中国科学技术大学
University of Science and Technology of China

Hybrid PACO & Pheromone Initialization for VRPTWs

Wei Shi[1], Thomas Weise[1], Raymond Chiong[2], and Bülent Çatay[3]
[1] University of Science and Technology of China,
[2] The University of Newcastle, Australia
[3] Sabanci University, Turkey

2015-12-10, CIPLS @ SSCI @ Cape Town, South Africa

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i and service time s_i required for satisfying the demand once a vehicle arrives

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i and service time s_i required for satisfying the demand once a vehicle arrives, which must happen in time window $[e_i, l_i]$

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i and service time s_i required for satisfying the demand once a vehicle arrives, which must happen in time window $[e_i, l_i]$
  - all vehicles must leave central depot c_0 after e_0 and arrive back before l_0

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i and service time s_i required for satisfying the demand once a vehicle arrives, which must happen in time window $[e_i, l_i]$
  - all vehicles must leave central depot c_0 after e_0 and arrive back before l_0
  - $d_{i\,j}$ is the time required to get to c_j from c_i

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i and service time s_i required for satisfying the demand once a vehicle arrives, which must happen in time window $[e_i, l_i]$
  - all vehicles must leave central depot c_0 after e_0 and arrive back before l_0
  - $d_{i\,j}$ is the time required to get to c_j from c_i
  - each customer must be visited exactly once

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
  - homogeneous fleet of m vehicles with capacity k serves n geographically dispersed customers.
  - customer c_i has demand w_i and service time s_i required for satisfying the demand once a vehicle arrives, which must happen in time window $[e_i, l_i]$
  - all vehicles must leave central depot c_0 after e_0 and arrive back before l_0
  - $d_{i\,j}$ is the time required to get to c_j from c_i
  - each customer must be visited exactly once
  - vehicle capacity and time windows must not be violated

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals:
  - f_1: number of vehicles needed to serve the customers (minimize)

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals:
  - f_1: number of vehicles needed to serve the customers (minimize)
  - f_2: total travel distance (minimize)

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals:
  - f_1: number of vehicles needed to serve the customers (minimize)
  - f_2: total travel distance (minimize)
  - f_1 often considered as more important, since using more vehicles costs more than driving a bit longer

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals
- A permutation $\pi = (c_i, c_j, \dots)$ of the cities can be used to encode a solutiion

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals
- A permutation $\pi = (c_i, c_j, \dots)$ of the cities can be used to encode a solutiion:
  - first vehicle leaves depot c_0 and travels to c_i servicing it at $b_i = \max\{e_i, e_0 + t_{0\,i}\}$

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals
- A permutation $\pi = (c_i, c_j, \ldots)$ of the cities can be used to encode a solutiion:
  - first vehicle leaves depot c_0 and travels to c_i servicing it at $b_i = \max\{e_i, e_0 + t_{0\,i}\}$,
  - then travels to c_j, servicing it at $b_j = \max\{e_j, b_i + s_i + t_{i\,j}\}$.

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals
- A permutation $\pi = (c_i, c_j, \dots)$ of the cities can be used to encode a solutiion:
  - first vehicle leaves depot c_0 and travels to c_i servicing it at $b_i = \max\{e_i, e_0 + t_{0\,i}\}$,
  - then travels to c_j, servicing it at $b_j = \max\{e_j, b_i + s_i + t_{i\,j}\}$.
  - if vehicle capacity is exhausted or no other customer can be visited in time-window restriction, vehicle returns to c_0

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals
- A permutation $\pi = (c_i, c_j, \dots)$ of the cities can be used to encode a solutiion:
  - first vehicle leaves depot c_0 and travels to c_i servicing it at $b_i = \max\{e_i, e_0 + t_{0\,i}\}$,
  - then travels to c_j, servicing it at $b_j = \max\{e_j, b_i + s_i + t_{i\,j}\}$.
  - if vehicle capacity is exhausted or no other customer can be visited in time-window restriction, vehicle returns to c_0
  - next vehicle is used

Introduction: The VRPTW

- Vehicle Routing Problem with Time Windows (VRPTW): well-known *NP*-hard distribution logistics problem
- Two optimization goals
- A permutation $\pi = (c_i, c_j, \dots)$ of the cities can be used to encode a solutiion:
  - first vehicle leaves depot c_0 and travels to c_i servicing it at $b_i = \max\{e_i, e_0 + t_{0\,i}\}$,
  - then travels to c_j, servicing it at $b_j = \max\{e_j, b_i + s_i + t_{i\,j}\}$.
  - if vehicle capacity is exhausted or no other customer can be visited in time-window restriction, vehicle returns to c_0
  - next vehicle is used, until all customers are satisfied

