GRANADA, JUEVES 10 DE MAYO 2012

Universidad de Granada

RED LOC

RED DE LOCALIZACIÓN Y PROBLEMAS AFINES

MATHEURISTICS: Hybridizing Metaheuristics and Mathematical Programming

Introduction: Metaheuristics
Some Main Parts:
The Corridor Method
An Application?
Conclusions and Outlook
(And then some new talks?)

Marco Caserta, Stefan Voß (University of Hamburg)
Moshe Sniedovich (University of Melbourne)

Introduction: Practical Point of View

Practice: great variety of discrete optimization problems
- hard, approximate modeling, imprecise data, ...

⇒heuristics

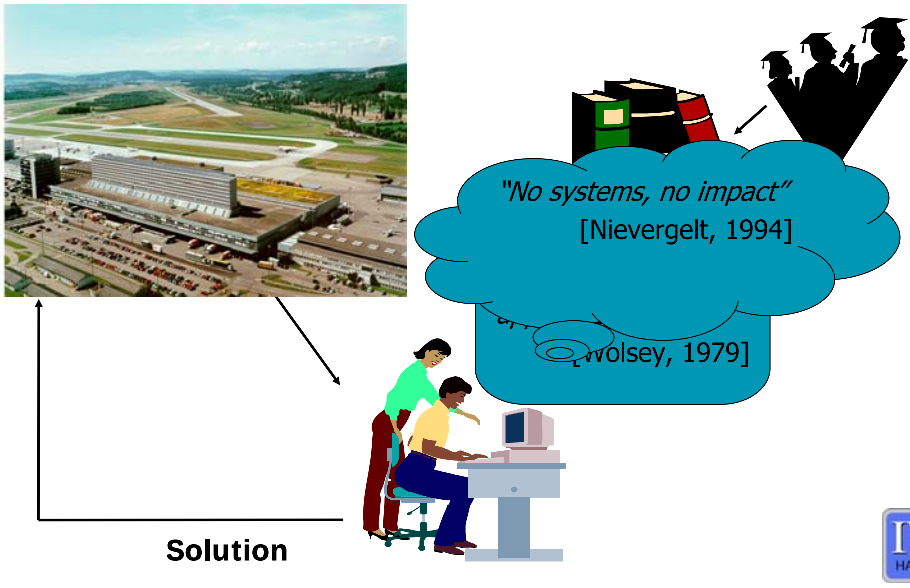
Ultimate goal: decision support concerning real-world problems
- many problems, which are diverse and change over time
- in practice we need easy-to-use application systems

⇒we need support for the development of corresponding software

Strategy: build reusable and robust software components for heuristic search that embody the knowledge of research

IWI HAMBURG

New Real-world Problem — Research



3 © Stefan Voß

What We Aim For in This Talk

Discuss where we are and possible options for advancing metaheuristics.
Solvers have become so powerful that it pays to "MIP it," but …

There is not only just one method!

Examples:

- "Evolutionary …," "Ants…" etc.
- Tabu Search ≠ Tabu Search
- Pilot Method
- Popmusic

Where could this community go next?

Overall: Give some simple (and hopefully yet effective) ideas; food for thought.

4 © Stefan Voß

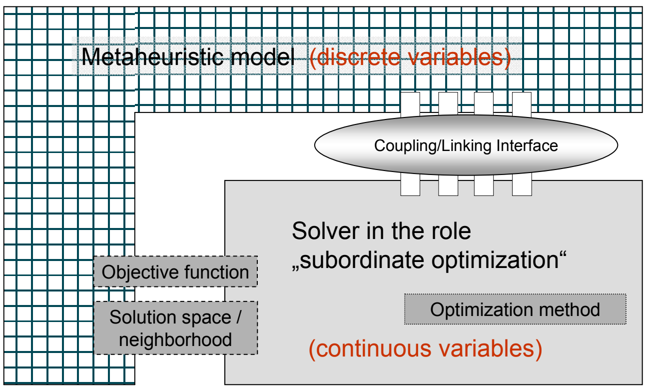
RED LOC
RED DE LOCALIZACIÓN Y PROBLEMAS AFINES

By the Way:

"Tabu search has its antecedents in methods designed to cross boundaries of feasibility or local optimality standardly treated as barriers, and to systematically impose and release constraints to permit exploration of otherwise forbidden regions." (Glover/Laguna)

