GRASP with path-relinking for the quadratic assignment problem

Mauricio G.C. Resende, AT&T Research

joint work with:

Carlos A.S. Oliveira, U. of Florida

Panos M. Pardalos, U. of Florida

Talk given at WEA 2004

Angra dos Reis, Brazil

May 2004

Summary

- The quadratic assignment problem (QAP)
- GRASP for QAP
- Path-relinking for QAP
- Computational results
- Concluding remarks

Summary

- The quadratic assignment problem (QAP)
- GRASP for QAP
- Path-relinking for QAP
- Computational results
- Concluding remarks

Summary

- The quadratic assignment problem (QAP)
- GRASP for QAP
- Path-relinking for QAP
- Computational results
- Concluding remarks

Summary

- The quadratic assignment problem (QAP)
- GRASP for QAP
- Path-relinking for QAP
- Computational results
- Concluding remarks

Summary

- The quadratic assignment problem (QAP)
- GRASP for QAP
- Path-relinking for QAP
- Computational results
- Concluding remarks

Quadratic assignment problem (QAP)

- Given N facilities $f_1, f_2, \ldots, f_N$ and N locations $l_1, l_2, \ldots, l_N$
- Let $A^{N \times N} = (a_{i,j})$ be a positive real matrix where $a_{i,j}$ is the flow between facilities f_i and f_j
- Let $B^{N \times N} = (b_{i,j})$ be a positive real matrix where $b_{i,j}$ is the distance between locations l_i and l_j

Quadratic assignment problem (QAP)

- Given N facilities $f_1, f_2, \ldots, f_N$ and N locations $l_1, l_2, \ldots, l_N$
- Let $A^{N \times N} = (a_{i,j})$ be a positive real matrix where $a_{i,j}$ is the flow between facilities f_i and f_j
- Let $B^{N \times N} = (b_{i,j})$ be a positive real matrix where $b_{i,j}$ is the distance between locations l_i and l_j

Quadratic assignment problem (QAP)

- Given N facilities $f_1,f_2,\ldots,f_N$ and N locations $l_1,l_2,\ldots,l_N$
- Let $A^{N \times N} = (a_{i,j})$ be a positive real matrix where $a_{i,j}$ is the flow between facilities f_i and f_j
- Let $B^{N \times N} = (b_{i,j})$ be a positive real matrix where $b_{i,j}$ is the distance between locations l_i and l_j

Quadratic assignment problem (QAP)

- Let $p: \{1,2,\ldots,N\} \rightarrow \{1,2,\ldots,N\}$ be an assignment of the N facilities to the N locations
- Define the cost of assignment p to be

$$c(p) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{i,j} b_{p(i),p(j)}$$

- QAP: Find a permutation vector $p \in \prod_N$ that minimizes the assignment cost:

$$\min c(p): \text{subject to } p \in \prod_N$$

Quadratic assignment problem (QAP)

- Let p: {1,2,...,N} $\rightarrow${1,2,...,N} be an assignment of the N facilities to the N locations
- Define the cost of assignment p to be

$$c(p) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{i,j} b_{p(i),p(j)}$$

- QAP: Find a permutation vector $p \in \prod_N$ that minimizes the assignment cost:

$$\min c(p): \text{subject to } p \in \prod_N$$

Quadratic assignment problem (QAP)

- Let p: $\{1,2,\ldots,N\} \rightarrow \{1,2,\ldots,N\}$ be an assignment of the N facilities to the N locations
- Define the cost of assignment p to be

$$c(p) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{i,j} b_{p(i),p(j)}$$

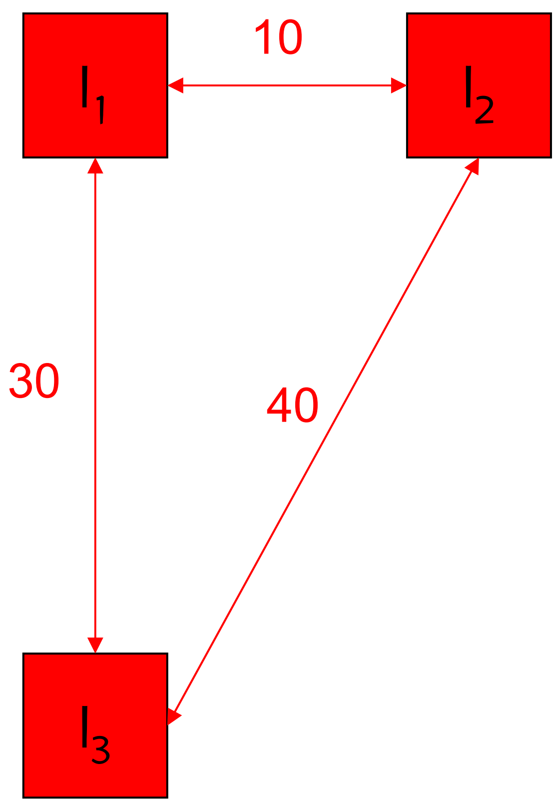
- QAP: Find a permutation vector $p \in \prod_N$ that minimizes the assignment cost:

$$\min c(p): \text{subject to } p \in \prod_N$$

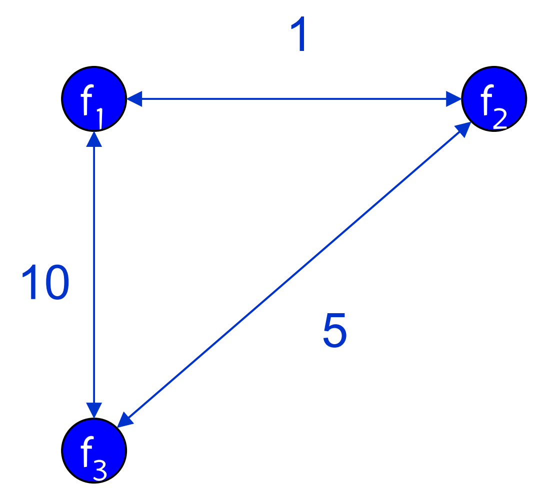
AT&T

Quadratic assignment problem (QAP)

cost of assignment: 10×1+ 30×10 + 40×5 = 510

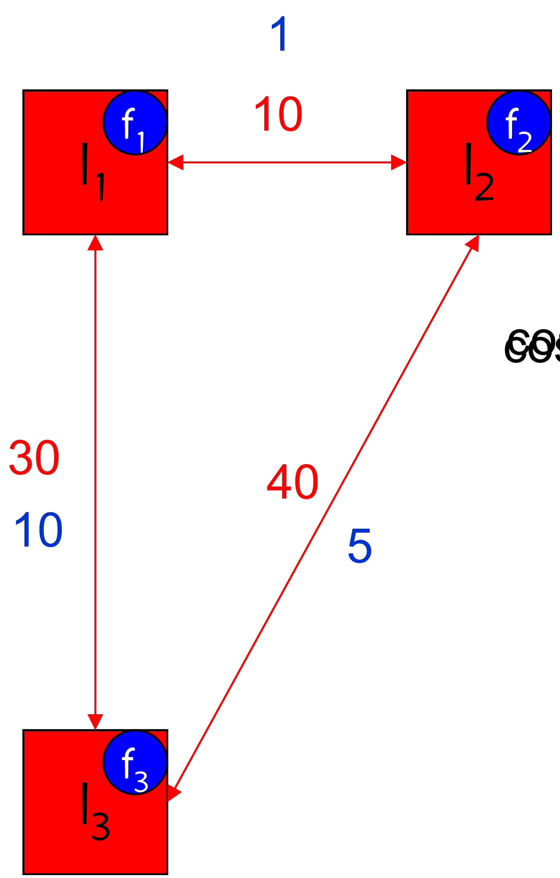


locations and distances



facilities and flows

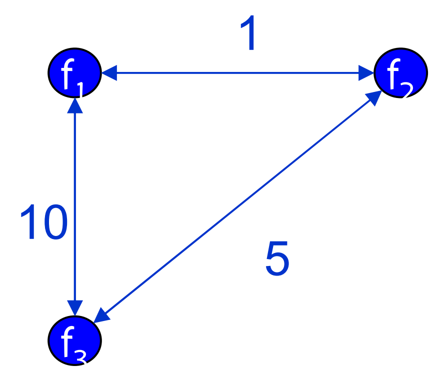
Quadratic assignment problem (QAP)



swap locations of facilities f_2 and f_3

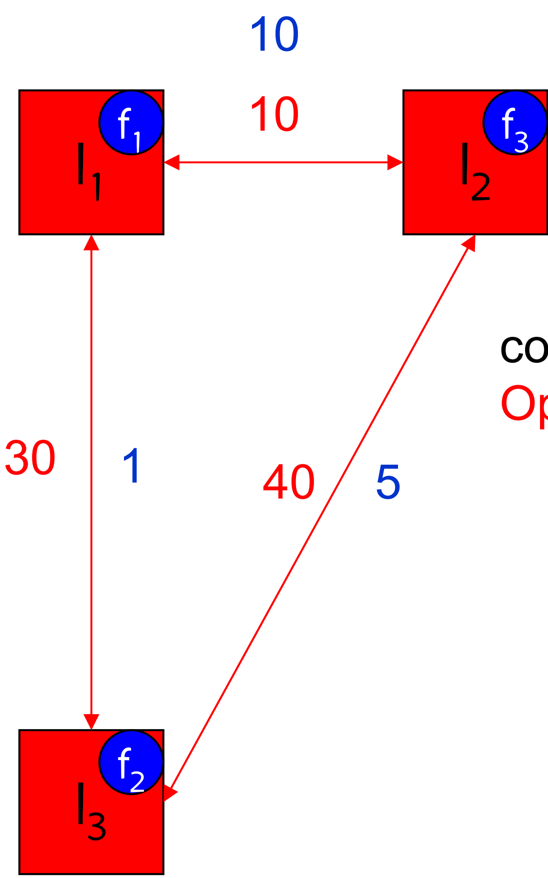
cost of assignment: $10\times1 + 30\times10 + 40\times5 = 510$

cost of assignment: $10\times10 + 30\times1 + 40\times5 = 330$



facilities and flows

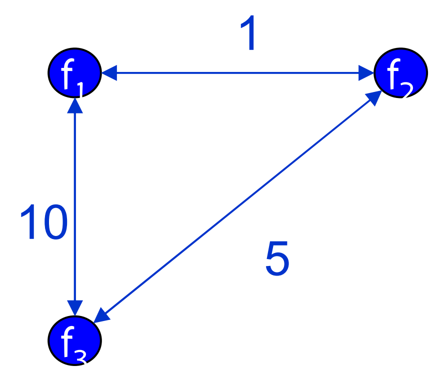
Quadratic assignment problem (QAP)



swap locations of facilities f_1 and f_3

cost of assignment: $10\times10+ 30\times5 + 40\times1 = 290$

Optimal!



facilities and flows

GRASP for QAP

- GRASP ✷ multi-start metaheuristic: greedy randomized construction, followed by local search (Feo & Resende, 1989, 1995; Festa & Resende, 2002; Resende & Ribeiro, 2003)
- GRASP for QAP
  - Li, Pardalos, & Resende (1994): GRASP for QAP
  - Resende, Pardalos, & Li (1996): Fortran suboroutines for dense QAPs
  - Pardalos, Pitsoulis, & Resende (1997): Fortran subroutines for sparse QAPs
  - Fleurent & Glover (1999): memory mechanism in construction

GRASP for QAP

- GRASP ✷ multi-start metaheuristic: greedy randomized construction, followed by local search (Feo & Resende, 1989, 1995; Festa & Resende, 2002; Resende & Ribeiro, 2003)
- GRASP for QAP
  - Li, Pardalos, & Resende (1994): GRASP for QAP
  - Resende, Pardalos, & Li (1996): Fortran subroutines for dense QAPs
  - Pardalos, Pitsoulis, & Resende (1997): Fortran subroutines for sparse QAPs
  - Fleurent & Glover (1999): memory mechanism in construction

AT&T

GRASP for QAP

```
repeat {
    x = GreedyRandomizedConstruction(▪);
    x = LocalSearch(x);
    save x as x* if best so far;
}
return x*;
```

Construction

- Stage 1: make two assignments $\{f_i \rightarrow l_k ; f_j \rightarrow l_l\}$
- Stage 2: make remaining N−2 assignments of facilities to locations, one facility/location pair at a time

Construction

- Stage 1: make two assignments $\{f_i \rightarrow l_k \,;\, f_j \rightarrow l_l\}$
- Stage 2: make remaining N–2 assignments of facilities to locations, one facility/location pair at a time

Stage 1 construction

- sort distances $b_{i,j}$ in increasing order:
  $b_{i(1),j(1)} \leq b_{i(2),j(2)} \leq \cdot \cdot \cdot \leq b_{i(N),j(N)}$.
- sort flows $a_{k,l}$ in decreasing order:
  $a_{k(1),l(1)} \geq a_{k(2),l(2)} \geq \cdot \cdot \cdot \geq a_{k(N),l(N)}$.
- sort products:
  $a_{k(1),l(1)} \cdot b_{i(1),j(1)},\ a_{k(2),l(2)} \cdot b_{i(2),j(2)},\ \ldots,\ a_{k(N),l(N)} \cdot b_{i(N),j(N)}$
- among smallest products, select $a_{k(q),l(q)} \cdot b_{i(q),j(q)}$ at random:
  corresponding to assignments $\{f_{k(q)} \rightarrow l_{i(q)}\ ;\ f_{l(q)} \rightarrow l_{j(q)}\}$