- Related Work

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution

Related Work & Contribution

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution:
  - we optimize both objectives at once with a hierarchical approach

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution:
  - we optimize both objectives at once with a hierarchical approach: Solution π_i is better than π_2 if $f_1(\pi_1) < f_1(\pi_2)$ or $f_1(\pi_1) = f_1(\pi_2)$ and $f_2(\pi_1) < f_2(\pi_2)$

Related Work & Contribution

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution:
  - we optimize both objectives at once with a hierarchical approach: Solution π_i is better than π_2 if $f_1(\pi_1) < f_1(\pi_2)$ or $f_1(\pi_1) = f_1(\pi_2)$ and $f_2(\pi_1) < f_2(\pi_2)$, which is closest to the nature of the problem.

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution:
  - we optimize both objectives at once with a hierarchical approach: Solution π_i is better than π_2 if $f_1(\pi_1) < f_1(\pi_2)$ or $f_1(\pi_1) = f_1(\pi_2)$ and $f_2(\pi_1) < f_2(\pi_2)$, which is closest to the nature of the problem.
  - we investigte the Min-Max Ant System (MMAS) [16], the Ant Colony System (ACS) [17], our previously developed Initialized ACO (IACO) [18], and the Population-based ACO (PACO) algorithm [19, 20]

Related Work & Contribution

University of Science and Technology of China

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution:
  - we optimize both objectives at once with a hierarchical approach: Solution π_i is better than π_2 if $f_1(\pi_1) < f_1(\pi_2)$ or $f_1(\pi_1) = f_1(\pi_2)$ and $f_2(\pi_1) < f_2(\pi_2)$, which is closest to the nature of the problem.
  - we investigte the Min-Max Ant System (MMAS) [16], the Ant Colony System (ACS) [17], our previously developed Initialized ACO (IACO) [18], and the Population-based ACO (PACO) algorithm [19, 20]
  - we investigate and improve pheromone matrix initialization methods for these methods

Related Work & Contribution

- Related Work:
  - exact method: mainly focus on minizing distance, only feasible on small-scale problems
  - metaheuristics [1]: GAs [2, 3], ACO [4], SA [5, 6], TS [7–9], Adaptive Large Neighborhood Search [10], Variable Neighborhood Search [11], and hybrid methods [12–14]
  - two-stage approaches very common [12, 15]: first reduce number of vehicles, then reduce distance
- Contribution:
  - we optimize both objectives at once with a hierarchical approach: Solution π_i is better than π_2 if $f_1(\pi_1) < f_1(\pi_2)$ or $f_1(\pi_1) = f_1(\pi_2)$ and $f_2(\pi_1) < f_2(\pi_2)$, which is closest to the nature of the problem.
  - we investigte the Min-Max Ant System (MMAS) [16], the Ant Colony System (ACS) [17], our previously developed Initialized ACO (IACO) [18], and the Population-based ACO (PACO) algorithm [19, 20]
  - we investigate and improve pheromone matrix initialization methods for these methods
  - we hybridize the algorithm to further improve the result quality

Existing ACO Methods

中国科学技术大学
University of Science and Technology of China

- Solomon benchmark set [21]: 25-, 50-, and 100-customer instance sets

Existing ACO Methods

- Solomon benchmark set [21]: 25-, 50-, and 100-customer instance sets
- 20 independent runs per instance, 300'000 FEs per run

Existing ACO Methods

- Solomon benchmark set [21]: 25-, 50-, and 100-customer instance sets
- 20 independent runs per instance, 300'000 FEs per run