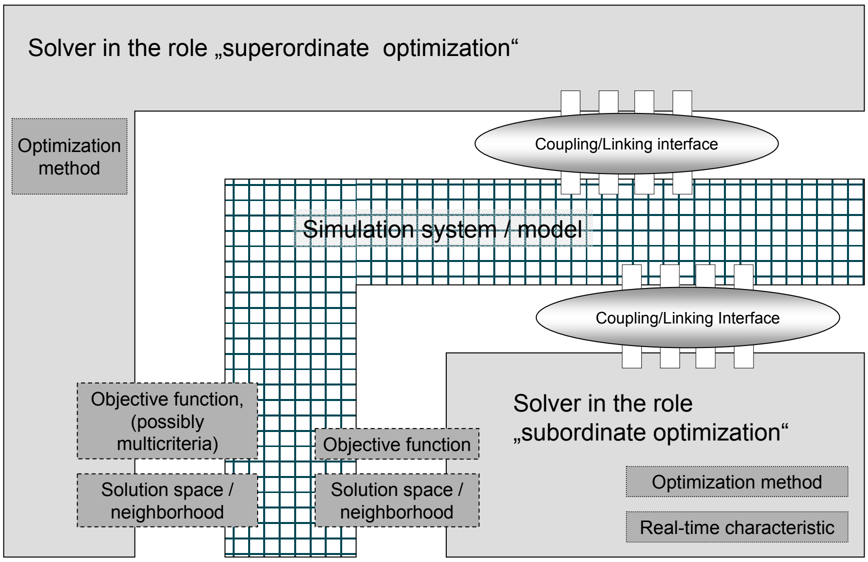
Stefan Voß **(University of Hamburg)**

Metaheuristic and Optimization



IWI HAMBURG

Simulation and Optimization



Matheuristics are made by the interoperation of metaheuristics and mathematical programming (MP) techniques. An essential feature is the exploitation in some part of the algorithms of features derived from the mathematical model of the problems of interest, thus the definition "*model-based metaheuristics*" appearing in the title of some events of the conference series dedicated to matheuristics.

Special Issues on Matheuristics:

Journal of Heuristics, 15 (3), 2009

Annals of Information Systems, 10, 2009

Discrete Applied Mathematics, under review

Hybridization

IWI HAMBURG

What does it mean: "Hybridizing" ?

Some method + Local search ?

Some method + Discrete event simulation ?

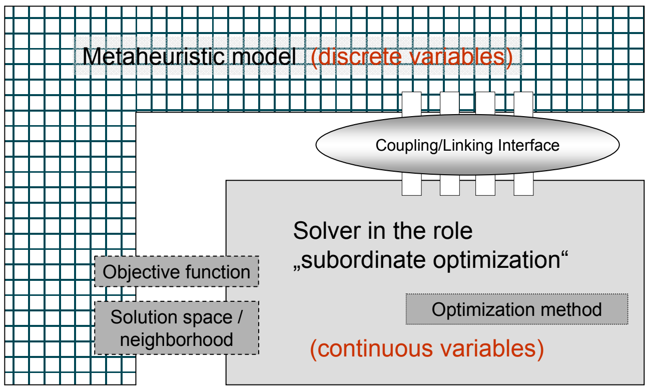
Some method + Some other method ?

- e.g.: metaheuristic + metaheuristic ?
- e.g.: metaheuristic + exact algorithm (e.g. B&C) ?

More ??

IWI HAMBURG

Metaheuristic and Optimization

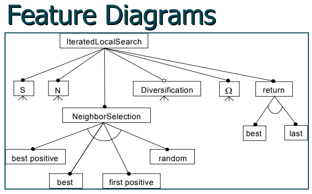


Domain Analysis: Metaheuristics

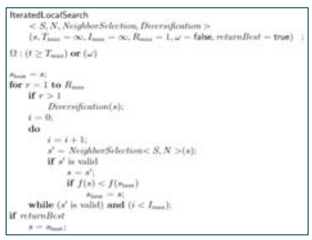
POPMUSIC

- Taillard/Voß (2002). In: C.C. Ribeiro and P. Hansen (Eds.), *Essays and Surveys in Metaheuristics*, Kluwer, Boston (2002).

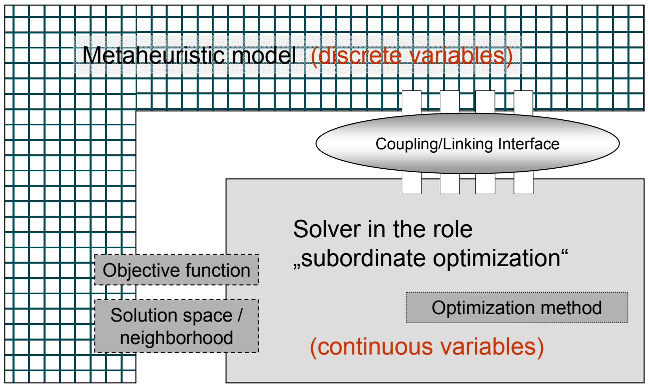
Feature Diagrams



Pseudo Code



Metaheuristic and Optimization



13 © Stefan Voß

Move-based vs. Method-based Neighborhoods
(Paradigms?)