AT&T

Stage 1 construction

- sort distances $b_{i,j}$ in increasing order: $b_{i(1),j(1)} \leq b_{i(2),j(2)} \leq \cdots \leq b_{i(N),j(N)}$.
- sort flows $a_{k,l}$ in decreasing order: $a_{k(1),l(1)} \geq a_{k(2),l(2)} \geq \cdots \geq a_{k(N),l(N)}$.
- sort products: $a_{k(1),l(1)} \cdot b_{i(1),j(1)},\ a_{k(2),l(2)} \cdot b_{i(2),j(2)},\ \ldots,\ a_{k(N),l(N)} \cdot b_{i(N),j(N)}$
- among smallest products, select $a_{k(q),l(q)} \cdot b_{i(q),j(q)}$ at random: corresponding to assignments $\{f_{k(q)} \rightarrow l_{i(q)}\ ;\ f_{l(q)} \rightarrow l_{j(q)}\}$

Stage 1 construction

- sort distances $b_{i,j}$ in increasing order:
  $b_{i(1),j(1)} \leq b_{i(2),j(2)} \leq \cdots \leq b_{i(N),j(N)}$.
- sort flows $a_{k,l}$ in decreasing order:
  $a_{k(1),l(1)} \geq a_{k(2),l(2)} \geq \cdots \geq a_{k(N),l(N)}$.
- sort products:
  $a_{k(1),l(1)} \cdot b_{i(1),j(1)}, a_{k(2),l(2)} \cdot b_{i(2),j(2)}, \ldots, a_{k(N),l(N)} \cdot b_{i(N),j(N)}$
- among smallest products, select $a_{k(q),l(q)} \cdot b_{i(q),j(q)}$ at random: corresponding to assignments $\{f_{k(q)} \rightarrow l_{i(q)} ; f_{l(q)} \rightarrow l_{j(q)}\}$

AT&T

Stage 1 construction

- sort distances $b_{i,j}$ in increasing order:
  $b_{i(1),j(1)} \leq b_{i(2),j(2)} \leq \cdots \leq b_{i(N),j(N)}$.
- sort flows $a_{k,l}$ in decreasing order:
  $a_{k(1),l(1)} \geq a_{k(2),l(2)} \geq \cdots \geq a_{k(N),l(N)}$.
- sort products:
  $a_{k(1),l(1)} \cdot b_{i(1),j(1)},\ a_{k(2),l(2)} \cdot b_{i(2),j(2)},\ \ldots,\ a_{k(N),l(N)} \cdot b_{i(N),j(N)}$
- among smallest products, select $a_{k(q),l(q)} \cdot b_{i(q),j(q)}$ at random: corresponding to assignments $\{f_{k(q)} \rightarrow l_{i(q)}\ ;\ f_{l(q)} \rightarrow l_{j(q)}\}$

Stage 2 construction

- If $\Omega = \{(i_1,k_1),(i_2,k_2), \ldots, (i_q,k_q)\}$ are the q assignments made so far, then
- Cost of assigning $f_j \rightarrow l_l$ is $c_{j,l} = \sum_{i,k \in \Gamma} a_{i,j} b_{k,l}$
- Of all possible assignments, one is selected at random from the assignments having smallest costs and is added to Ω

AT&T

Stage 2 construction

- If $\Omega = \{(i_1,k_1),(i_2,k_2), \ldots, (i_q,k_q)\}$ are the q assignments made so far, then
- Cost of assigning $f_j \rightarrow l_l$ is $$c_{j,l} = \sum_{i,k \in \Gamma} a_{i,j} b_{k,l}$$
- Of all possible assignments, one is selected at random from the assignments having smallest costs and is added to Ω

Stage 2 construction

- If $\Omega = \{(i_1,k_1),(i_2,k_2), \ldots, (i_q,k_q)\}$ are the q assignments made so far, then
- Cost of assigning $f_j \rightarrow l_l$ is $c_{j,l} = \sum_{i,k \in \Gamma} a_{i,j} b_{k,l}$

- Of all possible assignments, one is selected at random from the assignments having smallest costs and is added to Ω

Sped up in Pardalos, Pitsoulis, & Resende (1997) for QAPs with sparse A or B matrices.

Swap based local search

a) For all pairs of assignments $\{f_i \rightarrow l_k\ ;\ f_j \rightarrow l_l\}$, test if swapped assignment $\{f_i \rightarrow l_l\ ;\ f_j \rightarrow l_k\}$ improves solution.

b) If so, make swap and return to step (a)

Swap based local search

a) For all pairs of assignments $\{f_i \rightarrow l_k \,;\, f_j \rightarrow l_l\}$, test if swapped assignment $\{f_i \rightarrow l_l \,;\, f_j \rightarrow l_k\}$ improves solution.

b) If so, make swap and return to step (a)

repeat (a)-(b) until no swap improves current solution

Path-relinking

- Path-relinking:
  - Intensification strategy exploring trajectories connecting elite solutions: Glover (1996)
  - Originally proposed in the context of tabu search and scatter search.
  - Paths in the solution space leading to other elite solutions are explored in the search for better solutions:
    - selection of moves that introduce attributes of the guiding solution into the current solution

Path-relinking

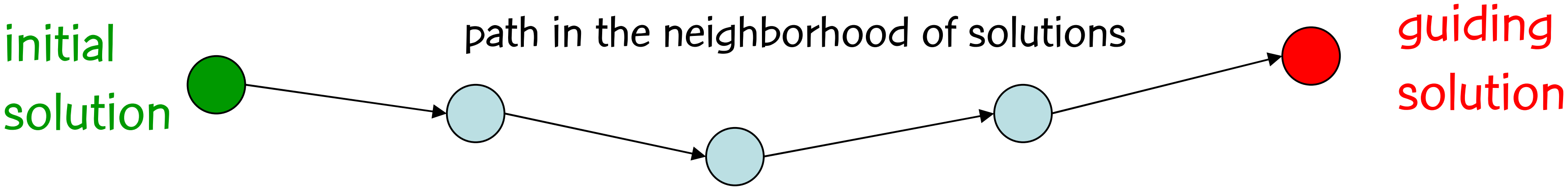
- Path-relinking:
  - Intensification strategy exploring trajectories connecting elite solutions: Glover (1996)
  - Originally proposed in the context of tabu search and scatter search.
  - Paths in the solution space leading to other elite solutions are explored in the search for better solutions:
    - selection of moves that introduce attributes of the guiding solution into the current solution

Path-relinking

- Path-relinking:
  - Intensification strategy exploring trajectories connecting elite solutions: Glover (1996)
  - Originally proposed in the context of tabu search and scatter search.
  - Paths in the solution space leading to other elite solutions are explored in the search for better solutions:
    - selection of moves that introduce attributes of the guiding solution into the current solution

Path-relinking

- Exploration of trajectories that connect high quality (elite) solutions:

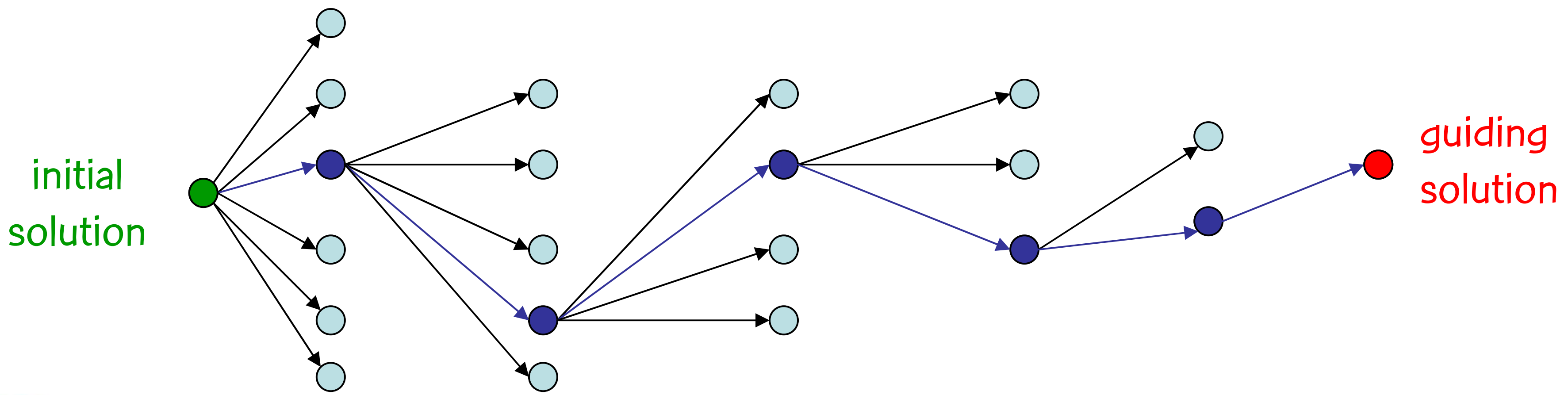


Path-relinking

- 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 evaluated and the best move is selected:

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 evaluated and the best move is selected:



Path-relinking

Combine solutions x and y

$\Delta(x,y)$: symmetric difference between x and y

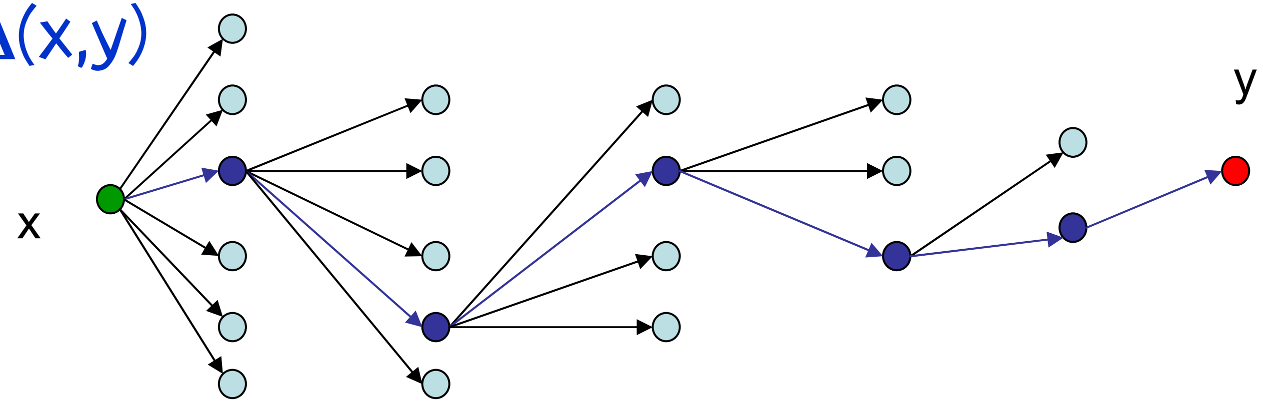
while ($|\Delta(x,y)| > 0$) {

evaluate moves corresponding in $\Delta(x,y)$

make best move

update $\Delta(x,y)$

}



GRASP with path-relinking

- Originally used by Laguna and Martí (1999).
- Maintains a set of elite solutions found during GRASP iterations.
- After each GRASP iteration (construction and local search):
  - Use GRASP solution as initial solution.
  - Select an elite solution uniformly at random: guiding solution.
  - Perform path-relinking between these two solutions.

GRASP with path-relinking

- Originally used by Laguna and Martí (1999).
- Maintains a set of elite solutions found during GRASP iterations.
- After each GRASP iteration (construction and local search):
  - Use GRASP solution as initial solution.
  - Select an elite solution uniformly at random: guiding solution.
  - Perform path-relinking between these two solutions.

GRASP with path-relinking

- Originally used by Laguna and Martí (1999).
- Maintains a set of elite solutions found during GRASP iterations.
- After each GRASP iteration (construction and local search):
  - Use GRASP solution as initial solution.
  - Select an elite solution uniformly at random: guiding solution.
  - Perform path-relinking between these two solutions.

GRASP with path-relinking

Repeat for Max_Iterations:

Construct a greedy randomized solution.