		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
ACS	IACO	0	40	16	0	52	4	0	40	16	1	51	4	0	44	12	0	54	2	0	124	44	1	157	10
ACS	MMAS	0	49	7	1	51	4	0	41	15	9	36	11	2	36	18	35	7	14	2	126	40	45	94	29
ACS	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
ACS	PACO-EBS	0	47	9	0	54	2	0	48	8	0	54	2	0	55	1	0	52	4	0	150	18	0	160	8
ACS	PACO-QBS	0	45	11	0	55	1	0	49	7	0	55	1	0	56	0	0	56	0	0	150	18	0	166	2
IACO	MMAS	0	17	39	13	28	15	9	7	40	38	7	11	35	1	20	56	0	0	44	25	99	107	35	26
IACO	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
IACO	PACO-EBS	0	9	47	9	28	19	0	31	25	4	43	9	0	42	14	2	39	15	0	82	86	15	110	43
IACO	PACO-QBS	0	13	43	8	34	14	0	37	19	4	49	3	0	54	2	0	52	4	0	104	64	12	135	21
MMAS	PACO-ABS	5	1	50	15	12	29	0	31	25	1	49	6	0	54	2	0	56	0	5	86	77	16	117	35
MMAS	PACO-EBS	3	1	52	19	12	25	0	29	27	0	49	7	0	53	3	0	56	0	3	89	82	19	117	3 2
MMAS	PACO-QBS	3	1	52	16	15	25	0	33	23	0	52	4	0	55	1	0	56	0	3	83	76	16	123	29
PACO-ABS	PACO-EBS	0	0	56	5	1	50	3	0	53	19	2	35	28	0	28	48	0	8	31	0	137	72	3	93
PACOABS	PACO-QBS	0	0	56	0	3	53	0	1	55	0	13	43	0	1	55	1	28	27	0	2	166	1	44	123
PACO-EBS	PACO-QBS	1	0	55	0	18	38	0	4	52	0	34	22	0	39	17	0	55	1	1	43	124	0	107	61

Mann-Whitey U test ($\alpha = 0.02$) comparison results for ACO algorithms ($-$ is better, $+$ is worse).

Existing ACO Methods

- Solomon benchmark set [21]: 25-, 50-, and 100-customer instance sets
- 20 independent runs per instance, 300'000 FEs per run

		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
ACS	IACO	0	40	16	0	52	4	0	40	16	1	51	4	0	44	12	0	54	2	0	124	44	1	157	10
ACS	MMAS	0	49	7	1	51	4	0	41	15	9	36	11	2	36	18	35	7	14	2	126	40	45	94	29
ACS	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
ACS	PACO-EBS	0	47	9	0	54	2	0	48	8	0	54	2	0	55	1	0	52	4	0	150	18	0	160	8
ACS	PACO-QBS	0	45	11	0	55	1	0	49	7	0	55	1	0	56	0	0	56	0	0	150	18	0	166	2
IACO	MMAS	0	17	39	13	28	15	9	7	40	38	7	11	35	1	20	56	0	0	44	25	99	107	35	26
IACO	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
IACO	PACO-EBS	0	9	47	9	28	19	0	31	25	4	43	9	0	42	14	2	39	15	0	82	86	15	110	43
IACO	PACO-QBS	0	13	43	8	34	14	0	37	19	4	49	3	0	54	2	0	52	4	0	104	64	12	135	21
MMAS	PACO-ABS	5	1	50	15	12	29	0	31	25	1	49	6	0	54	2	0	56	0	5	86	77	16	117	35
MMAS	PACO-EBS	3	1	52	19	12	25	0	29	27	0	49	7	0	53	3	0	56	0	3	89	82	19	117	3 2
MMAS	PACO-QBS	3	1	52	16	15	25	0	33	23	0	52	4	0	55	1	0	56	0	3	83	76	16	123	29
PACO-ABS	PACO-EBS	0	0	56	5	1	50	3	0	53	19	2	35	28	0	28	48	0	8	31	0	137	72	3	93
PACOABS	PACO-QBS	0	0	56	0	3	53	0	1	55	0	13	43	0	1	55	1	28	27	0	2	166	1	44	123
PACO-EBS	PACO-QBS	1	0	55	0	18	38	0	4	52	0	34	22	0	39	17	0	55	1	1	43	124	0	107	61

Mann-Whitey U test ($\alpha = 0.02$) comparison results for ACO algorithms ($-$ is better, $+$ is worse).

- ACS performs worst

- Solomon benchmark set [21]: 25-, 50-, and 100-customer instance sets
- 20 independent runs per instance, 300'000 FEs per run