Move-based neighborhoods: Neighborhood search based on topological concepts relating a given solution to similar solutions via local changes or moves. (Example: small homogeneous neighborhoods, very large scale neighborhoods)

1. Define a neighborhood
2. Find efficient methods for its exploration

Method-based neighborhoods: The basic structure of a neighborhood is determined by the needs and requirements of the method used to search it.

1. Assume a given method.
2. Define a neighborhood that suits the method.

(Model-based neighborhoods)

14 © Stefan Voß

The Corridor Method

IWI HAMBURG

The Corridor Method (CM) Sniedovich/Voß (2006)

The CM can be described as a local search method where neighborhoods are relatively large sets whose structure and size are compatible with the optimization method operating on it.

- The optimization problem under consideration is *large*.
- There is an optimization method for efficiently solving smaller instances of the problem.
- It is easy to generate (an initial) feasible solution.
- There is an efficient method for generating suitably large neighborhoods around feasible solutions to the problem on which the optimization method can be used.

M. Sniedovich, S. Voß.
The Corridor Method: a Dynamic Programming Inspired Metaheuristic.
Control and Cybernetics, 35(3):551-578, 2006.

IWI HAMBURG

An Application

Blocks Relocation

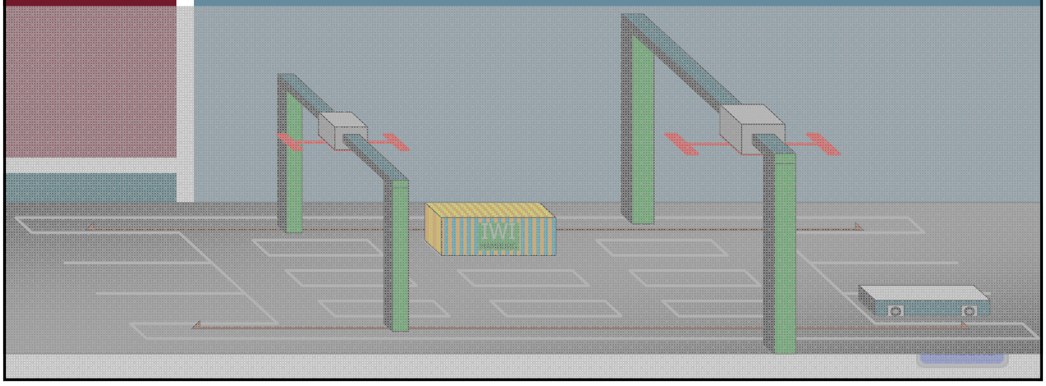
17 © Stefan Voß

IWI HAMBURG

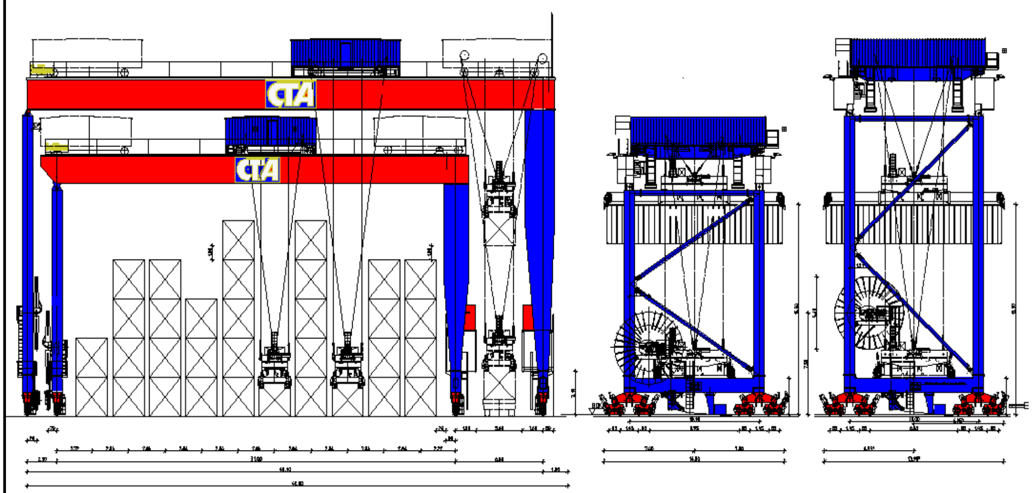
Example
(Container Terminal Altenwerder (CTA), Hamburg, Germany)

