META-HEURISTICS DEVELOPMENT FRAMEWORK: DESIGN AND APPLICATIONS

MASTER THESIS PRESENTATION

28th of July, 2004

WAN Wee Chong, Jason

National University of Singapore

School of Computing (SoC)

AGENDA

- **Introduction**
- **Literature Reviews**
- **Meta-heuristics Development Framework**
  - Components
  - Search Strategies
- ***Case Study:*** The Traveling Salesman Problem
  - Hybridized Schemes
  - Observations of Results
- **Project Summary**
- **Questions**

INTRODUCTION

INTRODUCTION

❑ Meta-heuristics

- Traditional Methods
  - Inadequate at solving large-scaled combinatorial optimization problems
  - Meta-heuristics matured rapidly as a solution
- Popular Meta-heuristics
  - E.g. Tabu Search, Simulated Annealing, Genetic Algorithms, Constraint Local Search
- New Meta-heuristics
  - E.g. Ants Colony Optimization, Path Re-linking
- Potential of hybridization
  - Diverse growth of meta-heuristics of various nature
  - Utilize the forte of meta-heuristics

INTRODUCTION

❑ Conventional Development Approach

- Developing from Scratch
- Waste of resources (Man and Machines)
- Absence of a standard template for benchmarking

❑ Demand for an *efficient development tool*

- Major reduction in developing time
- Standard platform for implementation and benchmarking
- Object oriented (discipline, ease of integration and extension)
- Facilitate rapid prototyping of new techniques

INTRODUCTION

Design Goals

- *Generic*
  - Work with most if not all combinatorial optimization problem
  - Able to model various search strategies
    - E.g. Hybridization, Intensification, Diversification
- *Reusability*
  - Offload repetitive search routines
  - Reuse both design and codes
- *Clarity*
  - Unambiguous interfaces that gives clarity
  - Allow rapid implementation of application

LITERATURE REVIEWS

LITERATURE REVIEWS

- **MDF**
  - Lau and Wan, 2004
- **Open TS**
  - Robert Harder, 2003
- **Localizer ++**
  - Michel and Hentenryck, 2001
- **Easy Local ++**
  - Gaspero and Schaef, 2001
- **Hot Frame**
  - Fink and Voß, 2002

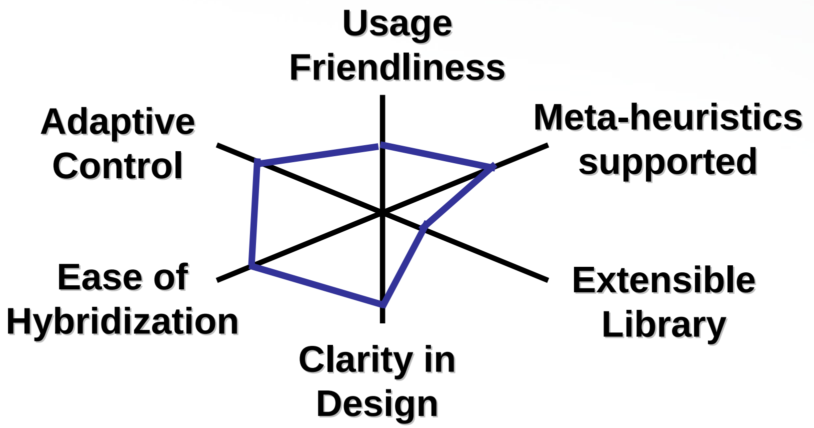
- **Comparison factors**

1. Usage Friendliness
2. Number of meta-heuristics supported
3. Clarity in Design
4. Adaptive Control
5. Ease of Hybridization
6. Extensible Library

LITERATURE REVIEWS

❑ MDF Features

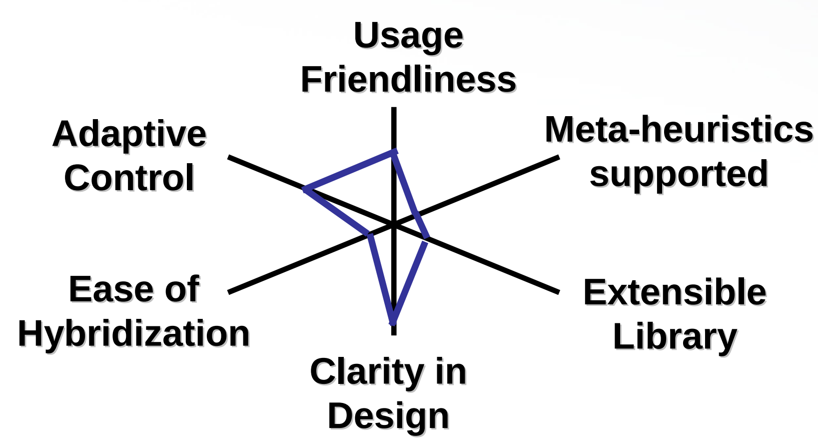
- C++ Meta-heuristics framework
- Adopt object-oriented design (OOD)
  - Using interfaces to achieve genericity
- Currently supported four (4) meta-heuristics
  - Tabu Search, Ants Colony Optimization, Simulated Annealing and Genetic Algorithm
- Centralized control mechanism for adaptive search
  - Using Request and Response Metaphor
  - Model into an "event-driven" search
- Include a software library
  - Speed-up development progress
  - Reduce coding errors



LITERATURE REVIEWS

Open TS Features

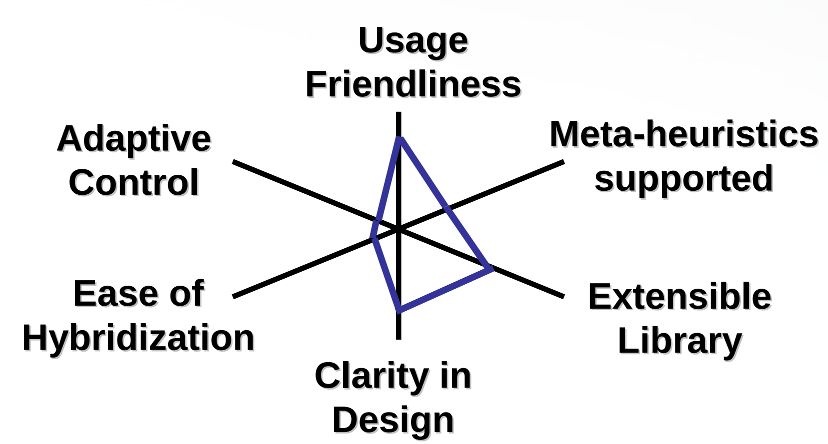
- Initialized by Computational Infrastructure for Operation Research (COIN-OR)
- Java-based Tabu Search
- Generic aspect achieved through interfaces
- Unambiguous interfaces that define clearly their collaborative roles in the algorithm
  - Solution, Move Manager, Move, Objective Function, Tabu list
- Support adaptive control through decentralized *Event Listeners*
- Limited support of software components
  - Simple/Complex Tabu List
  - Complex Move