		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
ACS	IACO	0	40	16	0	52	4	0	40	16	1	51	4	0	44	12	0	54	2	0	124	44	1	157	10
ACS	MMAS	0	49	7	1	51	4	0	41	15	9	36	11	2	36	18	35	7	14	2	126	40	45	94	29
ACS	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
ACS	PACO-EBS	0	47	9	0	54	2	0	48	8	0	54	2	0	55	1	0	52	4	0	150	18	0	160	8
ACS	PACO-QBS	0	45	11	0	55	1	0	49	7	0	55	1	0	56	0	0	56	0	0	150	18	0	166	2
IACO	MMAS	0	17	39	13	28	15	9	7	40	38	7	11	35	1	20	56	0	0	44	25	99	107	35	26
IACO	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
IACO	PACO-EBS	0	9	47	9	28	19	0	31	25	4	43	9	0	42	14	2	39	15	0	82	86	15	110	43
IACO	PACO-QBS	0	13	43	8	34	14	0	37	19	4	49	3	0	54	2	0	52	4	0	104	64	12	135	21
MMAS	PACO-ABS	5	1	50	15	12	29	0	31	25	1	49	6	0	54	2	0	56	0	5	86	77	16	117	35
MMAS	PACO-EBS	3	1	52	19	12	25	0	29	27	0	49	7	0	53	3	0	56	0	3	89	82	19	117	3 2
MMAS	PACO-QBS	3	1	52	16	15	25	0	33	23	0	52	4	0	55	1	0	56	0	3	83	76	16	123	29
PACO-ABS	PACO-EBS	0	0	56	5	1	50	3	0	53	19	2	35	28	0	28	48	0	8	31	0	137	72	3	93
PACOABS	PACO-QBS	0	0	56	0	3	53	0	1	55	0	13	43	0	1	55	1	28	27	0	2	166	1	44	123
PACO-EBS	PACO-QBS	1	0	55	0	18	38	0	4	52	0	34	22	0	39	17	0	55	1	1	43	124	0	107	61

Mann-Whitey U test ($\alpha = 0.02$) comparison results for ACO algorithms ($-$ is better, $+$ is worse).

- ACS performs worst
- PACO with QBS rule performs best

Existing ACO Methods

- Solomon benchmark set [21]: 25-, 50-, and 100-customer instance sets
- 20 independent runs per instance, 300'000 FEs per run

		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
ACS	IACO	0	40	16	0	52	4	0	40	16	1	51	4	0	44	12	0	54	2	0	124	44	1	157	10
ACS	MMAS	0	49	7	1	51	4	0	41	15	9	36	11	2	36	18	35	7	14	2	126	40	45	94	29
ACS	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
ACS	PACO-EBS	0	47	9	0	54	2	0	48	8	0	54	2	0	55	1	0	52	4	0	150	18	0	160	8
ACS	PACO-QBS	0	45	11	0	55	1	0	49	7	0	55	1	0	56	0	0	56	0	0	150	18	0	166	2
IACO	MMAS	0	17	39	13	28	15	9	7	40	38	7	11	35	1	20	56	0	0	44	25	99	107	35	26
IACO	PACO-ABS	0	12	44	9	31	16	0	35	21	5	45	6	0	54	2	0	51	5	0	101	67	14	127	27
IACO	PACO-EBS	0	9	47	9	28	19	0	31	25	4	43	9	0	42	14	2	39	15	0	82	86	15	110	43
IACO	PACO-QBS	0	13	43	8	34	14	0	37	19	4	49	3	0	54	2	0	52	4	0	104	64	12	135	21
MMAS	PACO-ABS	5	1	50	15	12	29	0	31	25	1	49	6	0	54	2	0	56	0	5	86	77	16	117	35
MMAS	PACO-EBS	3	1	52	19	12	25	0	29	27	0	49	7	0	53	3	0	56	0	3	89	82	19	117	3 2
MMAS	PACO-QBS	3	1	52	16	15	25	0	33	23	0	52	4	0	55	1	0	56	0	3	83	76	16	123	29
PACO-ABS	PACO-EBS	0	0	56	5	1	50	3	0	53	19	2	35	28	0	28	48	0	8	31	0	137	72	3	93
PACOABS	PACO-QBS	0	0	56	0	3	53	0	1	55	0	13	43	0	1	55	1	28	27	0	2	166	1	44	123
PACO-EBS	PACO-QBS	1	0	55	0	18	38	0	4	52	0	34	22	0	39	17	0	55	1	1	43	124	0	107	61

Mann-Whitey U test ($\alpha = 0.02$) comparison results for ACO algorithms ($-$ is better, $+$ is worse).

- ACS performs worst
- PACO with QBS rule performs best $\Rightarrow$ use from now on

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD (for PD, we test normal, uniform, and power distribution PDFs)

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD
- Define *VE* as a function which is larger if c_i and c_j are close and if c_j would be serviced at the end of its time window if visited directly after c_i

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD
- Define *VE* as a function which is larger if c_i and c_j are close
- Set $\tau_{ij}^0 \approx \max\left\{\frac{1}{n}, \int_{e_i}^{l_i} PD(x) * \mathit{VE}(i, j, x)dx\right\}$

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD
- Define *VE* as a function which is larger if c_i and c_j are close
- Set $\tau_{ij}^0 \approx \max\left\{\frac{1}{n}, \int_{e_i}^{l_i} PD(x) * \mathit{VE}(i,j,x)dx\right\}$
- Experiments with PACO-QBS and the three different probability distribution models show. . .