18 © Stefan Voß

“Double Rail-Mounted Gantry Cranes (DRMG)”



DRMG



Backoffice

The Blocks Relocation Problem (BRP)

<u>Given</u>

A bay with *n* blocks which have to be retrieved

<u>Assumptions</u> (Kim and Hong, 2006)

Retrieval order is given (1, 2, …, n)

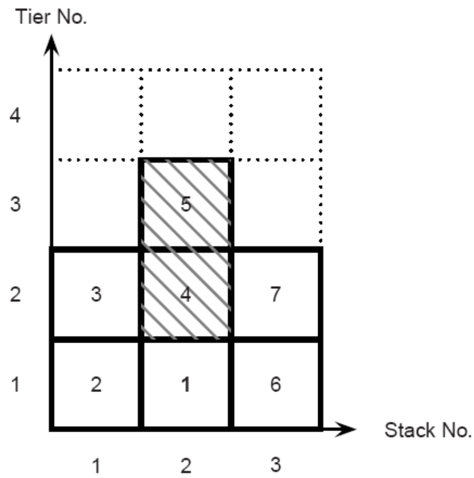
LIFO

Only the upper blocks may be moved

No pre-marshalling

<u>Objective</u>

Retrieval order such that the number of relocations is minimized



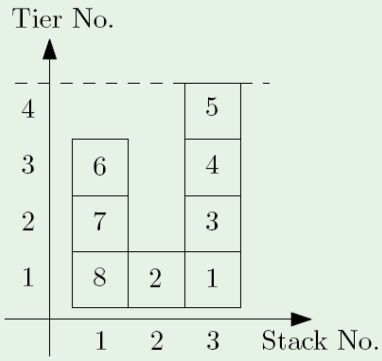
The Blocks Relocation Problem (BRP)

Assumptions discussion

Assumptions (Kim and Hong, 2006)

Only the upper blocks may be moved

No pre-marshalling



Four relocations are needed to clear the stacking area:
Move block 2 first to stack 1.

But: This is forbidden by assumption.

With assumption: Six relocations are needed.

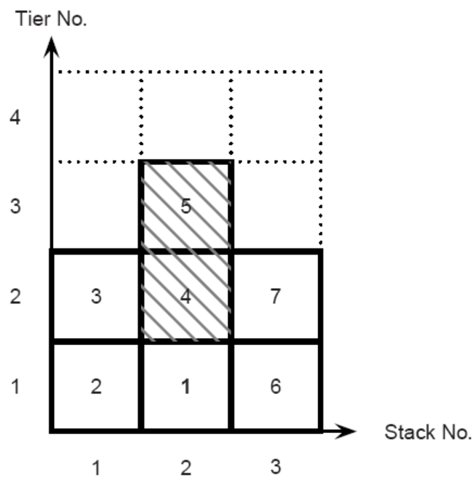
The Blocks Relocation Problem (BRP)

Dynamic Programming Formulation

State variable: s = (k, i, t,C)

k ∈ {1, . . . , n} is the block to be retrieved
i ∈ {1, . . . ,m} is the stack in which the target block is found
t is the list of blocks above the target block
C is the configuration of the remaining blocks

(Example: k = 1, i = 2, t = {5, 4}, and C = {{3, 2} , {7, 6}})



The Blocks Relocation Problem (BRP)

Dynamic Programming Formulation

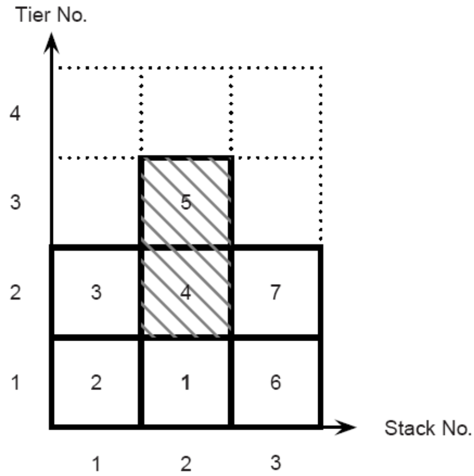
Decision variable: at each step two different cases may arise

(i) the target block has no other blocks placed above, i.e., t = ∅ → retrieve the target block

(ii) at least one block is still above the target block

Let τ be the uppermost block in the sequence t. The decision is about identifying which stack x block τ should be relocated to. Let us indicate with D(s) the set of all feasible values of x with respect to the current state

(Example: τ = 5 and D(s) = {1, 3})