LITERATURE REVIEWS

❑ Localizer ++ Features

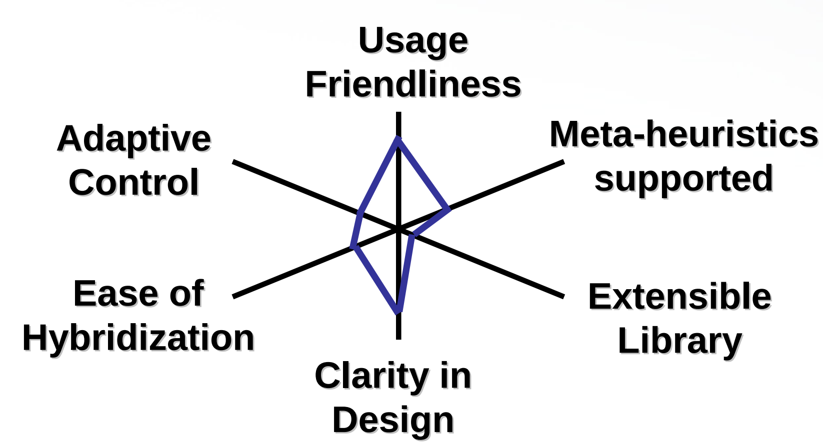
- C ++ Constraint-based Local search framework
- Require formulation of problem into its mathematical equivalent
  - E.g. Decision Variables, Objective Functions and Constraint Equations
- Has a two-level architecture
  - *Declarative components:* for data storage management
  - *Search components:* for defining search procedure
- Search components can incorporate various local search algorithms
  - Neighborhood operators
  - Tabu Lists
- Extensible constraint library
  - All-diff constraints



LITERATURE REVIEWS

Easy Local ++ Features

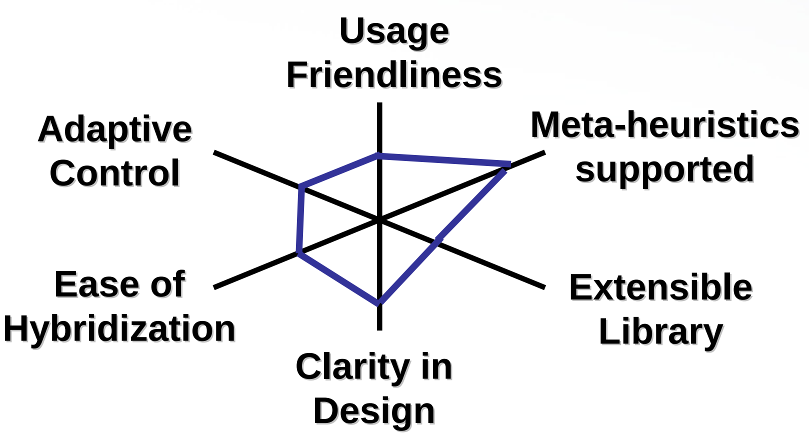
- C ++ Local search framework
- Adopt a four level architecture to implement local search techniques
  - *Basic Data:* for maintaining the states of search space
  - *Helpers:* for handling search actions such as exploration of neighborhood
  - *Runners:* for performing the routine of the meta-heuristics using the helpers
  - *Solvers:* for generating the initial solutions
- Additional support classes
  - *Kickers:* for diversification
  - *Testers*: for debugging
- Support limited hybridization
  - E.g. SA as diversifier
- Absence of component library



LITERATURE REVIEWS

❑ Hot Frame Features

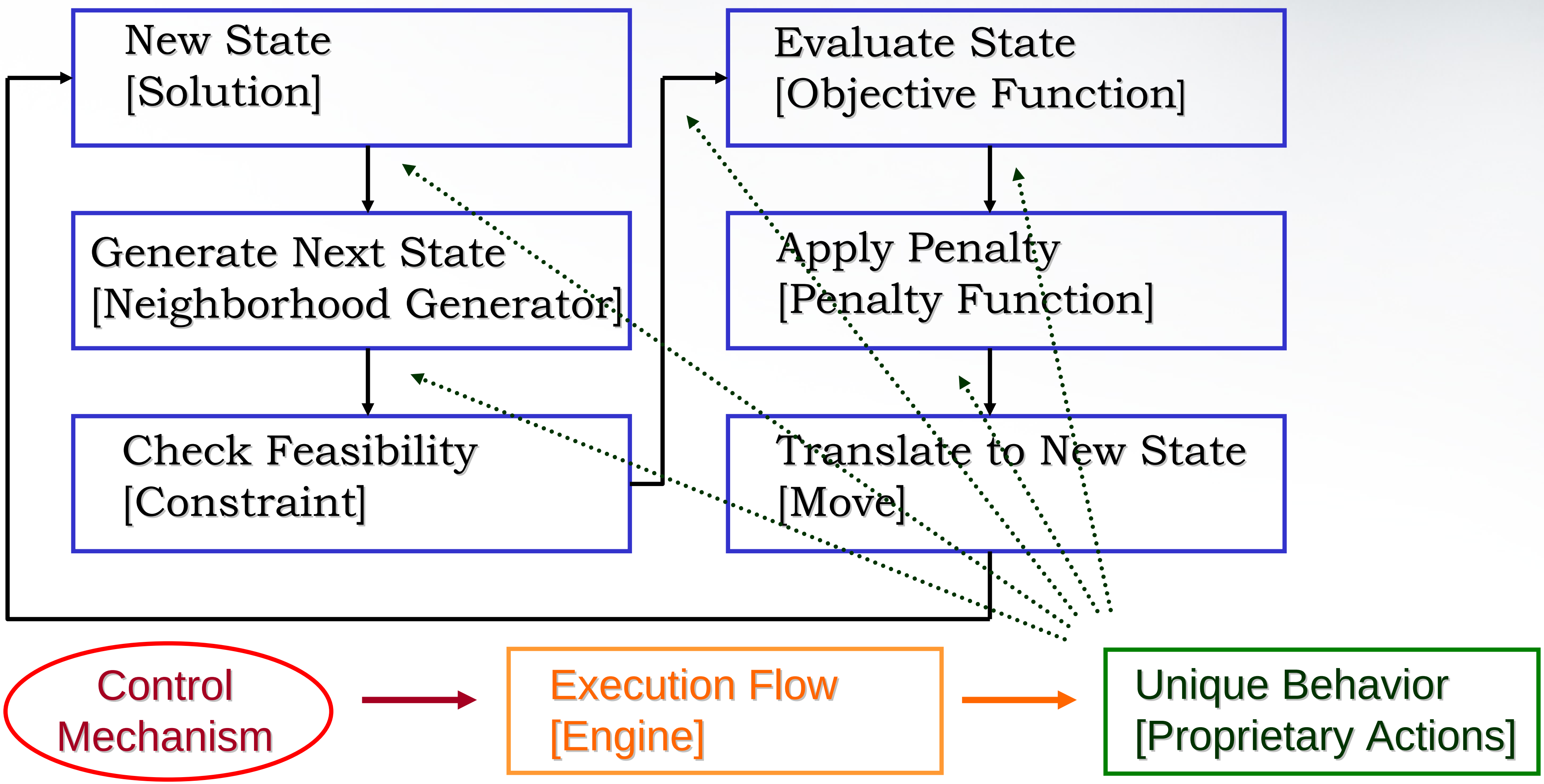
- C ++ Meta-heuristics framework
- Support various meta-heuristics and their derivations
  - Tabu Search, Simulated Annealing, Evolutionary Algorithms
- Use template design for meta-heuristics routines
- Use inheritance to override the meta-heuristics procedures
- Provide general software components
  - Reusable data structure classes
    - E.g. binary vectors, permutations, combined assignment and sequencing
  - Standard neighborhood operators
    - E.g. bit-flip, shift, swap moves
- Inflexibility in codes recycling