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD
- Define *VE* as a function which is larger if c_i and c_j are close
- Set $\tau_{ij}^{0} \approx \max\left\{\frac{1}{n}, \int_{e_i}^{l_i} PD(x) * \mathit{VE}(i,j,x)dx\right\}$
- Experiments with PACO-QBS and the three different probability distribution models show...

		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
NoIni	Normal	0	2	54	4	26	26	4	7	45	6	32	18	8	14	34	4	32	20	12	23	133	14	90	64
NoIni	Power	0	3	53	1	33	22	2	7	47	3	35	18	5	13	38	4	34	18	7	23	138	8	102	58
NoIni	Uniform	0	2	54	1	39	16	1	7	48	4	40	12	3	15	38	2	42	12	4	24	140	7	121	40
Normal	Power	0	0	56	1	13	42	0	2	54	1	8	47	0	1	55	1	6	49	0	3	165	3	27	138
Normal	Uniform	0	0	56	0	19	37	0	3	53	1	13	42	0	3	53	0	15	41	0	6	162	1	47	120
Power	Uniform	0	0	56	4	0	52	0	0	56	3	1	52	0	0	56	9	0	47	0	0	168	16	1	151

Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD
- Define *VE* as a function which is larger if c_i and c_j are close
- Set $\tau_{ij}^{0} \approx \max\left\{\frac{1}{n}, \int_{e_i}^{l_i} PD(x) * \mathit{VE}(i,j,x)dx\right\}$
- Experiments with PACO-QBS and the three different probability distribution models show that phromone-initialized PACO performs significantly better

		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
NoIni	Normal	0	2	54	4	26	26	4	7	45	6	32	18	8	14	34	4	32	20	12	23	133	14	90	64
NoIni	Power	0	3	53	1	33	22	2	7	47	3	35	18	5	13	38	4	34	18	7	23	138	8	102	58
NoIni	Uniform	0	2	54	1	39	16	1	7	48	4	40	12	3	15	38	2	42	12	4	24	140	7	121	40
Normal	Power	0	0	56	1	13	42	0	2	54	1	8	47	0	1	55	1	6	49	0	3	165	3	27	138
Normal	Uniform	0	0	56	0	19	37	0	3	53	1	13	42	0	3	53	0	15	41	0	6	162	1	47	120
Power	Uniform	0	0	56	4	0	52	0	0	56	3	1	52	0	0	56	9	0	47	0	0	168	16	1	151

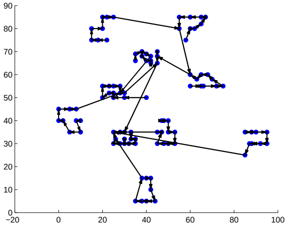
Pheromone Initialization

- Utilize static information from problem instance to initialize pheromones for PACO-QBS
- Model service begin time b_i as random variable PD
- Define *VE* as a function which is larger if c_i and c_j are close
- Set $\tau_{ij}^0 \approx \max\left\{\frac{1}{n}, \int_{e_i}^{l_i} PD(x) * \mathit{VE}(i,j,x)dx\right\}$
- Experiments with PACO-QBS and the three different probability distribution models show that phromone-initialized PACO performs significantly better and power distributed b performs best

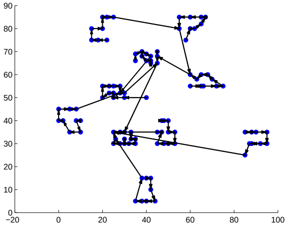
		Instances with 25 customers						Instances with 50 customers						Instances with 100 customers						Total					
		f_1			f_2			f_1			f_2			f_1			f_2			f_1			f_2		
Algorithm 1 vs. 2		-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0	-	+	0
NoIni	Normal	0	2	54	4	26	26	4	7	45	6	32	18	8	14	34	4	32	20	12	23	133	14	90	64
NoIni	Power	0	3	53	1	33	22	2	7	47	3	35	18	5	13	38	4	34	18	7	23	138	8	102	58
NoIni	Uniform	0	2	54	1	39	16	1	7	48	4	40	12	3	15	38	2	42	12	4	24	140	7	121	40
Normal	Power	0	0	56	1	13	42	0	2	54	1	8	47	0	1	55	1	6	49	0	3	165	3	27	138
Normal	Uniform	0	0	56	0	19	37	0	3	53	1	13	42	0	3	53	0	15	41	0	6	162	1	47	120
Power	Uniform	0	0	56	4	0	52	0	0	56	3	1	52	0	0	56	9	0	47	0	0	168	16	1	151