IWI HAMBURG

The Blocks Relocation Problem (BRP)

Dynamic Programming Formulation

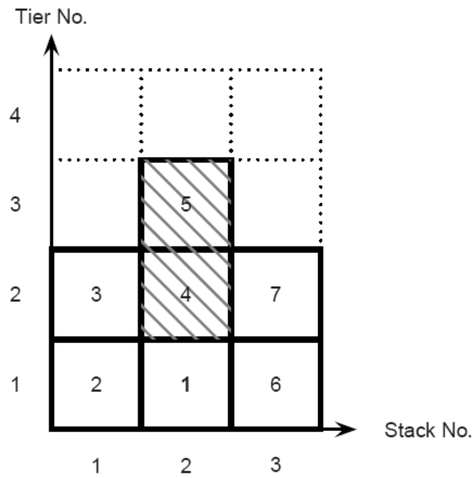
State transition function T(s, x):

Let s′ = (k′, i′, t′,C′) be the state obtained by applying decision x ∈ D(s) to the current state s.

s′ = T(s, x)

(i) t = ∅: k′ = k + 1, i′ is the stack in which block k+1 is currently located, t′ is the list of blocks above k + 1 and C′ = C is the configuration of the remaining blocks

(ii) k′ = k, i′ = i, t′ = t \ {τ}, and C′ depends on the application of move x to block τ (e.g., with respect to the figure let us suppose that x = 1 → s′ = T(s, 1) = (1, 2, {4} ,C′), where C′ = {{5, 3, 2} , {7, 6}})



IWI HAMBURG

Dynamic Programming

State Variable : $s = (k, i, t, C)$, with k target block, i stack of target block, t list of blocks above the target block, and C configuration of remaining blocks;

Decision Variable : if τ is the uppermost block in sequence t, x indicates which stack block τ is moved to ($D(s)$ is the set of all feasible values of x w.r.t. the current state s);

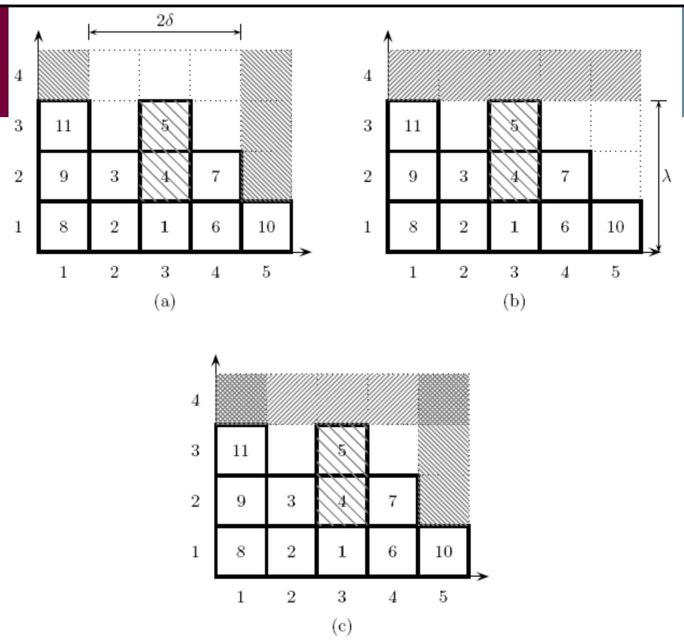
State Transition Function : a function T such that $s' = (k', i', t', C')$ is the state obtained by applying decision $x \in D(s)$ to the current state s, which is, $s' = T(s, x)$;

Functional Equation : DP "backward" functional equation

$$f(k,i,t,C) = \begin{cases} 1 + f(k+1, i', t', C), & t = \emptyset, \\ 1 + \min\limits_{x \in D(k,i,t,C)} \left\{ f(k, i, t \setminus \{\tau\}, C') \right\}, & t \neq \emptyset, \end{cases}$$

with $f(n, i, \emptyset, C) = 1$.

The Corridor



$$D(s, \delta, \lambda) = \{x \in \{1, \dots, m\} \setminus \{i\} : i - \delta \le x \le i + \delta, \ |c_x| < \lambda\}$$

Numerical Results

29 © Stefan Voß

Numerical Results

Bay Size		KH		CM			Corridor	
h	m	No.	Time†	No.	Time†	γ	δ	λ
3	3	7.1	0.1	5.4	0.10	0.00	1	4
3	4	10.7	0.1	6.5	0.10	0.00	1	4
3	5	14.5	0.1	7.3	0.10	0.00	1	4
3	6	18.1	0.1	7.9	0.15	0.00	2	4
3	7	20.1	0.1	8.6	0.10	0.01	2	4
3	8	26.0	0.1	10.5	0.20	0.01	2	4
4	4	16.0	0.1	9.9	0.20	0.02	2	5
4	5	23.4	0.1	16.5	0.50	0.01	2	5
4	6	26.2	0.1	19.8	0.50	0.03	2	5
4	7	32.2	0.1	21.5	0.50	0.03	2	5