DESIGN AND ARCHITECTURE

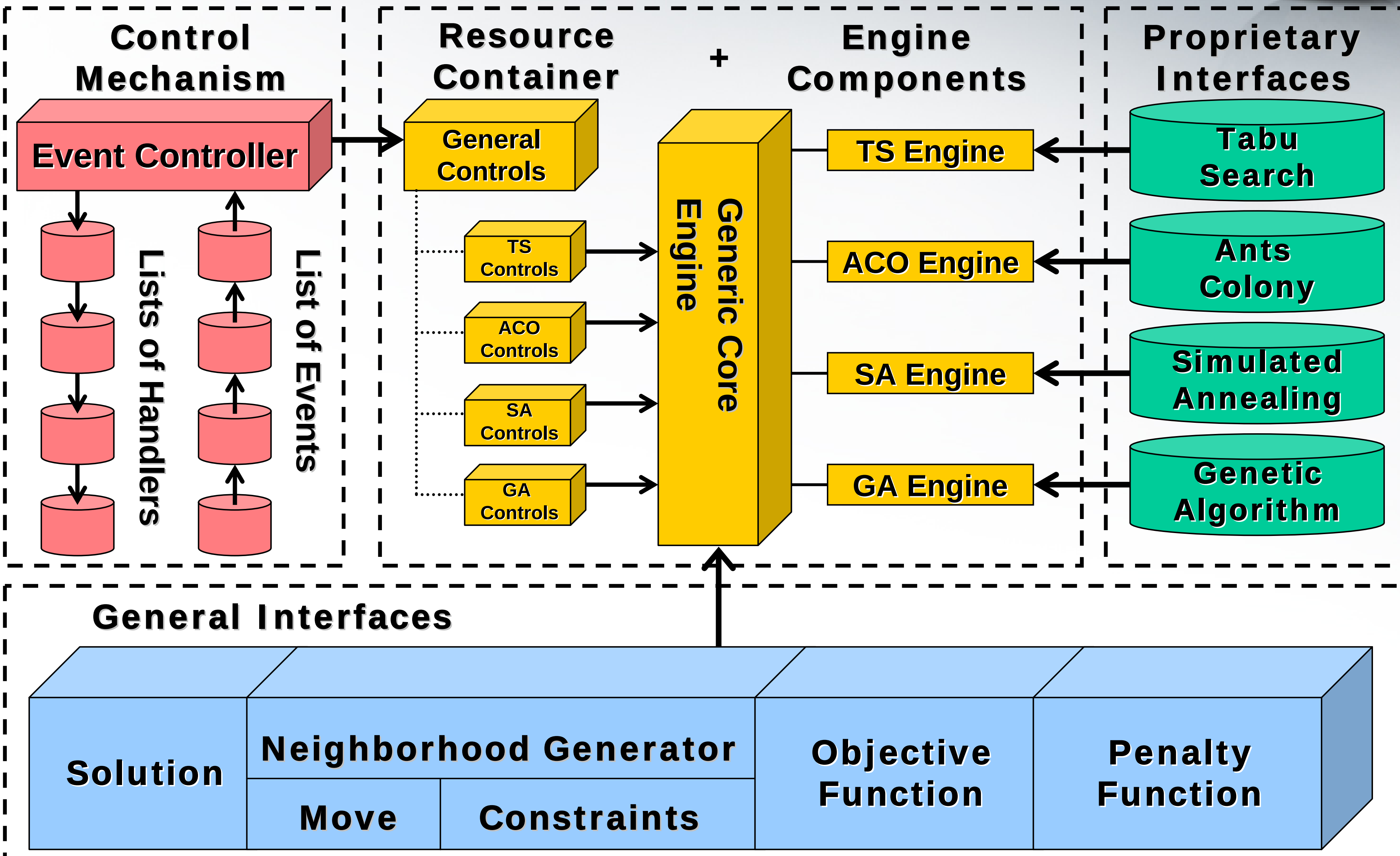
DESIGN AND ARCHITECTURE

❑ General routines of Meta-heuristics



DESIGN AND ARCHITECTURE

Overview of MDF



COMPONENTS OF MDF

General Interfaces

- Solution, Move,Constraint, Neighborhood Generator, Objective Function, Penalty Function

Proprietary Interfaces

- Tabu List and Aspiration Criteria *[Tabu Search]*
- Pheromone Trails and Local Heuristics *[Ant Colony Optimization]*
- Annealing Schedule *[Simulated Annealing]*
- Recombination and Population *[Genetic Algorithm]*

Engines

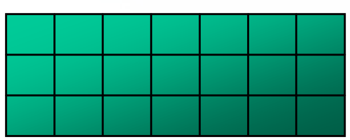
Control Mechanism

- Event
- Handler
- Event Controller

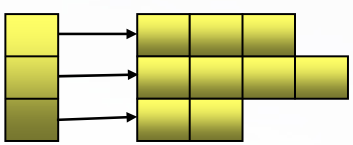
GENERAL INTERFACES

❑ **Solution** Interface

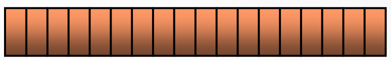
- Solution Representation
  - Make no assumption on the data structure type
  - Data are manipulated indirectly through other interfaces



2D Arrays



Arrays of Lists



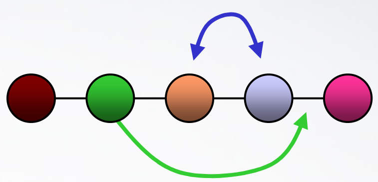
Array of Boolean

- **Cloning** Function
  - *Shallow cloning:* Copying the reference of the object
  - *Deep cloning:* Copying the contents of the object

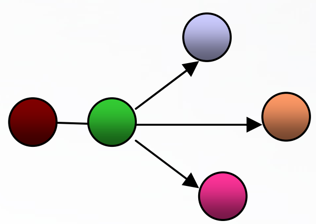
GENERAL INTERFACES

❑ **Move** Interface

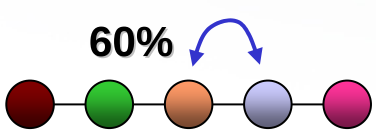
- Define the neighborhood
  - TS: Local Improvement
    - E.g. Exchange, Replace