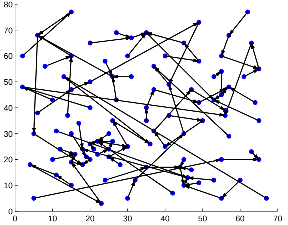
- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components

- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered



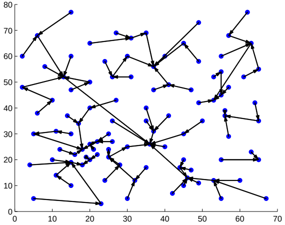
- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered, but not if customers and time windows are completely random





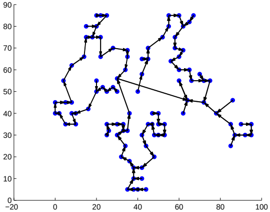
Improved Initialization

- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered, but not if customers and time windows are completely random
- Method 1: Change *VE* to put more pheromones on shorter edges

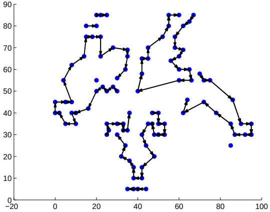


Improved Initialization

- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered, but not if customers and time windows are completely random
- Method 1: Change *VE* to put more pheromones on shorter edges
- Method 2: Keep initialized pheromone only on one edge per node; two choices maximum or difference selection (see paper)



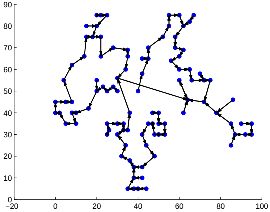
$\Downarrow$



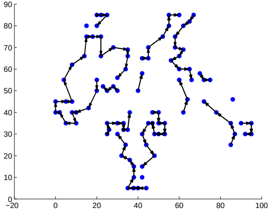
maximum selection

Improved Initialization

- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered, but not if customers and time windows are completely random
- Method 1: Change *VE* to put more pheromones on shorter edges
- Method 2: Keep initialized pheromone only on one edge per node; two choices maximum or difference selection (see paper)



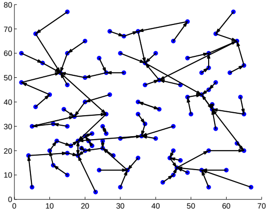
$\Downarrow$



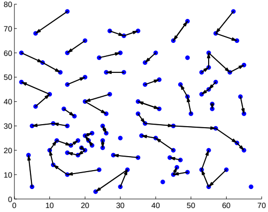
difference selection

中国科学技术大学 University of Science and Technology of China

- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered, but not if customers and time windows are completely random
- Method 1: Change *VE* to put more pheromones on shorter edges
- Method 2: Keep initialized pheromone only on one edge per node; two choices maximum or difference selection (see paper)



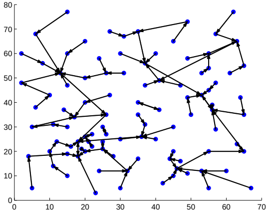
$\Downarrow$



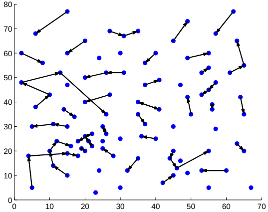
maximum selection

Improved Initialization

- Ideally, initialization should assign pheromones such that the edges with the strongest pheromones form larger tour components
- This works especially for instances where customers are clustered, but not if customers and time windows are completely random
- Method 1: Change *VE* to put more pheromones on shorter edges
- Method 2: Keep initialized pheromone only on one edge per node; two choices maximum or difference selection (see paper)



⇓



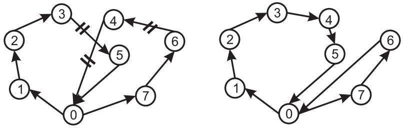
difference selection

Hybridize with Local Search

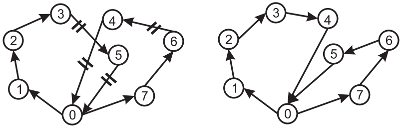
- Homberger and Gehring [12] proposed a hybrid metaheuristic that randomly selects one neighborhood from $\{N_{1-\mathit{insert}}, N_{1-\mathit{exchange}}, N_{2-\mathit{opt}}\}$ to refine solutions with local search