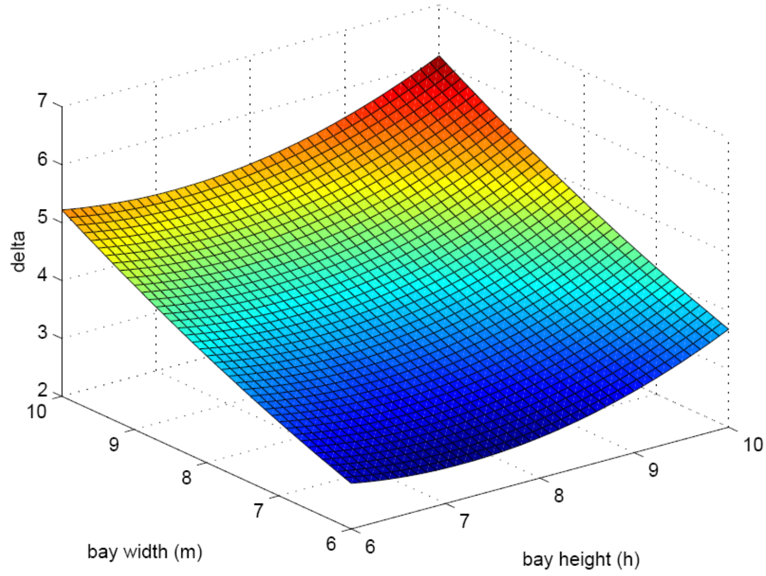
30 © Stefan Voß

Numerical Results

Bay Size		KH		CM		Corridor	
h	m	No.	Time†	No.	Time†	δ	λ
5	4	23.7	0.1	16.6	0.5	2	6
5	5	37.5	0.1	18.8	0.8	2	6
5	6	45.5	0.1	22.1	0.8	2	6
5	7	52.3	0.1	25.8	1.43	1	7
5	8	61.8	0.1	30.1	1.46	1	6
5	9	72.4	0.1	33.1	1.41	1	6
5	10	80.9	0.1	36.4	1.87	1	6

IWI HAMBURG

Response surface: δ = f(h,m).



IWI HAMBURG

Future Research (Example)

Redundancy Allocation Problem (RAP)

Allocation of redundant components within series-parallel systems

m is the total number of components within the system
r_{ij} is the reliability of component j within subsystem I
g is an increasing function describing a capacity constraint
Two different classes of problems: k=1 series-parallel systems, at least one k >1
x_{ij} = 1 indicates that component j of subsystem i is in the solution.

$$
\text{RAP:} \quad \left|\begin{array}{lll}
\max & R = \prod_{i=1}^{n} \left(1 - \prod_{j=1}^{m_i} (1 - r_{ij})^{x_{ij}} \right) & \\
\text{s.t.} & g_q(\mathbf{x}) \leq b_q & q = 1, \ldots, Q \\
 & \sum_{j=1}^{m_i} x_{ij} \geq k_i & i = 1, \ldots, n \\
 & \mathbf{x} \in \mathbb{B}^m &
\end{array}\right.
$$

Redundancy Allocation Problem (RAP)

Optimal allocation of redundant components within series-parallel systems

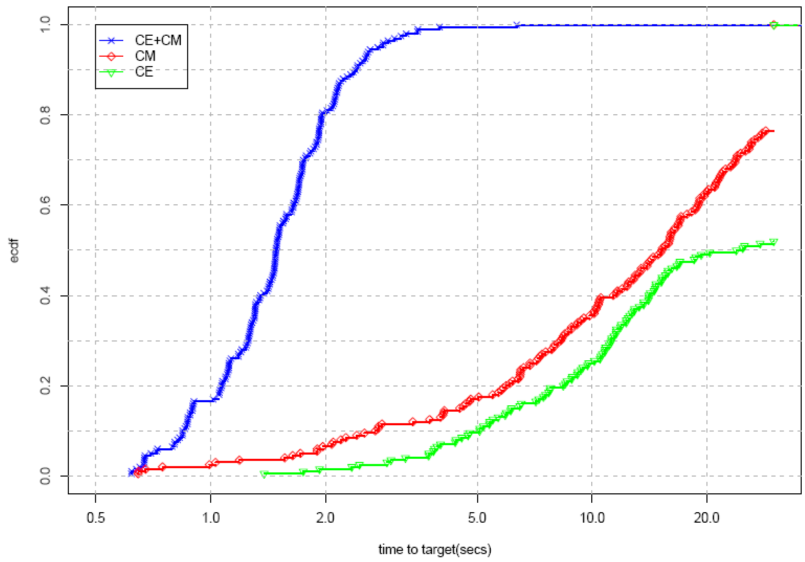
You and Chen (2005)
Computers & Operations Research