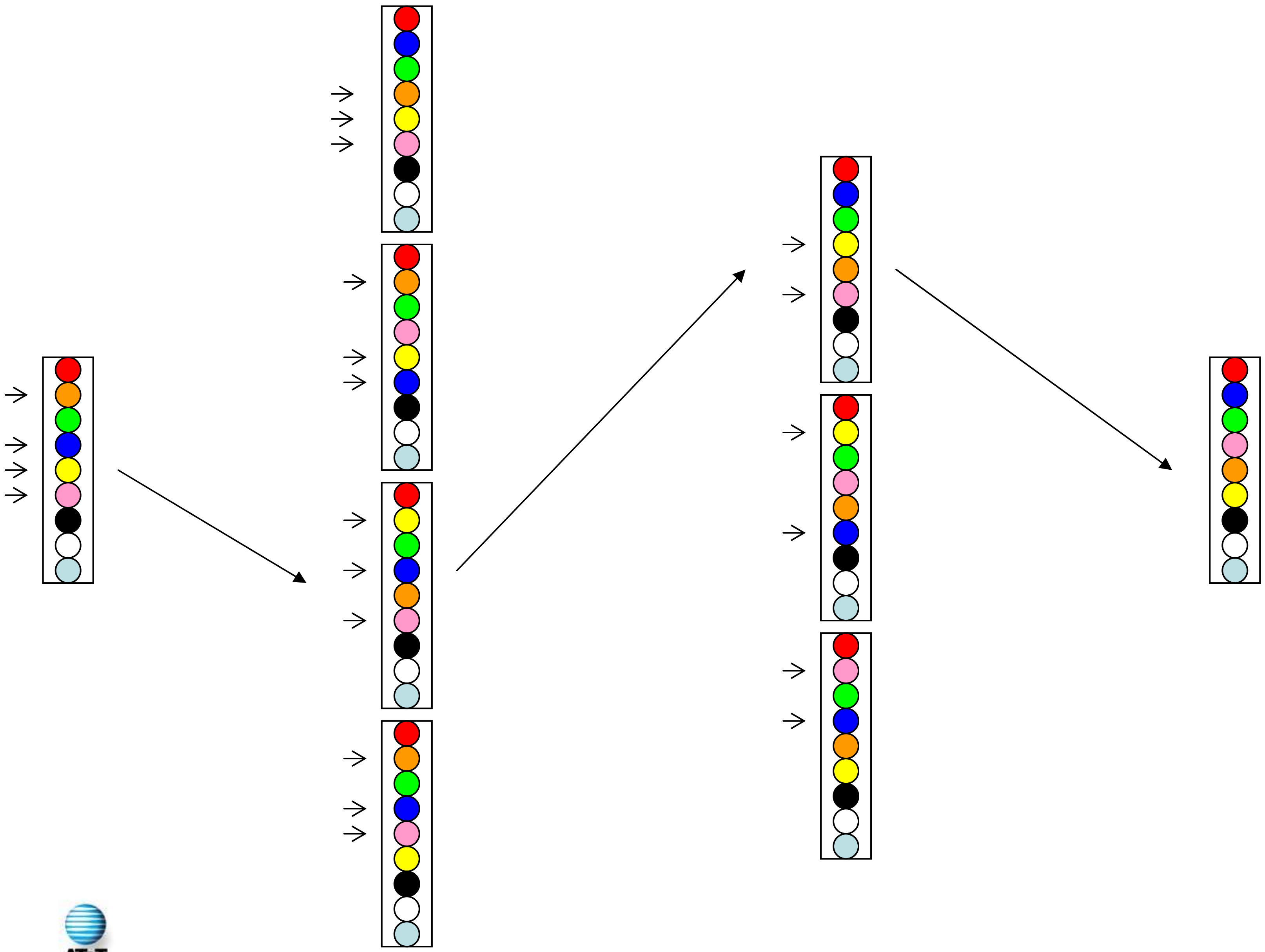
Use local search to improve the constructed solution.

Apply path-relinking to further improve the solution.

Update the pool of elite solutions.

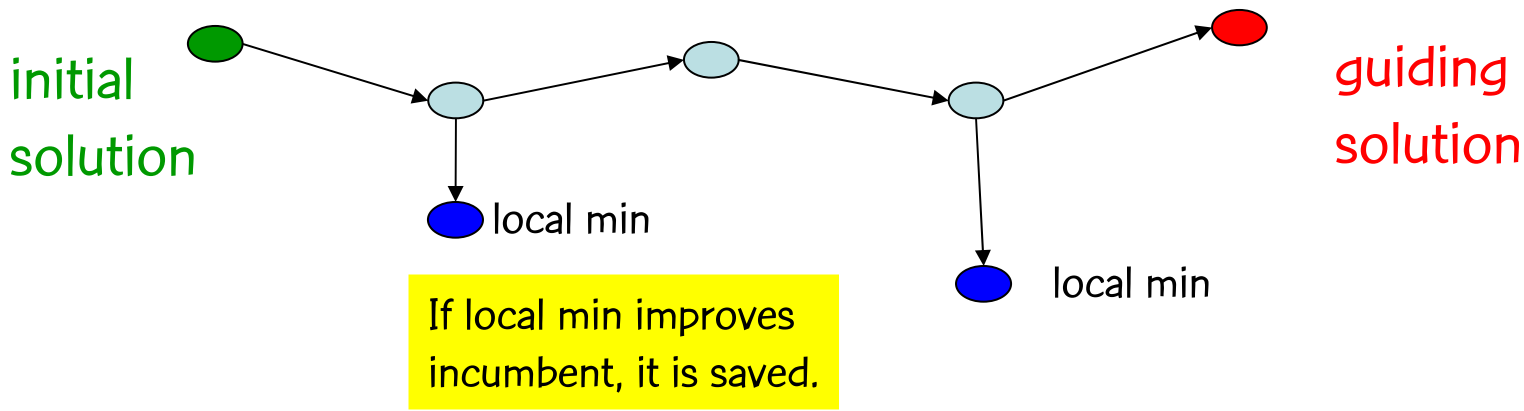
Update the best solution found.

PR for QAP (permutation vectors)



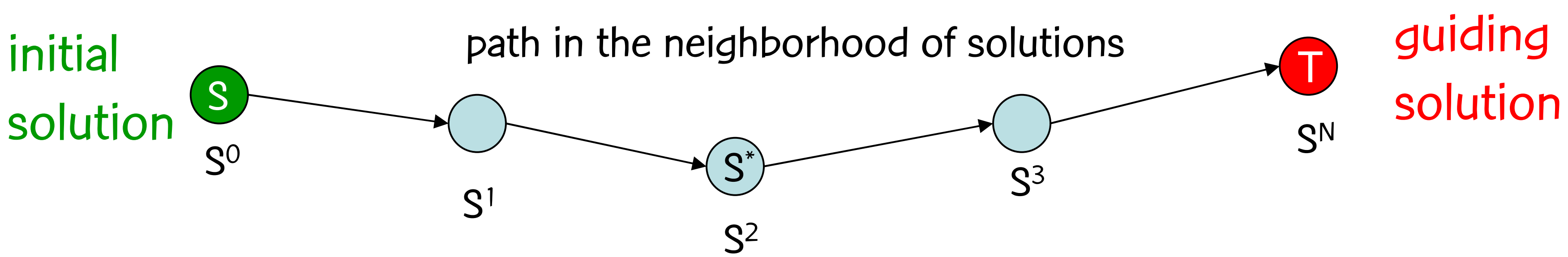
Path-relinking for QAP

If swap improves solution: local search is applied



Path-relinking for QAP

Results of path relinking: S^*

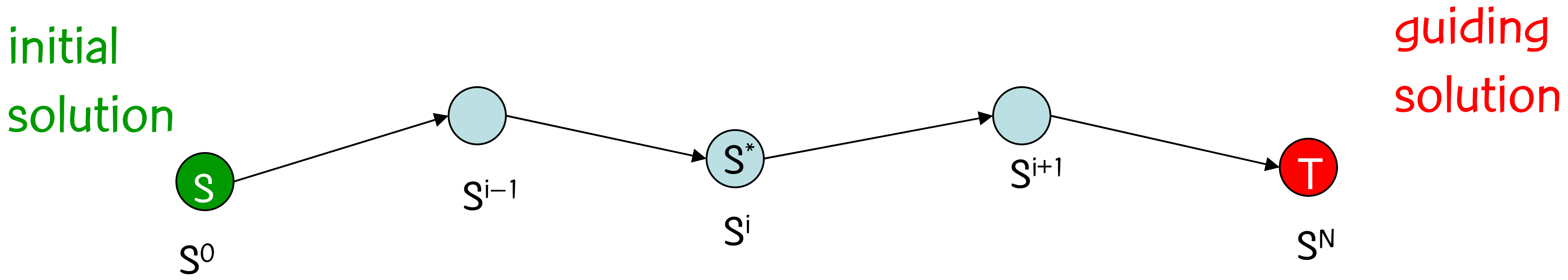


If $c(S^*) < \min \{c(S), c(T)\}$, and $c(S^*) \leq c(S^i)$, for $i=1,\ldots,N$, i.e. S^* is best solution in path, then S^* is returned.

Path-relinking for QAP

S^i is a local minimum w.r.t. PR:

$c(S^i) < c(S^{i-1})$ and $c(S^i) < c(S^{i+1})$, for all $i=1,\ldots,N$.



If path-relinking does not improve (S,T), then if S^i is a best local min w.r.t. PR: return $S^* = S^i$

If no local min exists, return $S^*=\text{argmin}\{S,T\}$

PR pool management

- S^* is candidate for inclusion in pool of elite solutions (P)
- If $c(S^*) < c(S^e)$, for all $S^e \in P$, then S^* is put in P
- Else, if $c(S^*) < \max\{c(S^e), S^e \in P\}$ and $|\Delta(S^*,S^e)| \geq 3$, for all $S^e \in P$, then S^* is put in P
- If pool is full, remove $\operatorname{argmin} \{|\Delta(S^*,S^e)|, \forall S^e \in P \text{ s.t. } c(S^e) \geq c(S^*)\}$

AT&T

PR pool management

- S^* is candidate for inclusion in pool of elite solutions (P)
- If $c(S^*) < c(S^e)$, for all $S^e \in P$, then S^* is put in P
- Else, if $c(S^*) < \max\{c(S^e), S^e \in P\}$ and $|\Delta(S^*,S^e)| \geq 3$, for all $S^e \in P$, then S^* is put in P
- If pool is full, remove $\operatorname{argmin} \{|\Delta(S^*,S^e)|, \forall S^e \in P \text{ s.t. } c(S^e) \geq c(S^*)\}$

PR pool management

- S^* is candidate for inclusion in pool of elite solutions (P)
- If $c(S^*) < c(S^e)$, for all $S^e \in P$, then S^* is put in P
- Else, if $c(S^*) < \max\{c(S^e), S^e \in P\}$ and $|\Delta(S^*,S^e)| \geq 3$, for all $S^e \in P$, then S^* is put in P
- If pool is full, remove $\operatorname{argmin} \{|\Delta(S^*,S^e)|, \forall S^e \in P \text{ s.t. } c(S^e) \geq c(S^*)\}$

PR pool management

- S^* is candidate for inclusion in pool of elite solutions (P)
- If $c(S^*) < c(S^e)$, for all $S^e \in P$, then S^* is put in P
- Else, if $c(S^*) < \max\{c(S^e), S^e \in P\}$ and $|\Delta(S^*,S^e)| \geq 3$, for all $S^e \in P$, then S^* is put in P
- If pool is full, remove $\text{argmin}\ \{|\Delta(S^*,S^e)|, \forall\ S^e \in P \text{ s.t. } c(S^e) \geq c(S^*)\}$

AT&T

PR pool management

S is initial solution for path-relinking: favor choice of target solution T with large symmetric difference with S.

This leads to longer paths in path-relinking.

Probability of choosing $S^e \in P$:

$$p(S^e) = \frac{|\Delta(S, S^e)|}{\sum_{R \in P} |\Delta(S, R)|}$$

Experimental results

- Compare GRASP with and without path-relinking.
- New GRASP code in C outperforms old Fortran codes: we use same code to compare algorithms
- All QAPLIB (Burkhard, Karisch, & Rendl, 1991) instances of size $N \leq 40$
- 100 independent runs of each algorithm, recording CPU time to find the best known solution for instance

Experimental results

- Compare GRASP with and without path-relinking.
- New GRASP code in C outperforms old Fortran codes: we use same code to compare algorithms
- All QAPLIB (Burkhard, Karisch, & Rendl, 1991) instances of size $N \leq 40$
- 100 independent runs of each algorithm, recording CPU time to find the best known solution for instance

Experimental results

- Compare GRASP with and without path-relinking.
- New GRASP code in C outperforms old Fortran codes: we use same code to compare algorithms
- All QAPLIB (Burkhard, Karisch, & Rendl, 1991) instances of size $N \leq 40$
- 100 independent runs of each algorithm, recording CPU time to find the best known solution for instance

Experimental results

- Compare GRASP with and without path-relinking.
- New GRASP code in C outperforms old Fortran codes: we use same code to compare algorithms
- All QAPLIB (Burkhard, Karisch, & Rendl, 1991) instances of size $N \leq 40$
- 100 independent runs of each algorithm, recording CPU time to find the best known solution for instance

Experimental results

- SGI Challenge computer (196 MHz R10000 processors (28) and 7 Gb memory)
- Single processor used for each run
- GRASP RCL parameter α chosen at random in interval [0,1] at each GRASP iteration.
- Size of elite set: 30
- Path-relinking done in both directions (S to T to S)
- Care taken to ensure that GRASP and GRASP with path-relinking iterations are in sync

Experimental results

- SGI Challenge computer (196 MHz R10000 processors (28) and 7 Gb memory)
- Single processor used for each run
- GRASP RCL parameter α chosen at random in interval [0,1] at each GRASP iteration.
- Size of elite set: 30
- Path-relinking done in both directions (S to T to S)
- Care taken to ensure that GRASP and GRASP with path-relinking iterations are in sync

Experimental results

- SGI Challenge computer (196 MHz R10000 processors (28) and 7 Gb memory)
- Single processor used for each run
- GRASP RCL parameter α chosen at random in interval [0,1] at each GRASP iteration.
- Size of elite set: 30
- Path-relinking done in both directions (S to T to S)
- Care taken to ensure that GRASP and GRASP with path-relinking iterations are in sync