Hybridize with Local Search

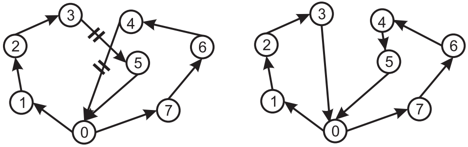
- Homberger and Gehring [12] proposed a hybrid metaheuristic that randomly selects one neighborhood from $\{N_{1-insert}, N_{1-exchange}, N_{2-opt}\}$ to refine solutions with local search



$N_{1-insert}$



$N_{1-exchange}$



N_{2-opt}

Hybridize with Local Search

- Homberger and Gehring [12] proposed a hybrid metaheuristic that randomly selects one neighborhood from $\{N_{1-insert}, N_{1-exchange}, N_{2-opt}\}$ to refine solutions with local search
- We adopt this mechanism into PI-PACO.

- Homberger and Gehring [12] proposed a hybrid metaheuristic that randomly selects one neighborhood from $\{N_{1-insert}, N_{1-exchange}, N_{2-opt}\}$ to refine solutions with local search
- We adopt this mechanism into PI-PACO.
- Hybrid PI-PACO with difference selection achieves better results than hybrid PACO without pheromone initialization

Type	Goal	Chen and Ting [14]	Sodsoon and Changyom [22]	hybrid PACO	hybrid PI-PACO maximum selection	hybrid PI-PACO difference selection
R1	f_1	12.83	13.83	12.83	12.92	**12.75**
	f_2	1203.56	1259.19	1204.06	1205.11	1203.67
C1	f_1	**10**	**10**	**10**	**10**	**10**
	f_2	828.76	838.12	828.61	828.60	**828.55**
RC1	f_1	12.50	12.63	12.75	12.63	**12.38**
	f_2	1363.84	1436.58	1381.42	1380.78	1380.54
R2	f_1	**3.09**	3.82	3.45	3.64	3.54
	f_2	**932.23**	980.98	1005.35	995.03	1006.38
C2	f_1	**3**	**3**	3	**3**	**3**
	f_2	**589.86**	591.13	590.71	589.93	**589.86**
RC2	f_1	**3.75**	4.5	4.13	4.38	4.13
	f_2	**1079.81**	1141.63	1113.59	1156.20	1109.8

- Homberger and Gehring [12] proposed a hybrid metaheuristic
- We adopt this mechanism into PI-PACO.
- Hybrid PI-PACO with difference selection achieves better results than hybrid PACO without pheromone initialization
- It outperforms the hybrid algorithm by Chen and Ting [14] on problem type C1 and achieves similar results on problem type C2

Type	Goal	Chen and Ting [14]	Sodsoon and Changyom [22]	hybrid PACO	hybrid PI-PACO maximum selection	hybrid PI-PACO difference selection
R1	f_1	12.83	13.83	12.83	12.92	**12.75**
	f_2	1203.56	1259.19	1204.06	1205.11	1203.67
C1	f_1	**10**	**10**	**10**	**10**	**10**
	f_2	828.76	838.12	828.61	828.60	**828.55**
RC1	f_1	12.50	12.63	12.75	12.63	**12.38**
	f_2	1363.84	1436.58	1381.42	1380.78	1380.54
R2	f_1	**3.09**	3.82	3.45	3.64	3.54
	f_2	**932.23**	980.98	1005.35	995.03	1006.38
C2	f_1	**3**	**3**	3	**3**	**3**
	f_2	**589.86**	591.13	590.71	589.93	**589.86**
RC2	f_1	**3.75**	4.5	4.13	4.38	4.13
	f_2	**1079.81**	1141.63	1113.59	1156.20	1109.8

- Homberger and Gehring [12] proposed a hybrid metaheuristic
- We adopt this mechanism into PI-PACO.
- Hybrid PI-PACO with difference selection achieves better results than hybrid PACO without pheromone initialization
- It is similar to the MMAS-VRPTW [22] but outperforms it on all except R2 instances in terms of the distance

Type	Goal	Chen and Ting [14]	Sodsoon and Changyom [22]	hybrid PACO	hybrid PI-PACO maximum selection	hybrid PI-PACO difference selection
R1	f_1	12.83	13.83	12.83	12.92	**12.75**
	f_2	1203.56	1259.19	1204.06	1205.11	1203.67
C1	f_1	**10**	**10**	**10**	**10**	**10**
	f_2	828.76	838.12	828.61	828.60	**828.55**
RC1	f_1	12.50	12.63	12.75	12.63	**12.38**
	f_2	1363.84	1436.58	1381.42	1380.78	1380.54
R2	f_1	**3.09**	3.82	3.45	3.64	3.54
	f_2	**932.23**	980.98	1005.35	995.03	1006.38
C2	f_1	**3**	**3**	3	**3**	**3**
	f_2	**589.86**	591.13	590.71	589.93	**589.86**
RC2	f_1	**3.75**	4.5	4.13	4.38	4.13
	f_2	**1079.81**	1141.63	1113.59	1156.20	1109.8