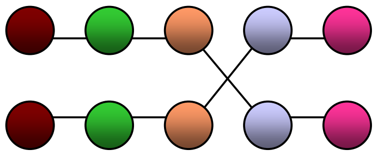
  - ACO: Constructing solution
    - E.g. Increment



  - SA: Probabilistic operation
    - E.g. Probabilistic Swap



  - GA: Recombination
    - E.g. One-point crossover



GENERAL INTERFACES

❑ **Constraint** Interface

- Measure the degree of violation
  - Boolean: Feasible / Infeasible
  - Integer: 0 = Feasible, 1 – n = number of constraints violated

- Useful for
  - Oscillating strategies
  - Dual Model formulation
  - Ranked (hierarchical) constraints based search

GENERAL INTERFACES

❑ **Neighborhood Generator** Interface

- Generate the possible next states
- Use Move and Constraint
  - *Move:* Generates all possible moves
  - *Constraint:* Ensure the moves are desirable

- Has a different “contextual meaning” for different meta-heuristics
  - TS: Generate all neighboring solutions
  - ACO: Generate the next possible paths
  - SA: Generate a random neighboring solution
  - GA: Generate the probability table for next generation selection

GENERAL INTERFACES

❑ **Objective Function** Interface

- Compute the objective value of solution
  - **Absolute** Computation
    - Calculate objective value from scratch
  - **Incremental** Computation
    - Calculate objective value from the previous solution
    - Usually applied to calculate the neighbor objective value from current solution

- Compare the objective values of two solutions
  - Determine maximizing or minimizing the objective value

GENERAL INTERFACES

Penalty Function Interface

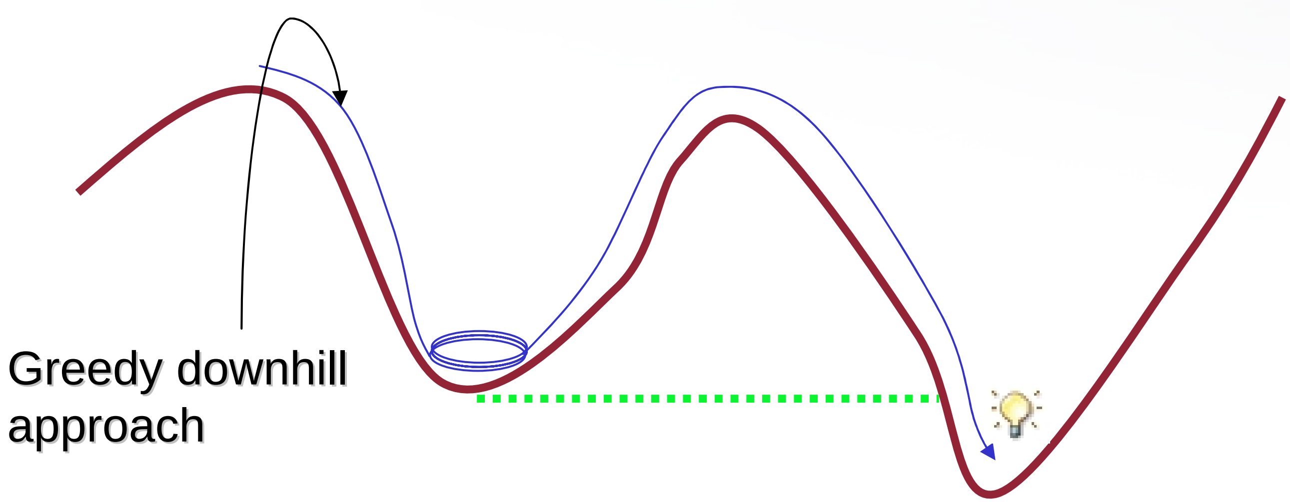
- Implementation of **soft constraints**
  - Some solutions are preferred even if their objective value is slightly lower
- Objective value
  - Temporary modified during comparison
  - Prevent re-application of penalty (bonus)
  - Maintain the correctness of the objective value

PROPRIETARY INTERFACES

❑ **Tabu List** Interface

- Memory technique generally used to reduce cycling
  - Short tenure leads to cycles
  - Long tenure decreases efficiency

Moving downhill



PROPRIETARY INTERFACES

Aspiration Criteria Interface

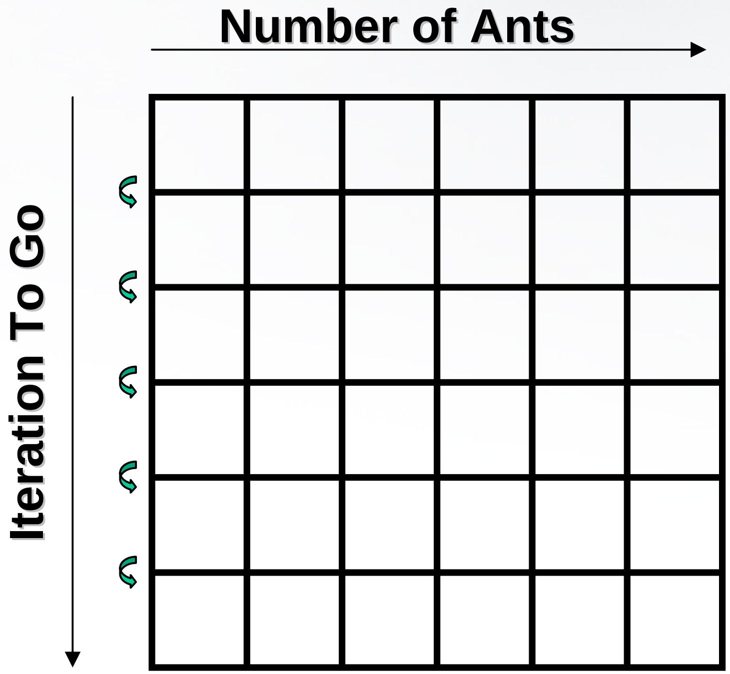
- Override tabu status of a neighbor if it meets certain criteria
  - Objective value is better than best-found solution
  - Difference between current and next states is large
  - Existence of certain sub-optimal structures

- Avoid accidentally missing good solutions
- Improve the search performance

PROPRIETARY INTERFACES

❑ **Pheromone Trails** Interface

- Record the pheromone density of the ants trails
- Influence the behavior of subsequent ants
- Pheromone Update
  - **Local** Decay
    - Enhance exploration
  - **Global** Update
    - Enhance exploitation
  - **Evaporation**
    - Reduce rapid convergence