No.	W	C	Best Found[a]	Time[b]
1	191	130	0.98681	2.93315
2	190	130	0.98642	3.36518
3	189	130	0.98592	3.55760
4	188	130	0.98538	3.07483
5	187	130	0.98469	4.61933
6	186	129	0.98418	3.31069
7	185	130	0.98350	4.73182
8	184	130	0.98299	4.17582
9	183	129	0.98226	4.17857
10	182	130	0.98152	2.07761
11	181	129	0.98103	4.81956
12	180	128	0.98029	4.06561
13	179	126	0.97950	3.34148
14	178	125	0.97840	2.83650
15	177	126	0.97760	3.04934
16	176	124	0.97669	1.08522
17	175	125	0.97571	4.82758
18	174	123	0.97493	4.71314
19	173	122	0.97383	2.03475
20	172	123	0.97303	5.06500
21	171	122	0.97193	4.33127
22	170	120	0.97076	4.16222
23	169	121	0.96929	4.58145
24	168	119	0.96813	1.75531
25	167	118	0.96634	4.39054
26	166	116	0.96504	2.45814
27	165	117	0.96371	4.32633
28	164	115	0.96242	3.00410
29	163	114	0.96064	5.04348
30	162	115	0.95919	1.68283
31	161	113	0.95803	2.56656
32	160	112	0.95571	2.07888
33	159	110	0.95456	1.22496

a: Values reported are all global optimum.
b: Wall-clock time measured in seconds.

Redundancy Allocation Problem (RAP)

Allocation of redundant components within series-parallel systems



The DNA Sequencing Problem

A well studied problem in computational molecular biology is the DNA sequencing problem. The goal is to determine the order in which sequences of nucleotides appear in an unknown fragment of the DNA.

Modeling using the Orienteering problem (OP)

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_{ij}^{i} x_{ij} \geq \lfloor \delta z^{i} \rfloor$$

A value of $\delta = 0$ makes the inequality redundant and, therefore, corresponds to eliminating the corridor, *i.e.*, to solving the original OP. Growing values of δ generate tighter corridors around the incumbent. Finally, a value of $\delta = 1$ generates the smallest possible corridor around the incumbent. Such corridor can be seen as a measure of maximum diversity allowed, in the sense that only solutions that share at least a certain amount of similar features with the incumbent are considered.

37 © Stefan Voß

IWI HAMBURG

The DNA Sequencing Problem

Problem Formulation: Orienteering ($\mathcal{NP}$-hard)

$$\max z(OP) = \sum_{v_i \in V} p_i y_i$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{v_j \in V \setminus \{v_i\}} x_{ij} = y_i, \quad v_i \in V && (1)\\
& \sum_{v_j \in V \setminus \{v_i\}} x_{ji} = y_i, \quad v_i \in V && (2)\\
& \sum_{(v_i, v_j) \in A} c_{ij} x_{ij} \leq c_{\max} && (3)\\
& 1 \leq u_i \leq m, \quad v_i \in V && (4)\\
& u_i - u_j + 1 \leq m\,(1 - x_{ij}), \quad (v_i, v_j) \in V && (5)\\
& x_{ij} \in \{0, 1\}, \quad (v_i, v_j) \in A && (6)\\
& u_i \in \mathbb{N}, \quad v_i \in V && (7)\\
& 0 \leq y_i \leq 1, \quad v_i \in V && (8)
\end{aligned}$$

38 © Stefan Voß

IWI HAMBURG

The DNA Sequencing Problem

S1. Initialization

- Generation of incumbent solutions using Cross Entropy $\{\pi^I\}$
- If *distance*$(\pi^I, \Omega) > \tau$, add π^I to Ω.