Experimental results

- SGI Challenge computer (196 MHz R10000 processors (28) and 7 Gb memory)
- Single processor used for each run
- GRASP RCL parameter α chosen at random in interval [0,1] at each GRASP iteration.
- Size of elite set: 30
- Path-relinking done in both directions (S to T to S)
- Care taken to ensure that GRASP and GRASP with path-relinking iterations are in sync

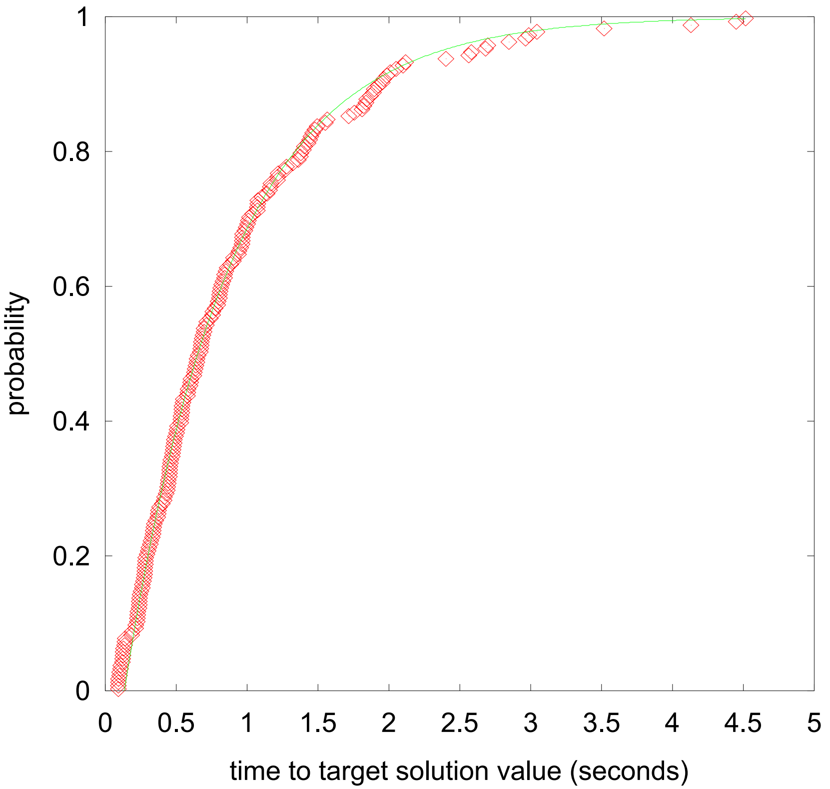
Experimental results

- SGI Challenge computer (196 MHz R10000 processors (28) and 7 Gb memory)
- Single processor used for each run
- GRASP RCL parameter α chosen at random in interval [0,1] at each GRASP iteration.
- Size of elite set: 30
- Path-relinking done in both directions (S to T to S)
- Care taken to ensure that GRASP and GRASP with path-relinking iterations are in sync

Experimental results

- SGI Challenge computer (196 MHz R10000 processors (28) and 7 Gb memory)
- Single processor used for each run
- GRASP RCL parameter α chosen at random in interval [0,1] at each GRASP iteration.
- Size of elite set: 30
- Path-relinking done in both directions (S to T to S)
- Care taken to ensure that GRASP and GRASP with path-relinking iterations are in sync

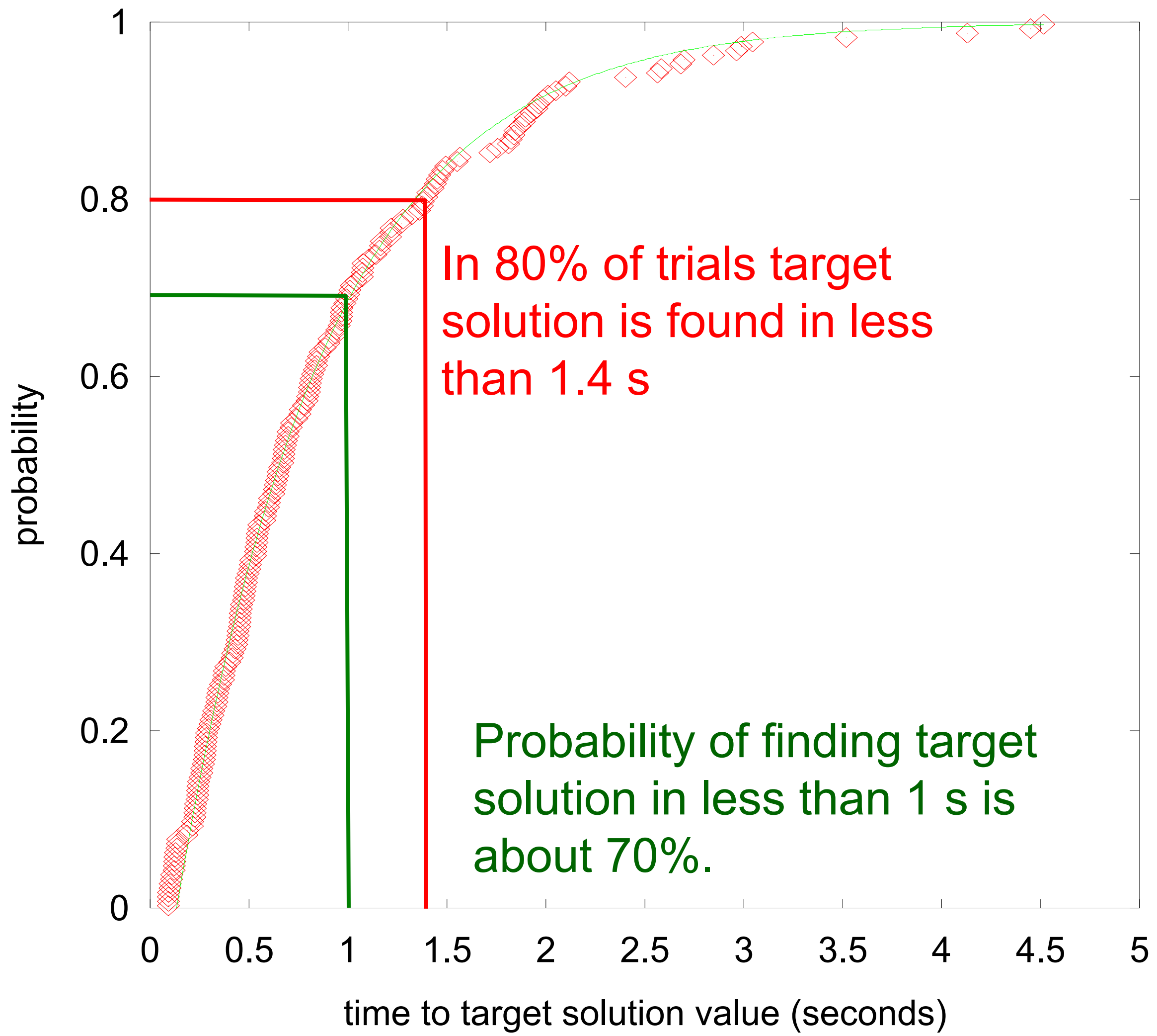
Time-to-target-value plots



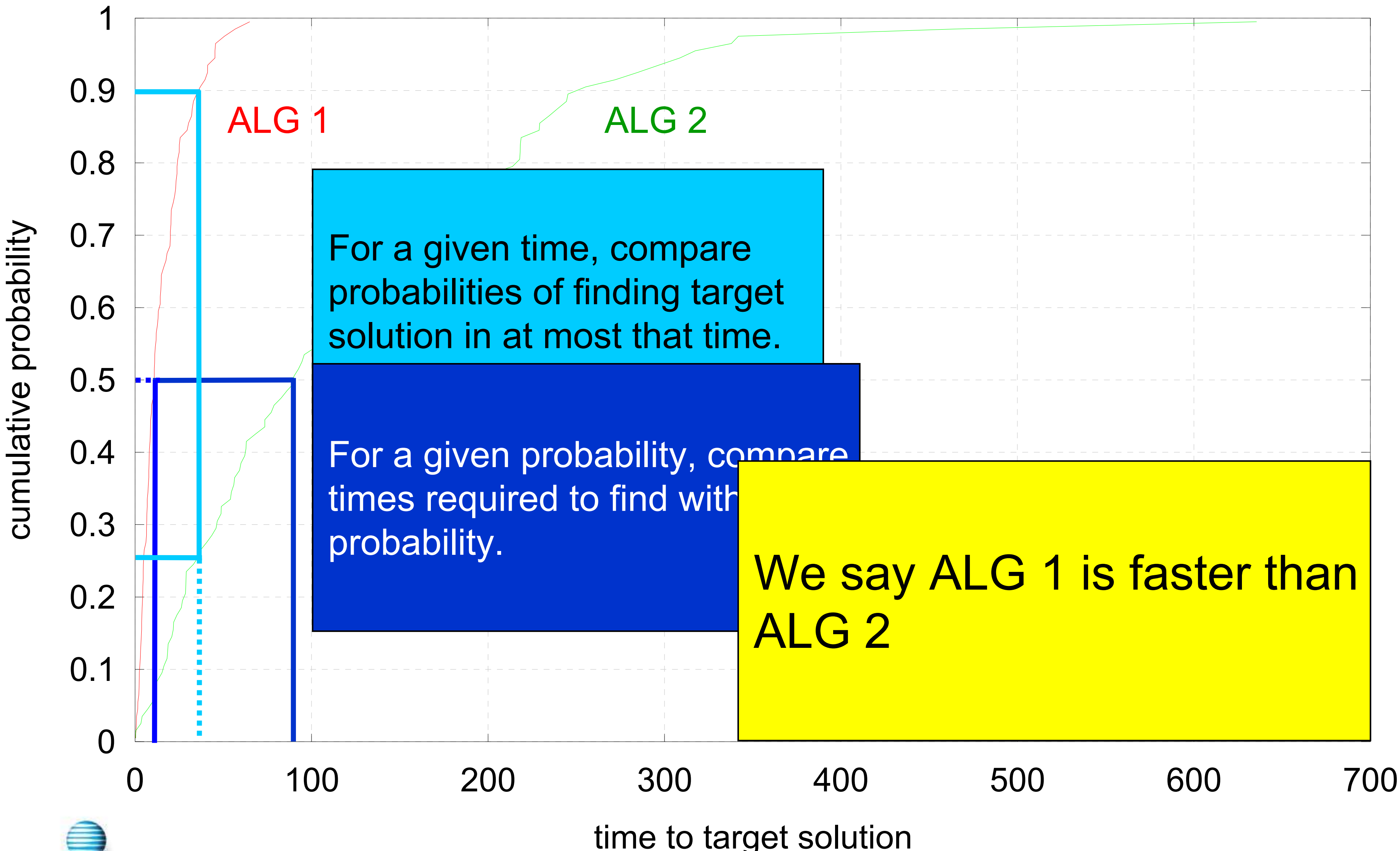
Sort times such that $t_1 \leq t_2 \leq \cdots \leq t_{100}$ and plot $\{t_i, p_i\}$, for $i=1,\ldots,N$, where $p_i = (i-.5)/100$

Random variable time-to-target-solution value fits a two-parameter exponential distribution (Aiex, Resende, & Ribeiro, 2002).