PROPRIETARY INTERFACES

❑ **Local Heuristics** Interface

- Incorporate the underlying heuristic in solving problem
- Reflect the quality of the path
- E.g. in TSP, local heuristic compute the quality of an arc
  - Q (arc) = 1 / distance
- Can be formulated as a function of multiple factors
  - E.g. Vehicle Routing Problem with Time Windows
    - Vehicle
    - Distance
    - Waiting Time

PROPRIETARY INTERFACES

❑ **Annealing Schedule** Interface

- Acceptance Probability *p*

$$\boxed{p = \text{exponential}\,(-|\Delta x/T_i|)}$$

- Δx : difference in objective values of current and next states
- T_i : cooling temperature
- Modeling T_i
  - If $T_i = 0$, the algorithm becomes greedy
  - If $T_i = \infty$, the algorithm becomes random
  - T_i is usually set to ∞ and gradually decrease to 0

PROPRIETARY INTERFACES

- **Recombination** Interface
  - Combine two individuals to produce two offspring
  - Crossover operators
    - **One-point** crossover
    - **Two-point** crossover
    - **Uniform** crossover
    - **Partially mixed** crossover
  - May incorporate a probability of crossover
    - Encode the probability of two individuals will actually breed

PROPRIETARY INTERFACES

- **Population** Interface
  - Contain individuals in a generation
  - Generate the **First Generation** pool
    - Created randomly or by randomized heuristics
    - Ensure diversity so as to prevent rapid convergence
  - Selection of subsequent generation
    - Sometimes parents are also preserved in the next generation
      - Prevent the loss of “good” genes
      - Lead to over-population
    - Discard the unwanted individuals in the mixed pool
      - “Survival of the fittest” rule
      - Rules can be specified by users

ENGINES

Tabu Search Engine

procedure
Initialize a current *Solution*
while terminating criteria not reached
    Neighborhood Generator generates a new neighborhood;
    Constraint discards any undesired neighbors;
    Objective Function evaluates selected neighbors;
    Penalty Function applied to neighbors;
    Tabu List and *Aspiration Criteria* are consulted;
    Move translates current *Solution* to best neighbor;
    if new *Solution* is better than best found *Solution*
        Clones and records new *Solution* as best found *Solution*;
    Tabu List is updated;
end procedure

ENGINES

Ant Colony Optimization Engine

procedure
Initialize the *Pheromone Trail*
while terminating conditions not reached
    while there is still ants in colony and
        while the *Solution* is not completed
            Neighborhood Generator generates a set of new trails;
            Constraint discards any impassible trails;
            Trail chosen by consulting *Local Heuristic* and *Pheromone Trail*
            Move translates the *Solution* with selected trail;
            Local *Pheromone Trail* Updated
    Objective Function evaluates solutions constructed by ants;
    Penalty Function is applied to determine the quality of solutions;
    Global *Pheromone Trail* is updated;
    If new *Solution* is better than best found *Solution*,
        Clones and records new *Solution* as best found *Solution*;
    Evaporation occurred in *Pheromone Trail*;
end procedure

ENGINES

Simulated Annealing Engine

```
procedure
Initialize a current Solution;
while terminating conditions not reached
    Neighborhood Generator generates a random neighbor;
    Constraint validates the feasibility of neighbor;
    Objective Function evaluates solutions;
    Penalty Function temporary adjusts the objective value;
    If new neighbor is better than current Solution
        Move translates Solution to neighbor;
    Else Consults the Annealing Schedule;
        If neighbor is accepted
            Move translates Solution to neighbor;
        Else    Current Solution remains unchanged;
    If new Solution is better than best found Solution
        Clones and records new Solution as best found Solution;
end procedure
```

ENGINES

Genetic Algorithm Engine

procedure
Initialize the first generation *Population*;
while terminating conditions not reached
- *Neighborhood Generator* selects *Solutions* for mating;
- *Recombination* crosses selected *Solutions* to form new children;
- *Move* mutates new children;
- *Constraint* discards infeasible children;
- *Objective Function* evaluates children;
- Children are mixed into the parent *Population*;
- *Penalty Function* adjusts the objective value of all *Solutions* in *Population*;
- *Population* discards unfit individuals until the population is balanced;
- If any *Solution* in *Population* is better than best found *Solution*
    - Clones and records new *Solution* as best found *Solution*;

end procedure

CONTROL MECHANISM

- Inspire from observing that search strategies enhance meta-heuristic performance
  - Intensification, diversification, hybridization
  - Provide strong motivation for strategy-based framework
- **ALL** strategies are defined by two components
  1. The time point in which the strategy is to be performed
  2. The actions preformed by the strategy
- **Request** and **Response** ***(R&R)*** Metaphor
  - Requests are specific time points in the search (Event)
  - Responses are actions to be performed (Handler)
  - The search process becomes an "***event-driven***" simulation

CONTROL MECHANISM

- **Atomic unit time** concept
  - The smallest unit time for a meta-heuristic to completely perform a set of routines
  - Definition varies across different meta-heuristics

Meta-heuristics	**Atomic unit time Definition**
Tabu Search	An iteration of the search
Ants Colony Optimization	The activity of an ant
Simulated Annealing	Generating a random move
Genetic Algorithm	A new generation formed

CONTROL MECHANISM

❑ **Event** Interface

- Implement the requests of user
  - New Best Solution Found
  - Series of Non-improving Moves
- Many-to-many relationship between events and handlers
  - One event can trigger multiple responses
  - Many events can trigger a same response
- Support three levels of priority
  - **INSTANT:** Execute the handler immediately
  - **NORMAL:** Execute the handler at the end of atomic unit time
  - **DELAYED:** Execute after all higher priority handlers are executed
- Hierarchical levels of priority allows user to control precisely the sequence of execution

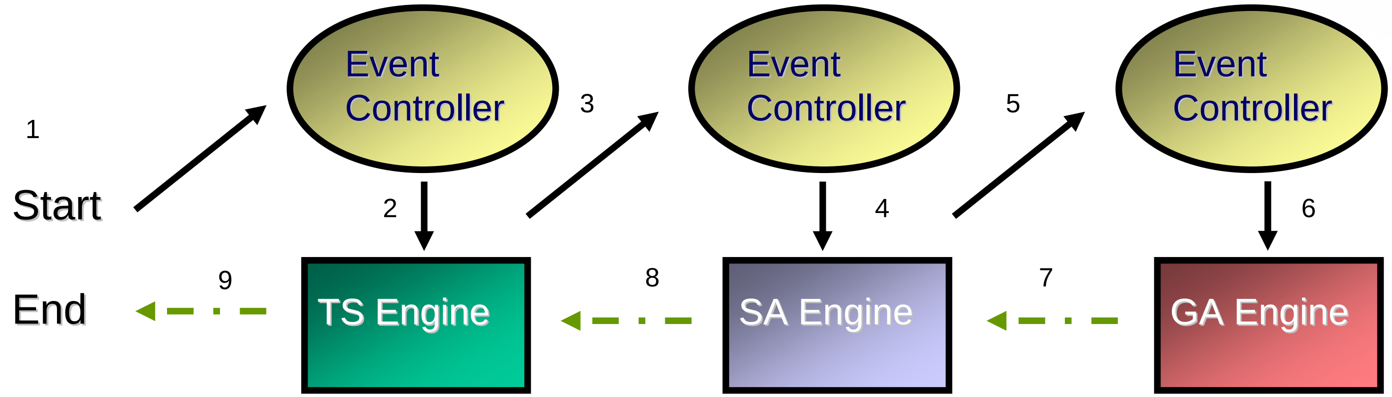
CONTROL MECHANISM

- **Handler** Interface
  - Implement the responses of user
  - **Parameters based** handlers
    - Implement adaptive parameters
      - Reactive Tabu List
      - Dynamic Annealing Schedule
  - **Techniques based** handlers
    - Incorporate additional objects to handle the actions
      - Intensification on Elite Solutions
      - Probabilistic Diversification
      - Hybridization

