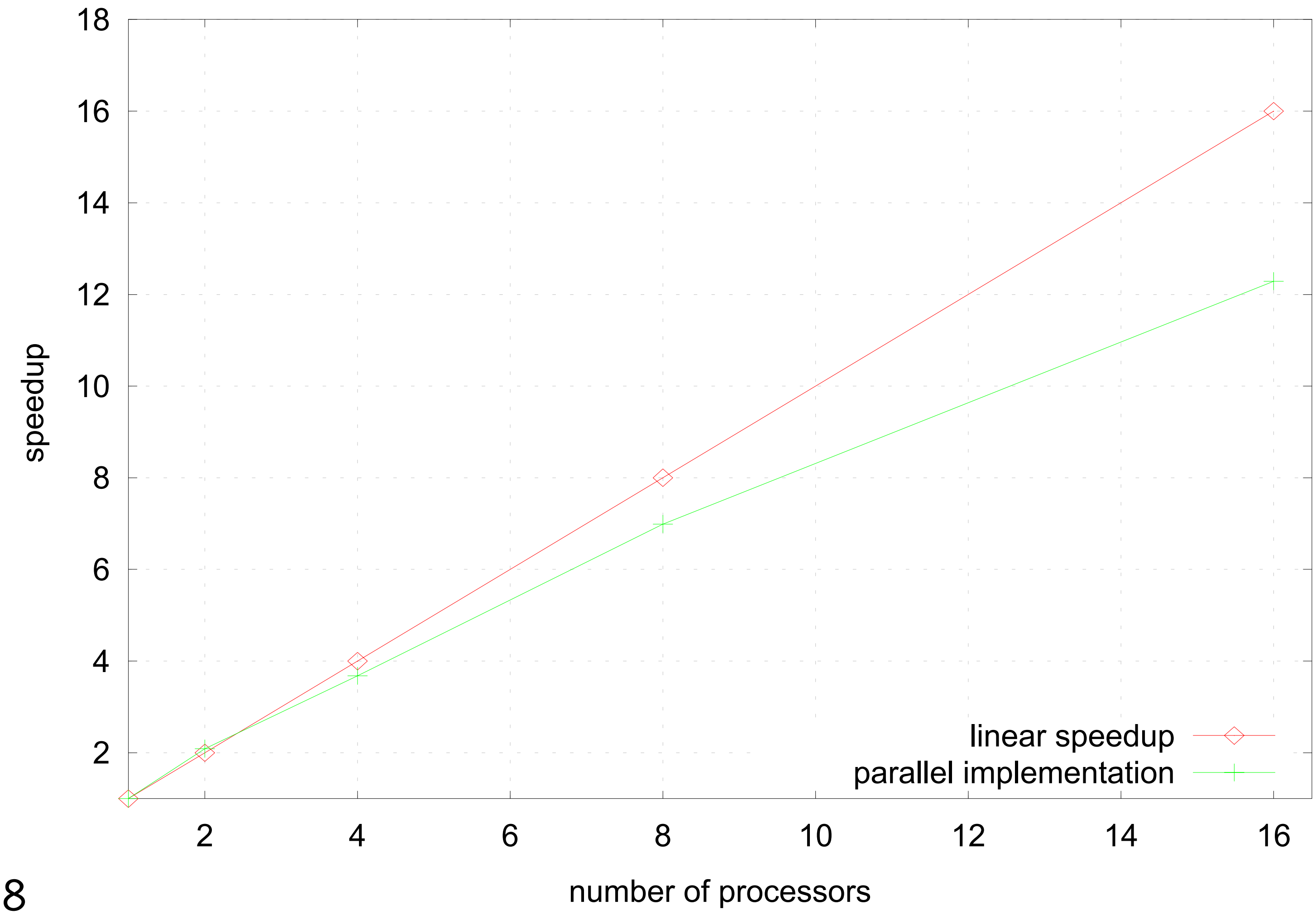


look4 = 8

Average speedup of 60 independent runs.

MPI implementation.

Balas & Saltzman 26.1



look4 = 8

Concluding remarks

- We show that memory mechanisms using path relinking improve performance of GRASP.
- Sophistication pays off: faster and better.
- Running time is exponentially distributed and parallel implementations enjoy good speedup.
- We have recently implemented a parallel algorithm with collaborating elite sets and observe super-linear speedup.
- Paper is available at http://www.research.att.com/~mgcr