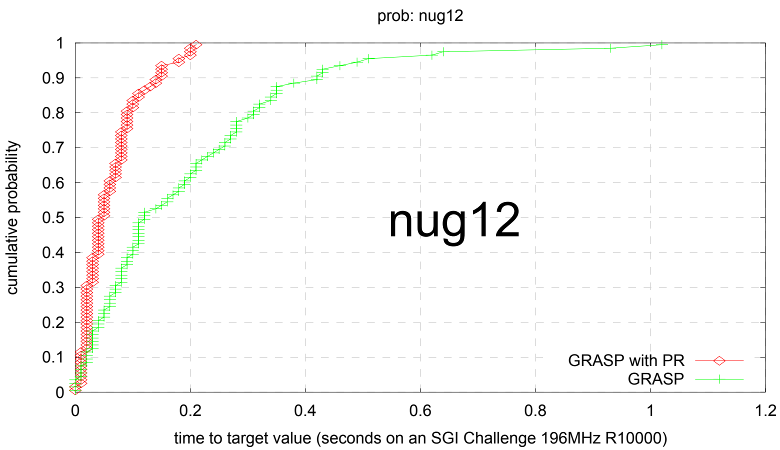
Time-to-target-value plots

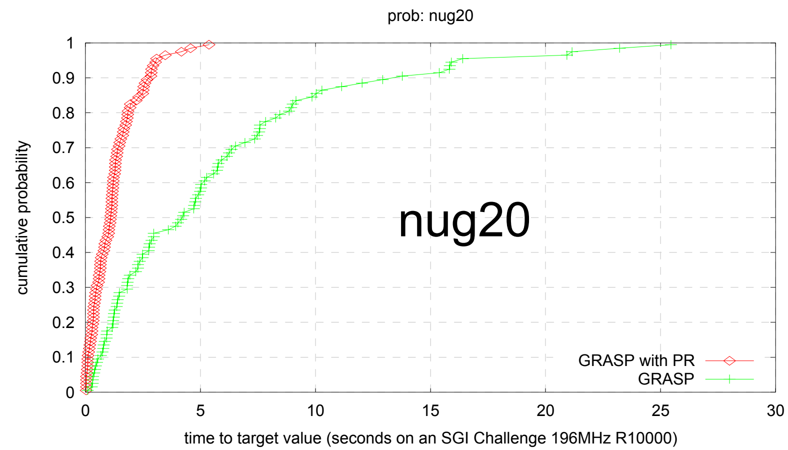


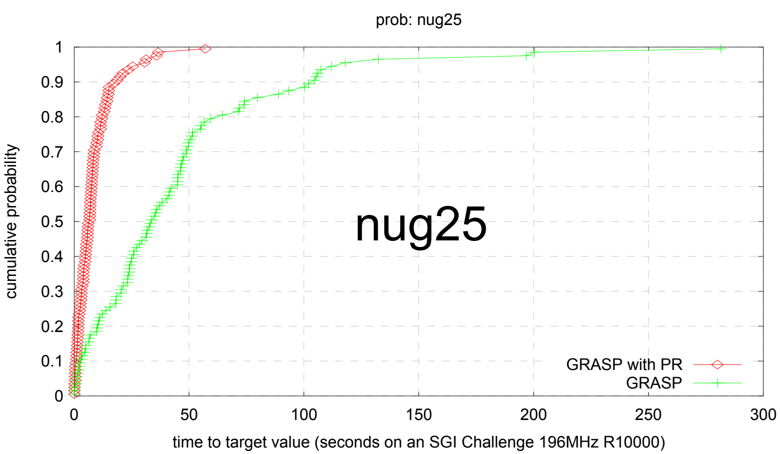
Time-to-target-value plots

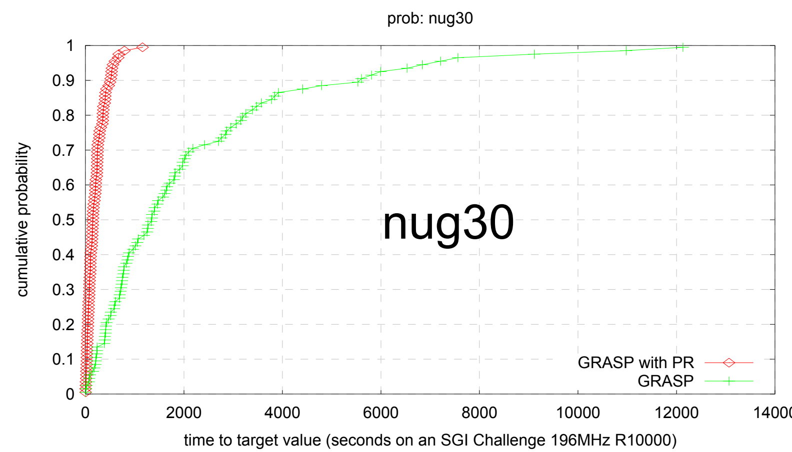


C.E. Nugent, T.E. Vollmann and J. Ruml [1968]

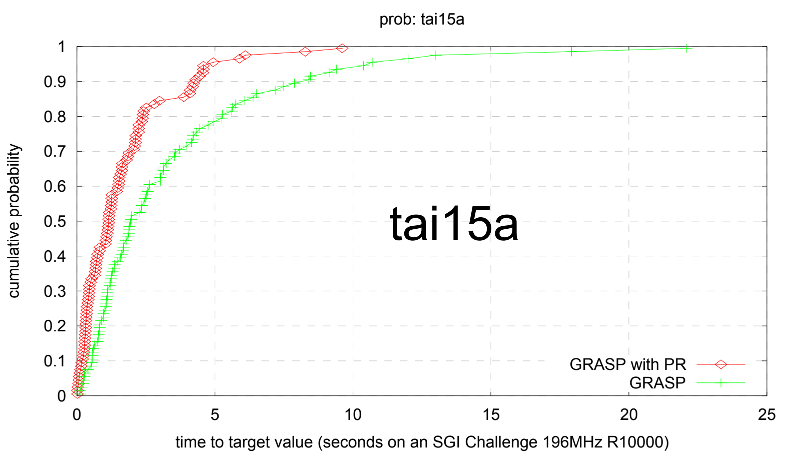


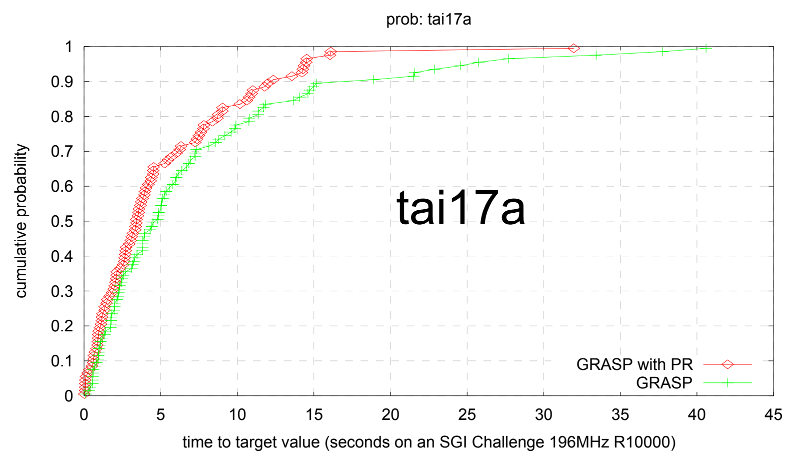


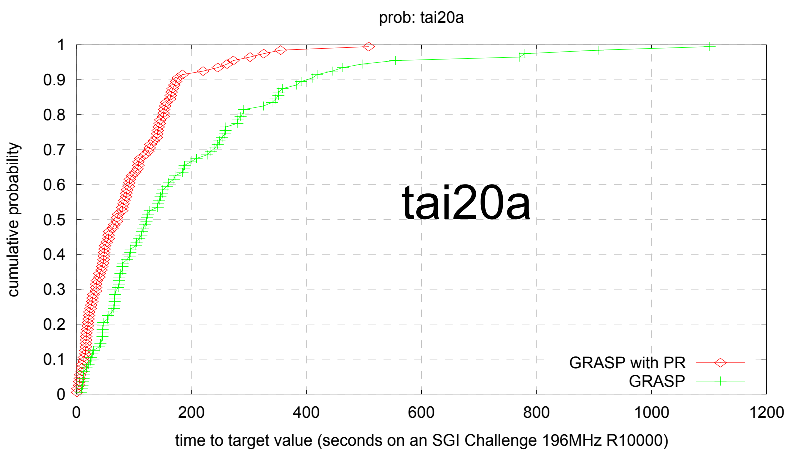


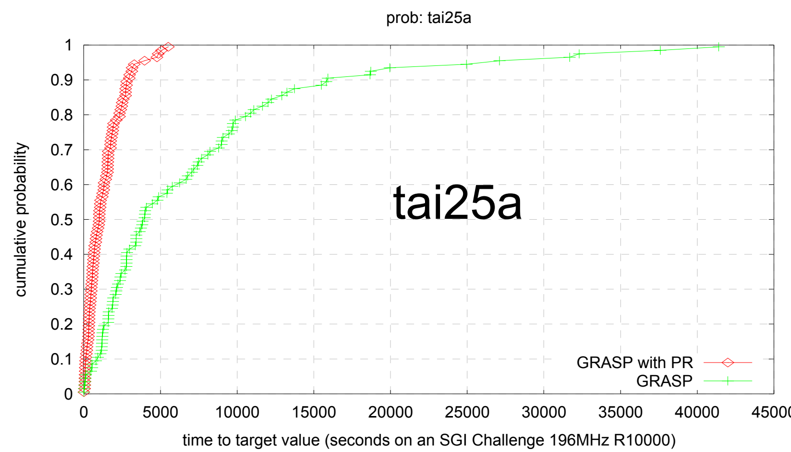


E.D. Taillard [1991, 1994]



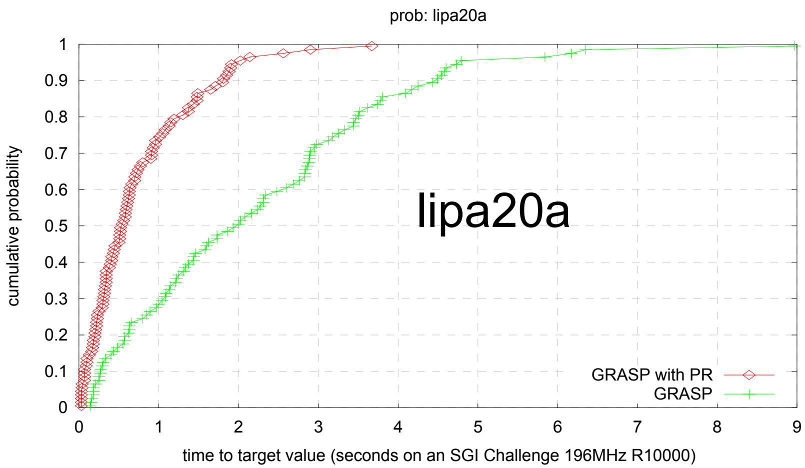


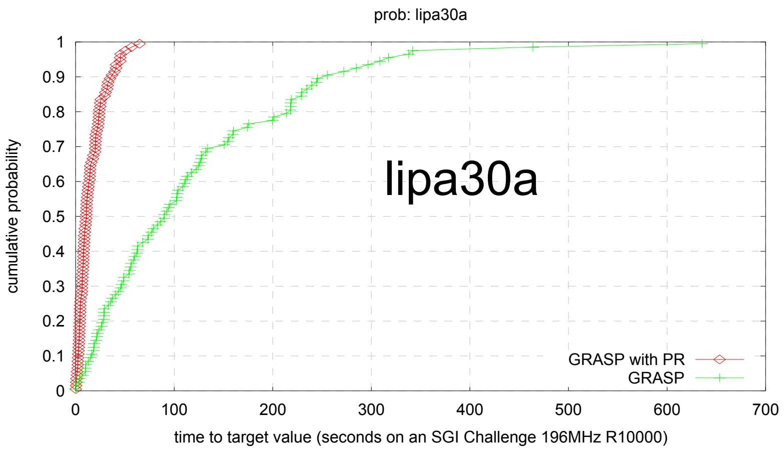


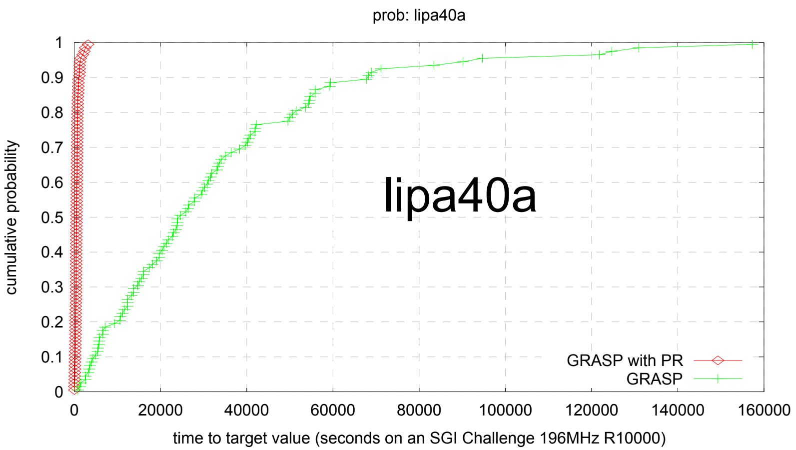


AT&T

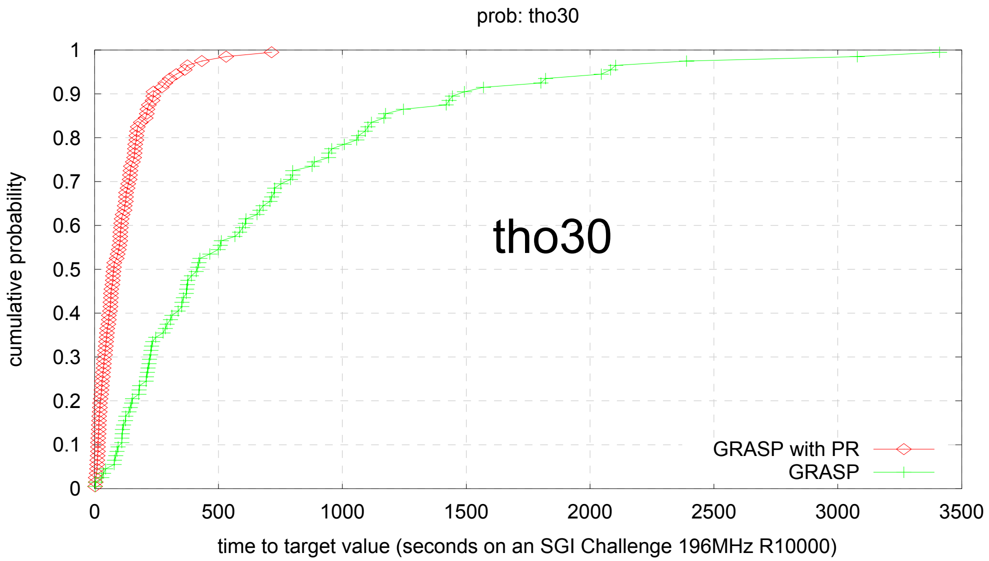
Y. Li and P.M. Pardalos [1992]