Control Mechanism

❑ Event Controller Class

- Search State
  - Operating meta-heuristic engine
  - Search parameters
  - Current Solution
- Control the search process by adjusting the search state
- *Special Case:* Hybridization
  - Implement a “Chain of Responsibility”



CASE STUDY:

Traveling Salesman Problems (TSP)

PROBLEM DEFINITION

Traveling Salesman Problem *(TSP)*

Let

$G = (V,A)$ be a graph,

where

$V = \{ v_1, v_2, \dots , v_n \}$ be a set of cities (vertex set),

And $A = \{ (v_i, v_j) : v_i, v_j \in V, i \neq j \}$ be the edge set,

Cost (r, s) = Cost (s, r) *(Symmetry)*

- A tour is defined as a *Hamiltonian* circuit passing exactly once through each point in vertices *V*.
- The *TSP* objective is to find a tour of minimum costs/distance

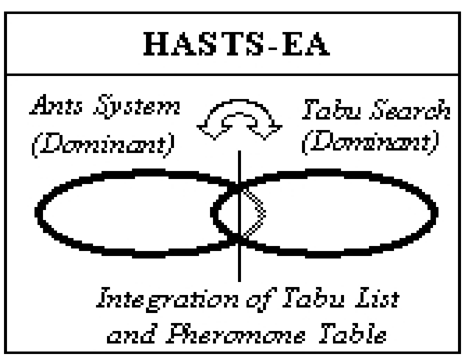
EXPERIMENT PLATFORM

- **System Specification**
  - Processor: Athlon XP 3.2 Ghz
  - Memory: 512 MB
  - Runtime: 90 seconds
    - Hybridize Schemes
    - Problem size
- **Operating System**
  - Window XP
- **Initial Solution**
  - Nearest Cities heuristic (Greedy heuristic)
  - Non-optimality of the last portion of the tour

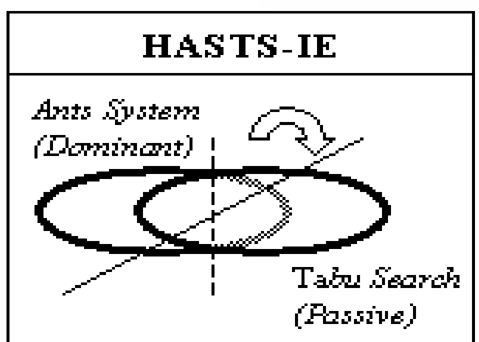
HASTS SCHEMES

❑ Hybrid Ants System and Tabu Search

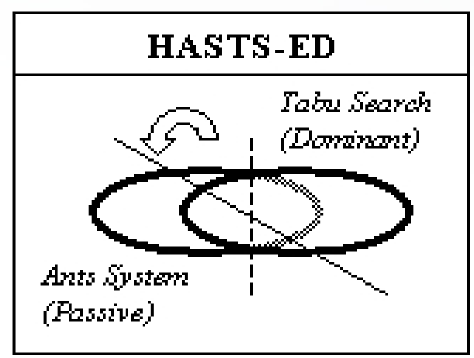
- Flexible hybrid scheme that spawns **four** *derived models*
  - Relative importance level
  - Degree of collaboration
  - “Master-Slave” relationship