- PACO best ACO for VRPTW

Summary

- PACO best ACO for VRPTW
- Pheromone matrix initialization makes it better

- PACO best ACO for VRPTW
- Pheromone matrix initialization makes it better
- Hybridization + pheromone matrix initialization is best

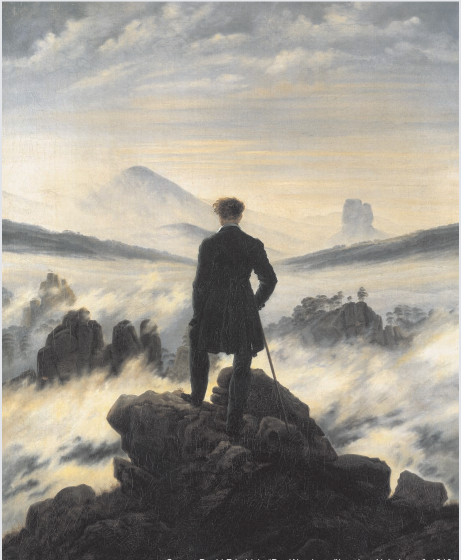
Summary

- PACO best ACO for VRPTW
- Pheromone matrix initialization makes it better
- Hybridization + pheromone matrix initialization is best
- Concept should be tested in offer domains, such as quadratic assignment poblems

谢谢！

Thank you.

Wei Shi[1], Thomas Weise[1], Raymond Chiong[2], and Bülent Çatay[3]
[1] University of Science and Technology of China,
[2] The University of Newcastle, Australia
[3] Sabanci University, Turkey



Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.
2. Jean Berger and Mohamed Barkaoui. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers and Operations Research*, 31(12):2037–2053, 2004.
3. Jean-Yves Potvin and Samy Bengio. The vehicle routing problem with time windows part ii: Genetic search. *INFORMS Journal on Computing*, 8(2):165–172, 1996.
4. Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, London, U.K., 1999.
5. Haibing Li and Andrew Lim. Local search with annealing-like restarts to solve the VRPTW. *European Journal of Operational Research*, 150(1):115–127, 2003.
6. Wen-Chyuan Chiang and Robert A. Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27, 1996.
7. Jörg Homberger and Hermann Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Information Systems and Operational Research*, 37:297–318, 1999.
8. Philippe Badeau, François Guertin, Michel Gendreau, Jean-Yves Potvin, and Eric Taillard. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5(2):109–122, 1997.
9. Robert A. Russell. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Science*, 29(2): 156–166, 1995.
10. David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34 (8):2403–2435, 2007.
11. Louis-Martin Rousseau, Michel Gendreau, and Gilles Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8(1):43–58, 2002.
12. Jörg Homberger and Hermann Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.
13. Jörg Homberger and Hermann Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.

14. Chia-Ho Chen and Ching-Jung Ting. A hybrid ant colony system for vehicle routing problem with time windows. *Journal of the Eastern Asia Society for Transportation Studies*, 6:2822–2836, 2005.
15. Fuh-Hwa Franklin Liu and Sheng-Yuan Shen. A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operational Research*, 118:485–504, 1999.
16. Thomas Stützle and Holger H. Hoos. Max-min ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.
17. Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE TEVC*, 1(1), 1997.
18. Wei Shi and Thomas Weise. An initialized aco for the vrptw. In *Proc. of the 14th Intl. Conf. on Intelligent Data Engineering and Automated Learning*, pages 93–100, Hefei, China, October 20–23, 2013. Springer.
19. Michael Guntsch and Martin Middemdorf. A population based approach for aco. In *Applications of Evolutionary Computing*, volume 2279 of *Lecture Notes in Computer Science*, pages 72–81, Kinsale, Ireland, April 2002. Springer.
20. Sabrina Oliveira, Mohamed Saifullah Hussin, and Thomas Stützle. A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. Technical Report 006, Université Libre de Bruxelles, IRIDIA, Bruxelles, Belgium, 2011.
21. Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.
22. Suphan Sodsoon and Preecha Changyom. Max-min ant system (mmas) for vehicle routing problem with time windows. *KKU Engineering Journal*, 38(3):313–323, 2011.