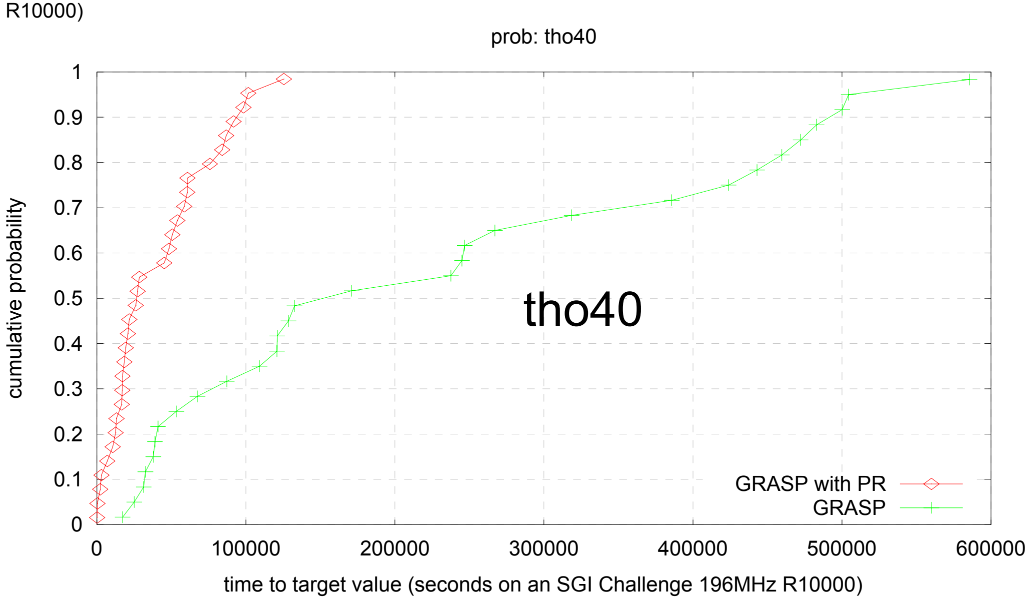




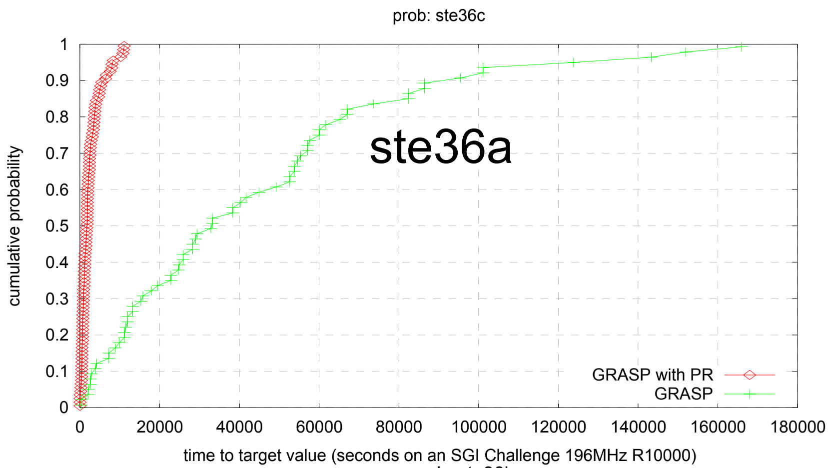


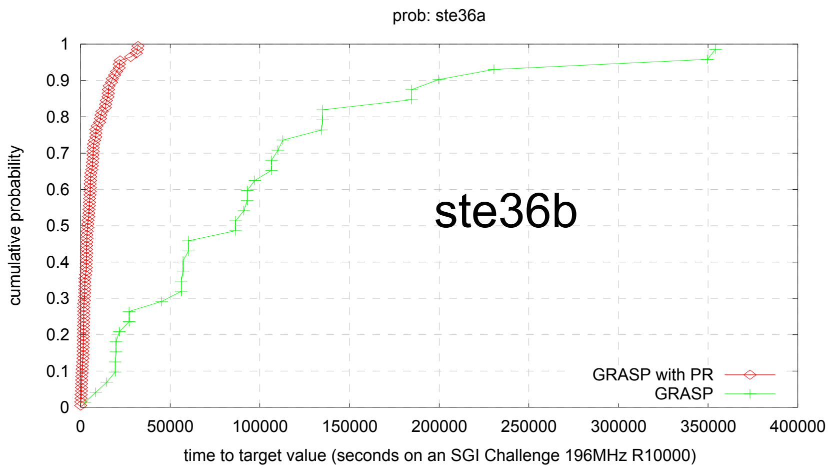
U.W. Thonemann and A. Bölte [1994]

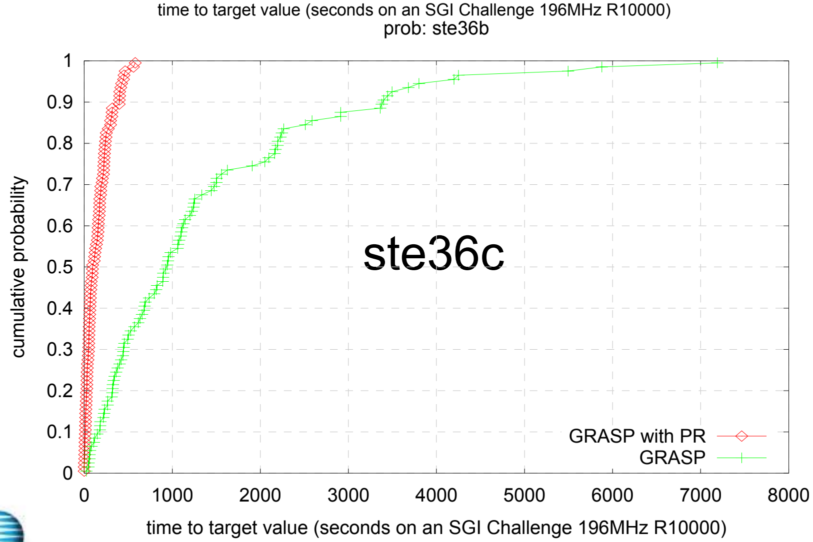




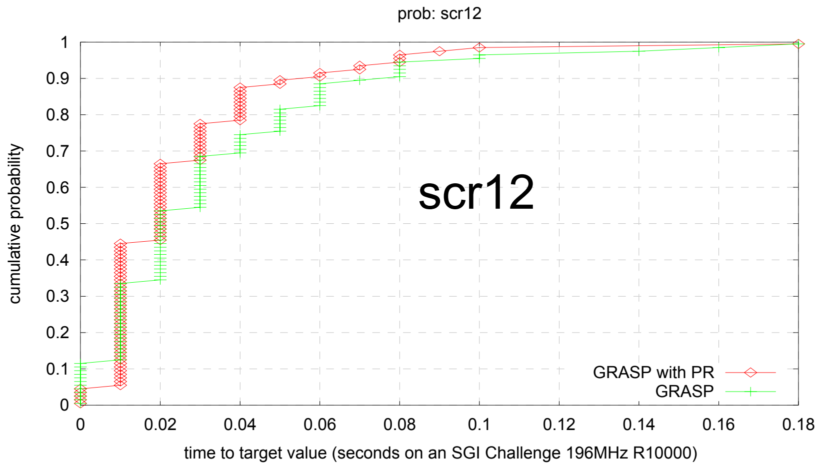
L. Steinberg [1961]

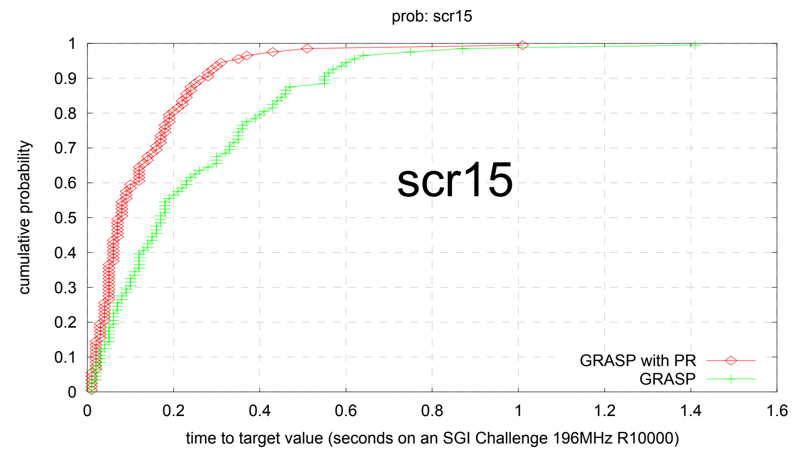


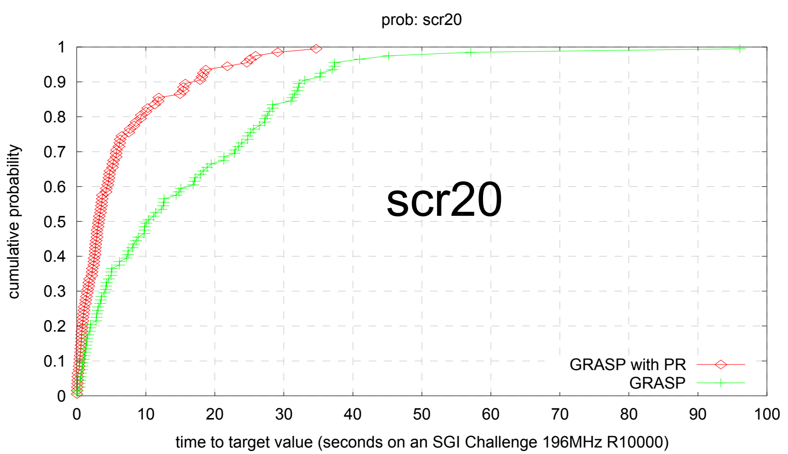




M. Scriabin and R.C. Vergin [1975]

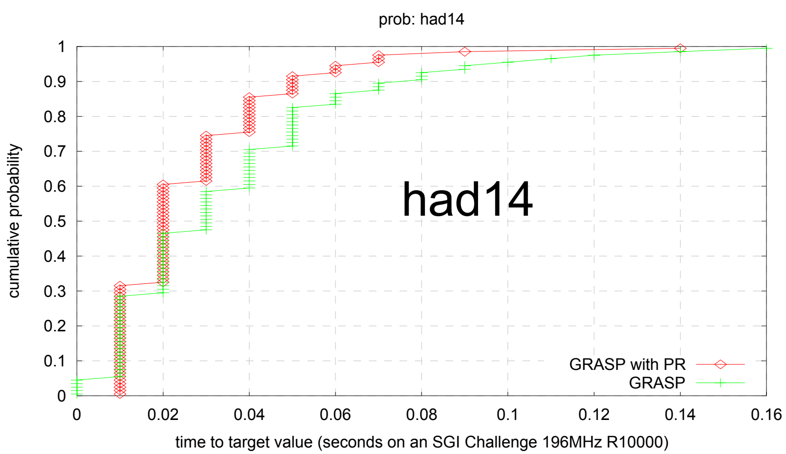


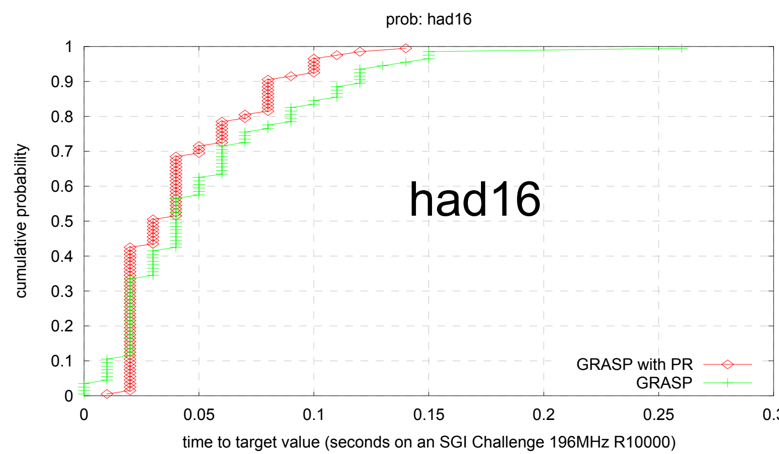


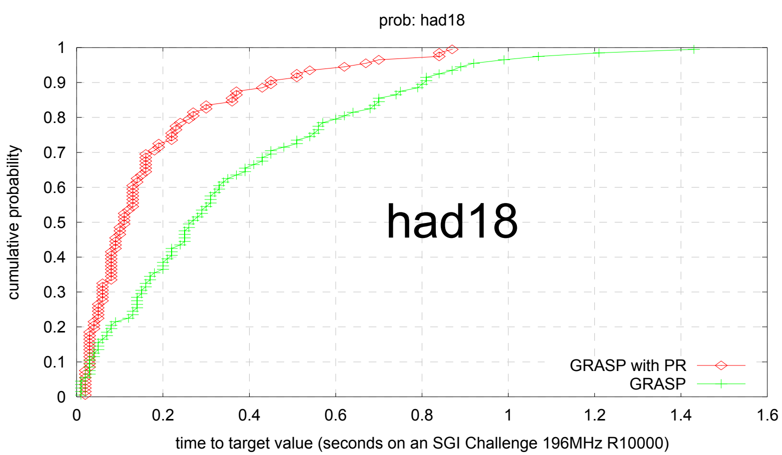


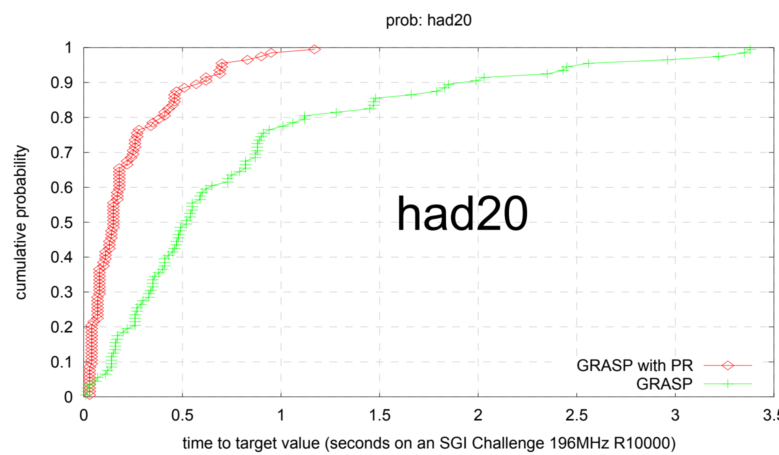
AT&T

S.W. Hadley, F. Rendl and H. Wolkowicz [1992]