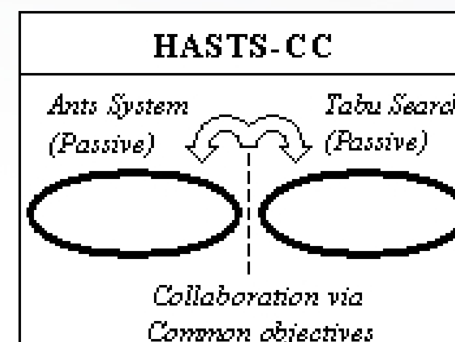
11 Model



10 Model



01 Model



00 Model

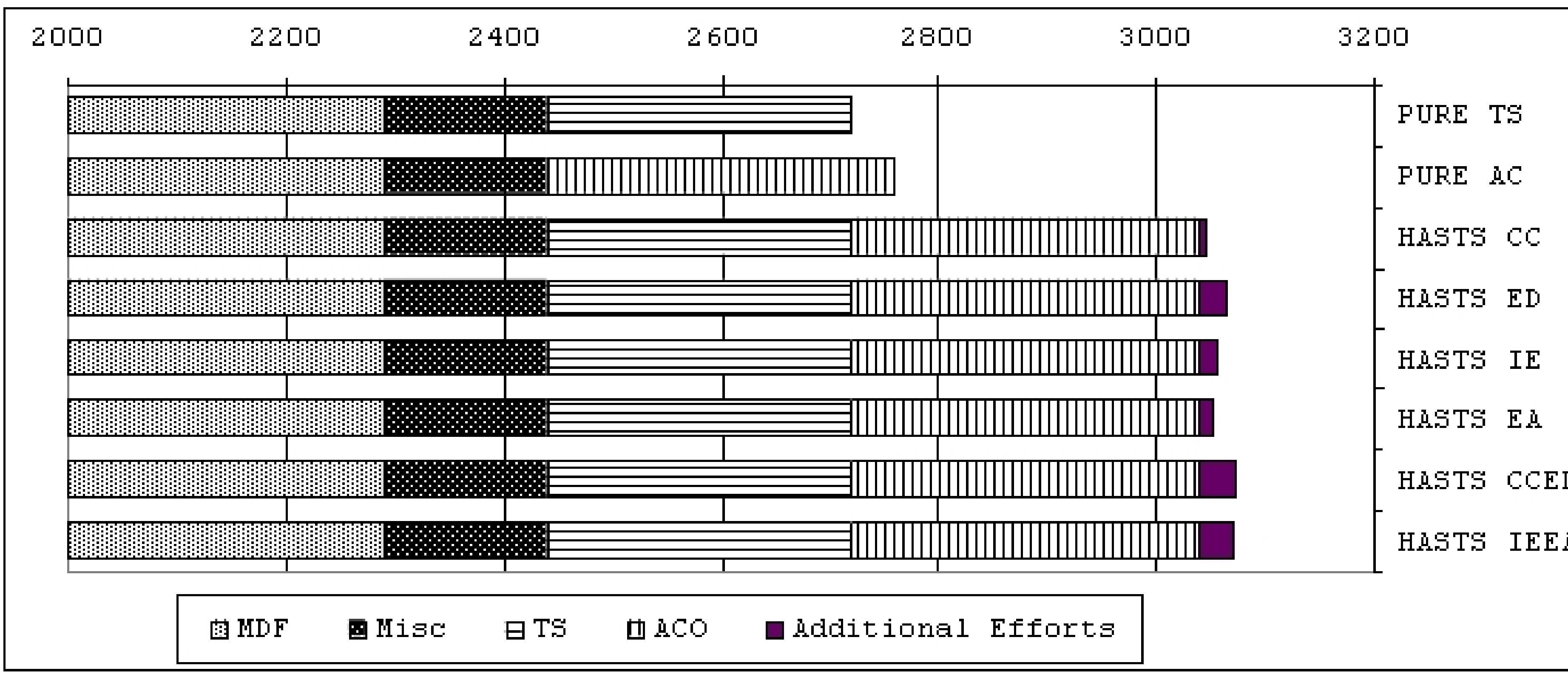
HASTS SCHEMES

HASTS Derived Schemes

- **Pure TS** (*Strict Tabu Search*)
  - Static tabu tenure
- **Pure ACO** (*Ants Colony System*)
  - Update using iteration-best ants
- **HASTS-EA** (*Empowered Ants*)
  - Use both pheromone trails and tabu list
- **HASTS-IE** (*Improved Exploitation*)
  - Use TS to improve on the solution found by iteration-best ant
- **HASTS-ED** (*Enhanced Diversification*)
  - Use ACS as a diversifier
- **HASTS-CC** (*Collaborative Coalition*)
  - Two phase search
- **Hyper Hybrid** Schemes (*Combined*)
  - HASTS-IEEA (IE with EA)
  - HASTS-CCED (CC with ED)

OBSERVATIONS OF RESULTS

- **Development Time** Comparison



OBSERVATIONS OF RESULTS

Name	Bound	Pure TS		Pure ACO		HASTS-EA		HASTS-IE	
Att48	10628	10755	1.19	10847	2.06	10860	2.18	**10628**	**0.00**
eil51	426	427	0.23	430	0.94	430	0.94	427	0.23
Pr76	108159	109186	0.95	111994	3.55	111435	3.03	**108159**	**0.00**
kroA100	21282	21296	0.07	21559	1.30	22092	3.81	**21282**	**0.00**
kroB100	22141	22235	0.42	23145	4.53	22936	3.59	22220	0.36
Wil101	629	**629**	**0.00**	649	3.18	638	1.43	**629**	**0.00**
Ch130	6110	6196	1.41	6492	6.25	6492	6.25	6124	0.23
kroA150	26524	27125	2.27	27682	4.37	27621	4.14	**26550**	**0.10**
kroB150	26130	26178	0.18	27909	6.81	28499	9.07	26132	0.01
d198	16780	15909	0.82	17397	10.25	17213	9.08	**15780**	**0.00**
kroA200	29368	29487	0.41	34087	16.07	35859	22.10	29565	0.67
kroB200	29437	30121	2.32	36980	25.62	36980	25.62	29813	1.28
a280	2579	2669	3.49	3157	22.41	3157	22.41	2598	0.74
Lin318	42029	43123	2.60	52156	24.10	50053	19.09	42777	1.78
pcb442	50778	52025	2.46	61979	22.06	61979	22.06	51873	2.16
STD Deviation			1.11		9.15		9.13		0.70

Case KROA-150 Case LIN-318

OBSERVATIONS OF RESULTS

Name	Bound	HASTS-ED		HASTS-CC		HASTS-CCED		HASTS-IEEA	
Att48	10628	**10628**	**0.00**	10653	0.24	**10628**	**0.00**	**10628**	**0.00**
eil51	426	**426**	**0.00**	**426**	**0.00**	**426**	**0.00**	**426**	**0.00**
Pr76	108159	**108159**	**0.00**	**108159**	**0.00**	**108159**	**0.00**	**108159**	**0.00**
kroA100	21282	**21282**	**0.00**	**21282**	**0.00**	21292	0.05	**21282**	**0.00**
kroB100	22141	22210	0.31	22200	0.27	22271	0.59	**22141**	**0.00**
wil101	629	**629**	**0.00**	**629**	**0.00**	**629**	**0.00**	**629**	**0.00**
ch130	6110	6128	0.29	6150	0.65	**6113**	**0.05**	**6113**	**0.05**
kroA150	26524	26767	0.92	26727	0.77	26762	0.90	**26525**	**0.00**
kroB150	26130	26152	0.08	26860	2.79	26391	1.00	**26130**	**0.00**
d198	16780	16876	0.61	15796	0.10	15799	0.12	15781	0.01
kroA200	29368	29668	1.02	29487	0.41	29603	0.80	**29479**	**0.38**
kroB200	29437	30121	2.32	30121	2.32	30121	2.32	**29543**	**0.36**
a280	2579	2658	3.06	2669	3.49	2654	2.91	**2579**	**0.00**
lin318	42029	42938	2.16	43123	2.60	43083	2.51	**42665**	**1.51**
pcb442	50778	51860	2.13	52025	2.46	51955	2.32	**51654**	**1.73**
STD Deviation			1.05		1.26		1.07		**0.56**

Case KROA-150 Case LIN-318

PROJECT SUMMARY

PROJECT SUMMARY

1. *We present a wide discussion on the current* ***state-of-art*** *meta-heuristics and their techniques*

2. *We present a novel approach of characterizing different meta-heuristics into their common behavior, which consequently enables codes reuse across different meta-heuristics*

3. *We describes the design and realization on how meta-heuristics can adopts a* ***Request and Response (R&R)*** *model that facilities the formation hybridized schemes and related strategies*

QUESTIONS

SUPPLEMENTARY MATERIALS

HASTS SCHEMES

Pure TS

- Implement ***Strict Tabu Search***
  - Static Tabu tenure

Solution:	Single-dimension integer array
Move:	Swap-edge operator
Constraint:	Not applicable
Neighborhood:	${}^{N}C_2$ *pairs*
Objective:	Sum of distance
Penalty:	Not applicable
Tabu List:	Tabu the “Moves” made
Aspiration:	Best-ever aspiration

HASTS SCHEMES

Pure ACO

- Implement ***Ants Colony System***
  - Incorporate Exploitation and Exploration factor *q0*
  - Update trails with the Iteration-best ants

Solution:	Single-dimension integer array
Move:	Incremental operator
Constraint:	Not applicable
Neighborhood:	List of Unvisited Nodes
Objective:	Sum of distance
Penalty:	Not applicable
Local Heuristic:	*1 / distance*
Pheromone :	On arcs of every city

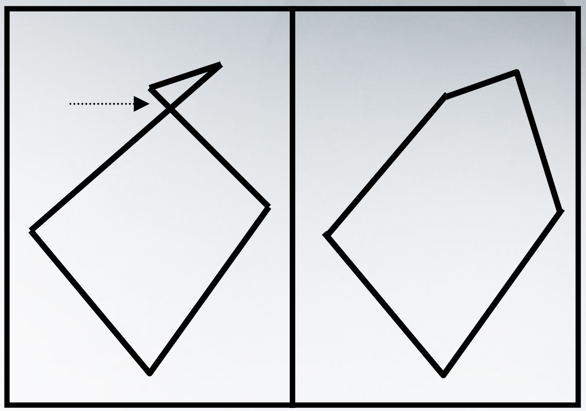
HASTS SCHEMES

❑ HASTS-EA (Empowered Ants)

- Inspired from
  - *Pheromone trail:* Preference Table for intensification
  - *Tabu List:* Forbidden List for diversification
- Design:
  - Embed Tabu List into ACO Neighborhood generator
  - Tabu List prevents ants in the same iteration to construct same solutions
  - Encourage both intensification and diversification
  - Useful when there are many local optimal
- Events
  - None
- Handlers
  - None

HASTS SCHEMES

HASTS-IE (Improved Exploitation)



- Inspired from
  - ACO suffer from “crossing” of edges in the tour
  - “Over deposition” of pheromone
- Design:
  - TS removes “crossing” of edges from the tour
  - Apply TS at the end of ACO iterations
  - Improve Exploitation
- Events
  - End of ACO iterations before the pheromone update
- Handlers
  - Apply TS on the iteration-best solution

HASTS SCHEMES

HASTS-ED (Enhanced Diversification)

- Inspired from
  - TS suffer from cycling
  - Prominent with static tenure
- Design:
  - ACO as a probabilistic diversifier
    - Randomly destroy sub-routes for reconstruction
  - Apply ACO to TS if *n* non-improving solutions are encountered
  - Enhance diversification
- Events
  - TS made *n* number of non-improving moves
- Handlers
  - Apply probabilistic diversification on best-found solution

HASTS SCHEMES

HASTS-CC (Collaborative Coalition)

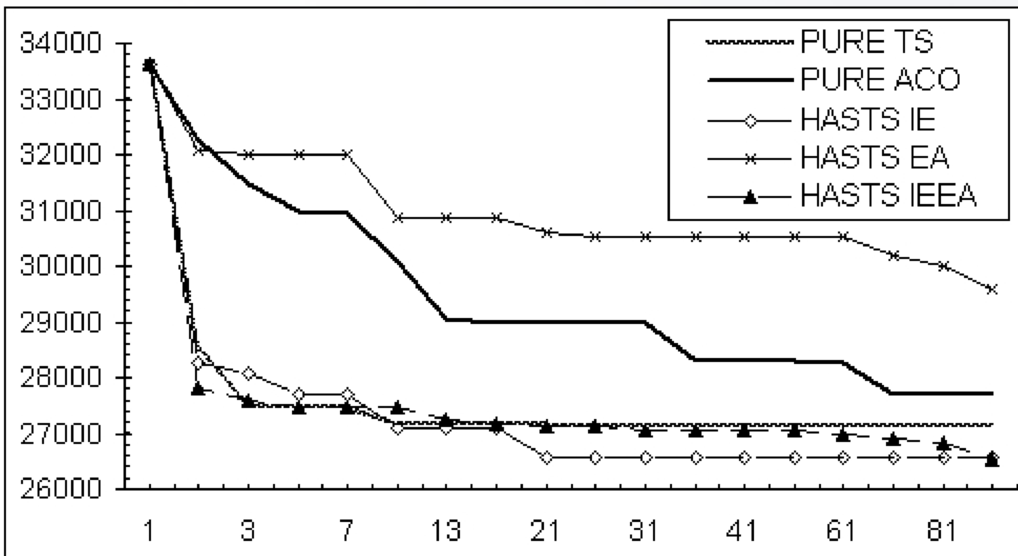
- Inspired from
  - ACO is an excellent constructing heuristic
  - TS is an excellent optimizing heuristic
- Design:
  - Two phase approach
    - ACO constructs a initial solution
    - TS then optimizes on the solution
  - Apply TS when all of ACO iterations are completed
- Events
  - ACO Engine stopped
- Handlers
  - Apply TS on ACO best-found solution

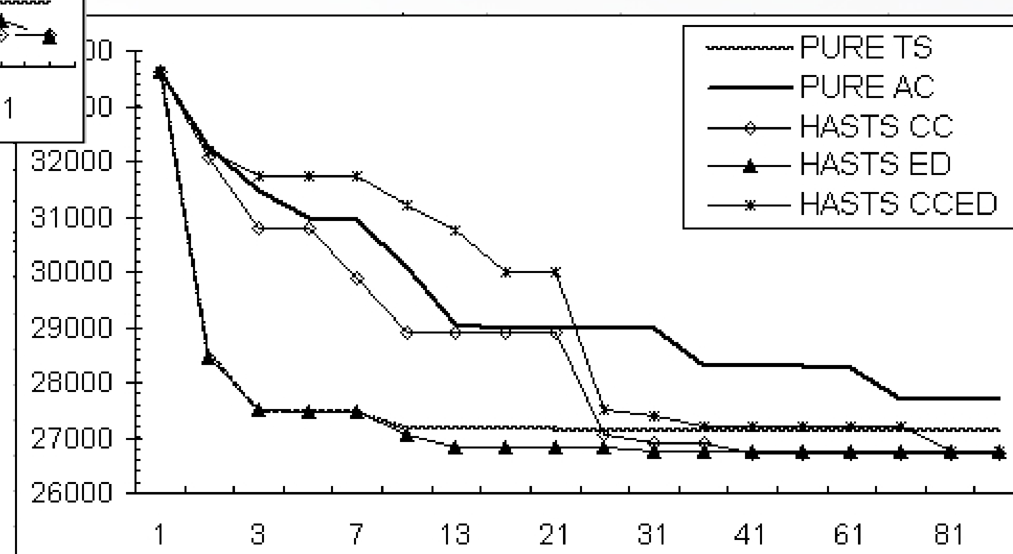
HASTS SCHEMES

- **Two naïve hyper-hybrid schemes**
  - **HASTS-IEEA** Hyper-hybrid
    - Combine HASTS-IE with HASTS-EA
    - Fuses tabu list from HASTS-EA into HASTS-IE
    - Combined hybrid has a more aggressive diversifying capability
  - **HASTS-CCED** Hyper-hybrid
    - Combine HASTS-ED with HASTS-CC
    - TS of HASTS-CC is replaced with HASTS-ED
    - Combined hybrid has enhanced optimizing phase

OBSERVATIONS OF RESULTS

❑ Case Examination: KROA-150





OBSERVATIONS OF RESULTS

❑ Case Examination: LIN-318