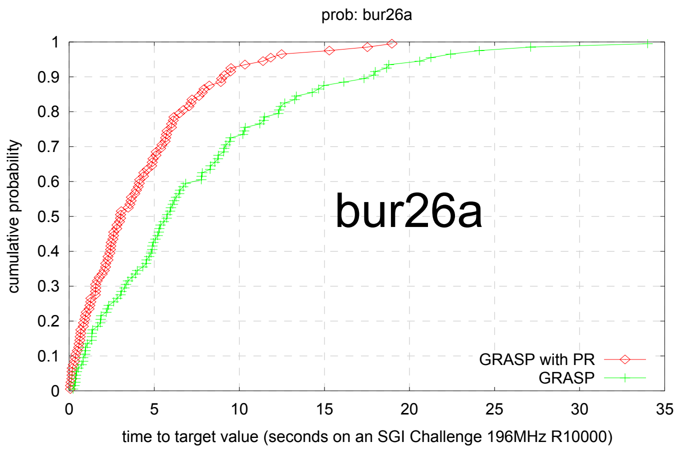


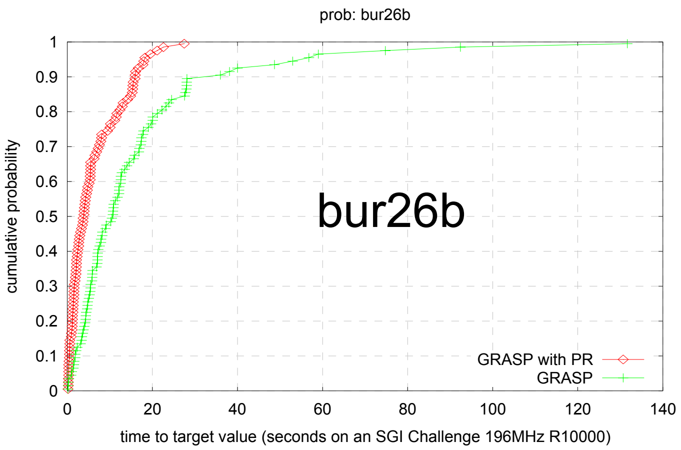


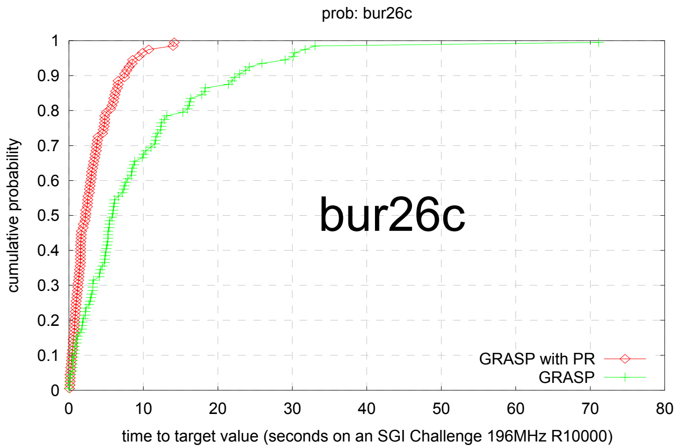


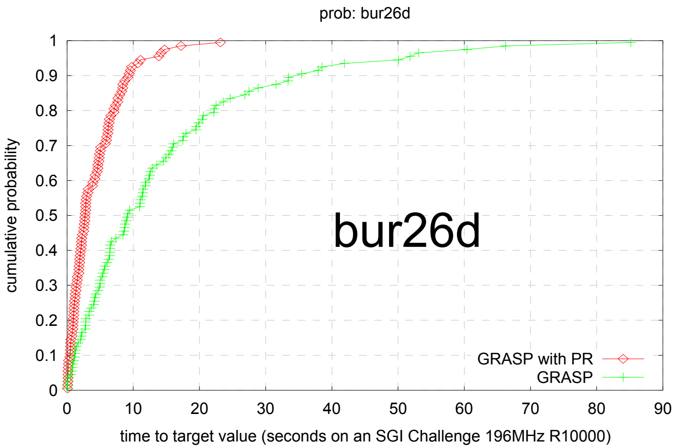


R.E. Burkard and J. Offermann [1977]



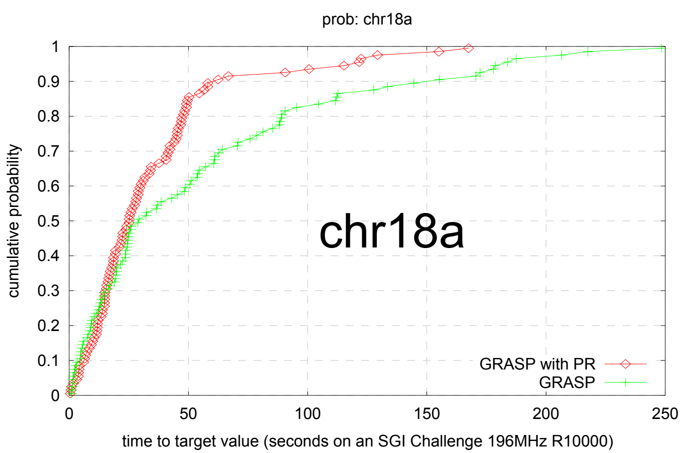


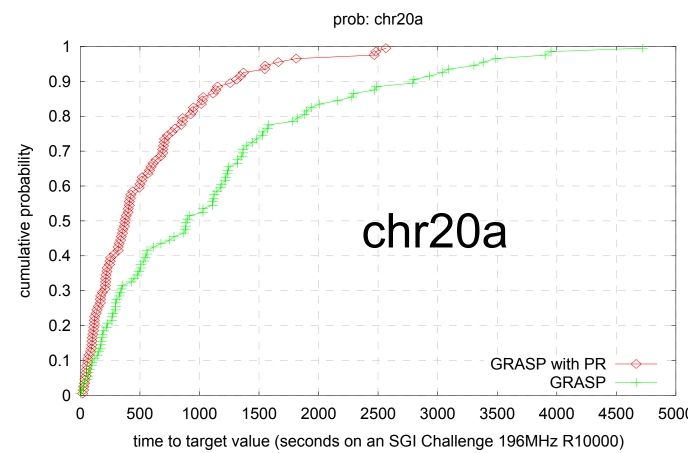


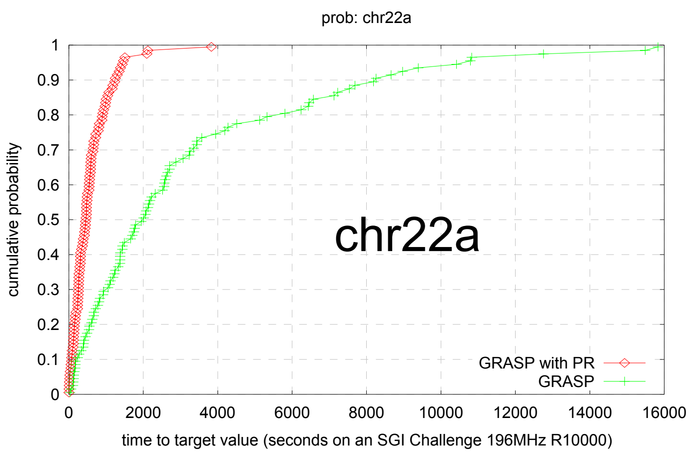


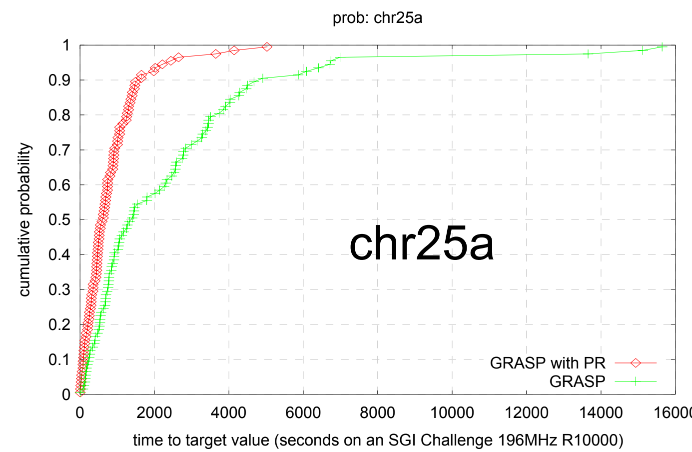
AT&T

N. Christofides and E. Benavent [1989]

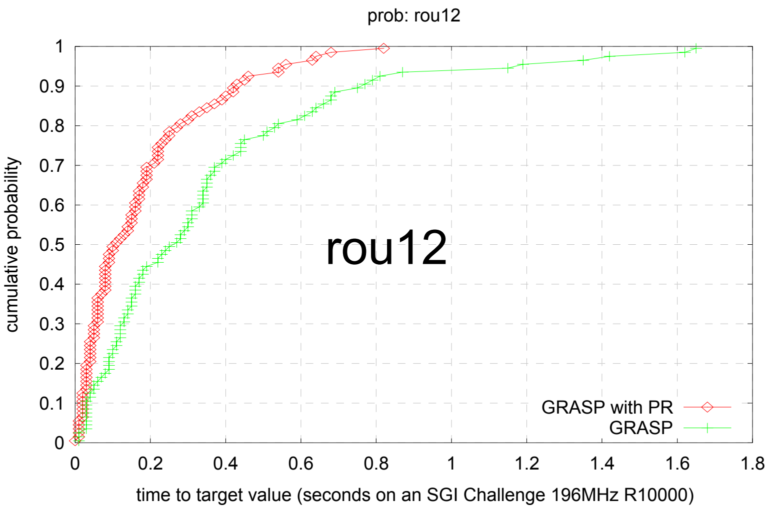


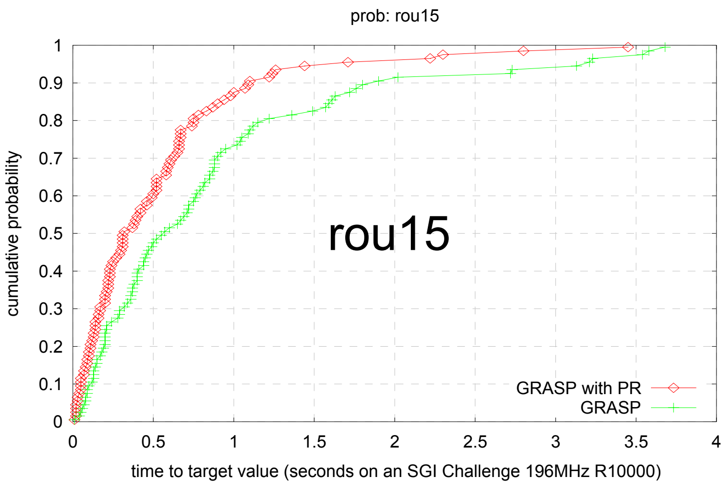


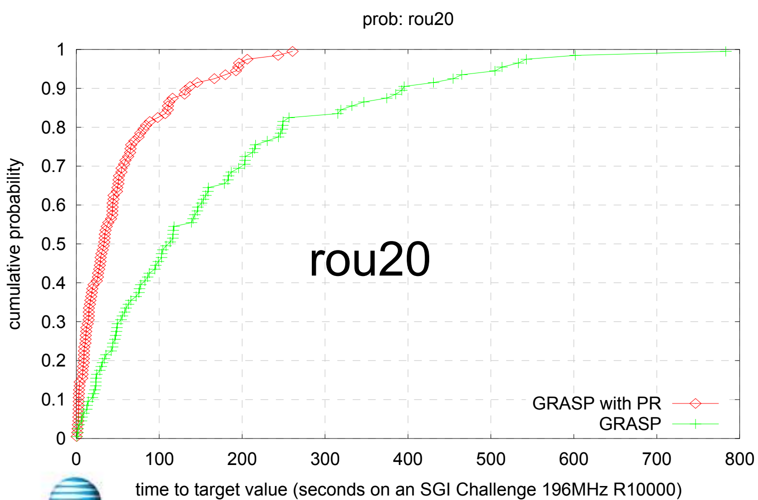




C. Roucairol [1987]

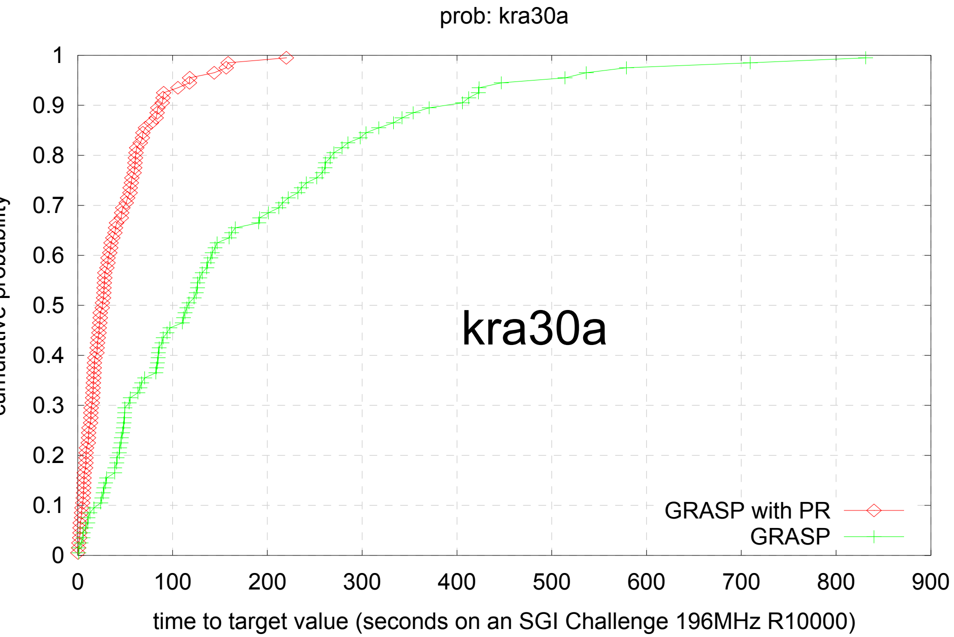


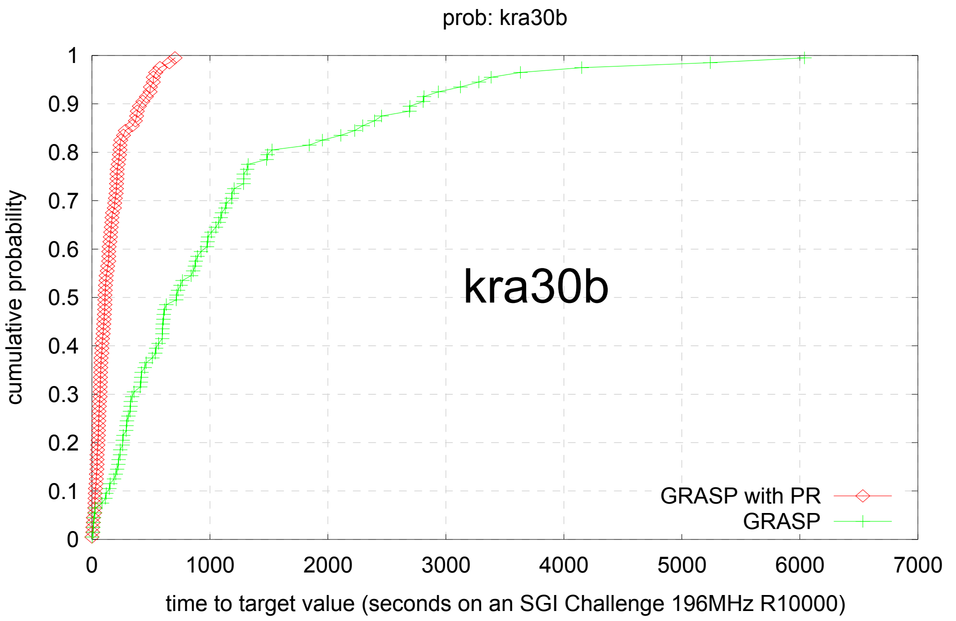




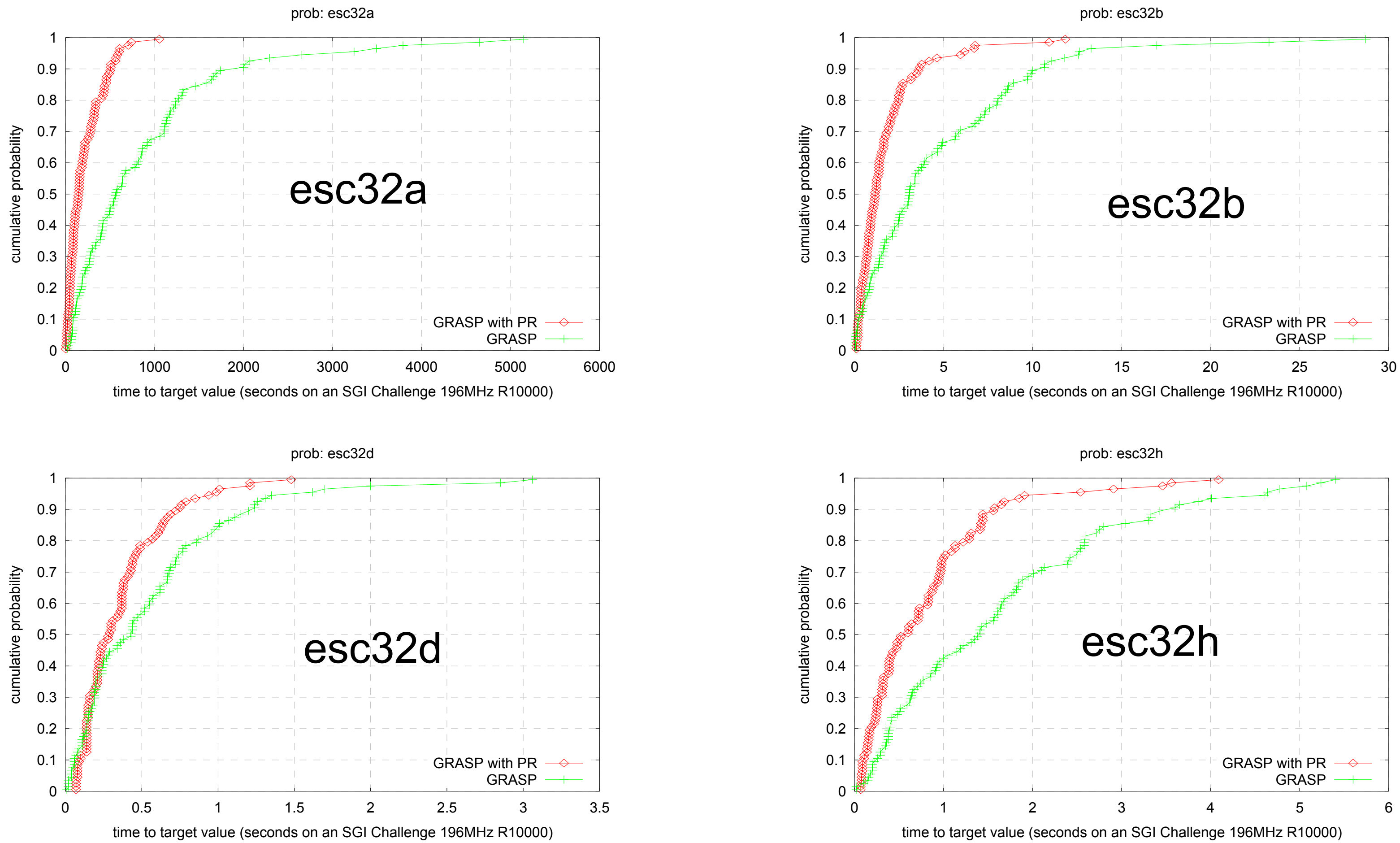
AT&T

J. Krarup and P.M. Pruzan [1978]

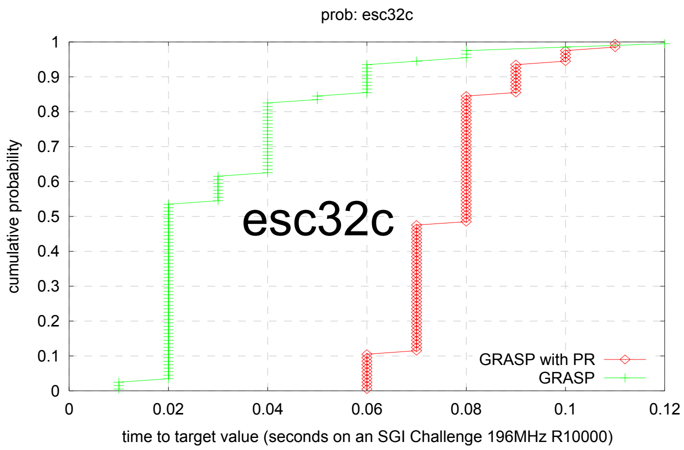


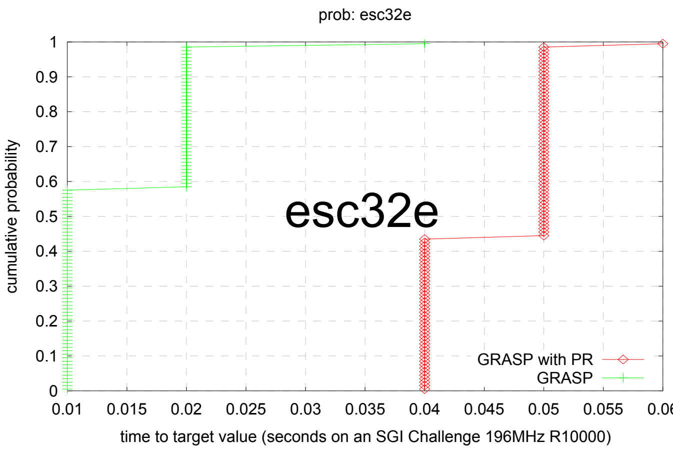


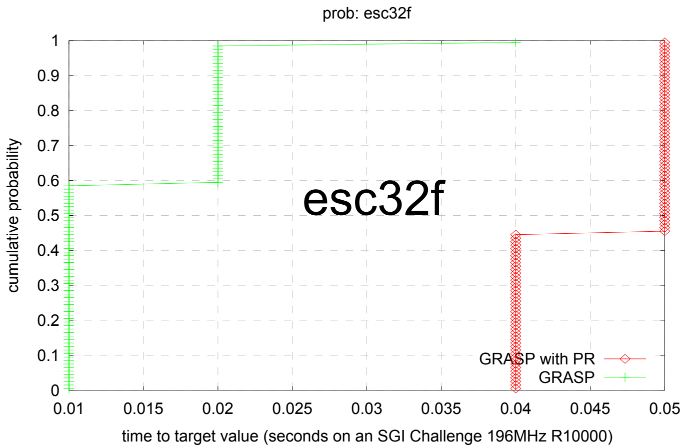
B. Eschermann and H.J. Wunderlich [1990]

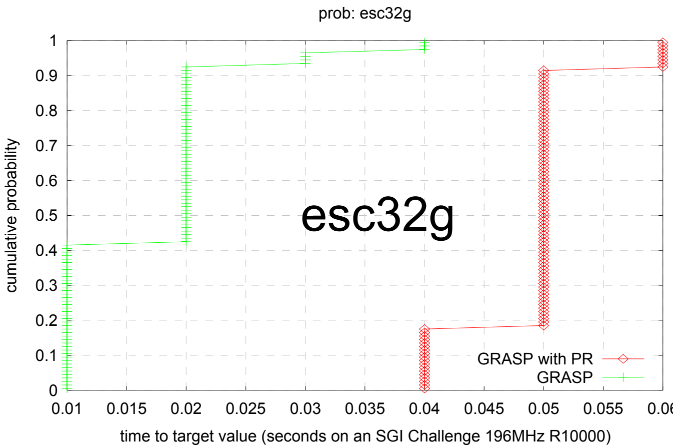


B. Eschermann and H.J. Wunderlich [1990]









AT&T

Concluding remarks

- New heuristic for the QAP is described.
- Path-relinking shown to improve performance of GRASP on almost all instances.
- Journal paper will also compare GRASP+PR with other heuristics for QAP on larger instances from QAPLIB.
- Experimental results and code are available at http://www.research.att.com/~mgcr/exp/gqapspr

Concluding remarks

- New heuristic for the QAP is described.
- Path-relinking shown to improve performance of GRASP on almost all instances.
- Journal paper will also compare GRASP+PR with other heuristics for QAP on larger instances from QAPLIB.
- Experimental results and code are available at http://www.research.att.com/~mgcr/exp/gqapspr

Concluding remarks

- New heuristic for the QAP is described.
- Path-relinking shown to improve performance of GRASP on almost all instances.
- Journal paper will also compare GRASP+PR with other heuristics for QAP on larger instances from QAPLIB.
- Experimental results and code are available at http://www.research.att.com/~mgcr/exp/gqapspr

Concluding remarks

- New heuristic for the QAP is described.
- Path-relinking shown to improve performance of GRASP on almost all instances.
- Journal paper will also compare GRASP+PR with other heuristics for QAP on larger instances from QAPLIB.
- Experimental results and code are available at http://www.research.att.com/~mgcr/exp/gqapspr

AT&T

My coauthors

Carlos A. S. Oliveira

Panos M. Pardalos