S2. Corridor Method$(\Omega, \delta, \delta^m)$

- Add the *corridor constraint* and solve the resulting MIP problem.
- Stopping criteria of the MIP solver:
  - maximum running time
  - maximum number feasible solutions
- π^c is the best solution found in $\mathcal{N}_\delta(\pi^I)$

S3. Update$(\pi^I, \delta, \delta^m)$

- If π^c is better than π^I, then set π^c as incumbent and go to **S2.**
- If no solution better than the incumbent has been found:
  - enhance the corridor and go to S2.; or
  - If corridor is too large, stop the corridor method and select another incumbent from Ω

The DNA Sequencing Problem

Corridor Definition and Calibration

Dynamic corridor definition

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_{ij}^{i} x_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{m}\left(1 - x_{ij}^{i}\right)(1 - x_{ij}) \geq \delta n$$

- $\delta \in [0, 1]$

Corridor Calibration

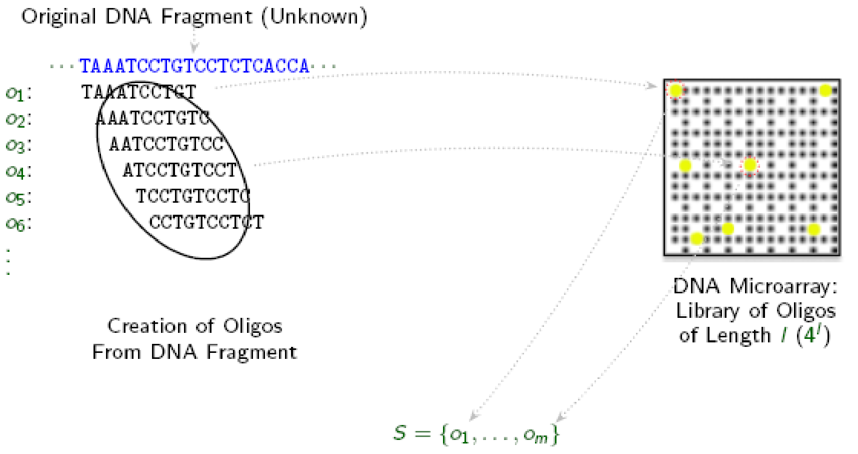
(i) improving solution is found within the corridor: δ is unchanged

(ii) no improving solution is found within corridor:

(ii.a) if $0.9\delta \geq \delta^m$, then $\delta \leftarrow 0.9\delta$; else

(ii.b) restart algorithm using a different incumbent from Ω

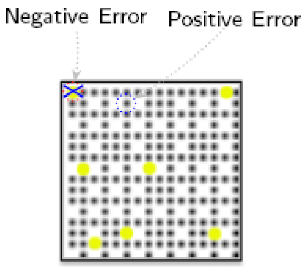
The DNA Sequencing Problem



Spectrum of a DNA fragment: set of oligonucleotides that hybridize with the DNA fragment

The DNA Sequencing Problem

DNA Sequencing Problem with Errors



- Positive Errors: inclusion into the *spectrum* of an oligo that is not in the original sequence.
- Negative Errors: exclusion from the *spectrum* of an oligo that is in the original sequence.

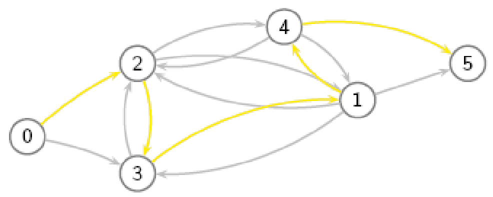
The Optimization Problem

Given an *unknown* DNA fragment of length n and a spectrum $S = \{o_1, o_2, \dots, o_m\}$, find:

- the permutation of elements of S, *i.e.*, $\pi(S)$ such that the maximum number of oligos from S is used without exceeding length n, *i.e.*,
- the permutation of oligos from S that has the maximum *alignment score* with the fragment.

The DNA Sequencing Problem

Problem Formulation: Orienteering



Linking Oligos

- o_i =TAAATCCTGT
- o_j = CCTGTCCTGT

Overlapping Degree Between o_i and o_j

o_i: TAAAT $\boxed{\text{CCTGT}}$

o_j: $\boxed{\text{CCTGT}}$ CCTGT

$\Rightarrow od(o_i, o_j) = 5$

DNA Sequencing as Orienteering Problem

- $G = (V, A)$:
  - V: set of nodes (oligos in S)
  - $A = \{v_i, v_j\}$: set of arcs (oligos o_i followed by oligo o_j)
- $p_i = 1$: prize at each node
- c_{ij}: inversely proportional to $od(o_i, o_j)$
- n: total length of the DNA sequence

The DNA Sequencing Problem

n	e	[10]				DNA-CM				
		Match	Deviation	Optimal	Time†	Match	Deviation	Optimal	Time♯	No. x^{z}
200	0.05	99.9	0.36	39/40	6.5	100	0.00	40/40	0.1	1
	0.20	99.2	3.47	37/40	8.5	100	0.00	40/40	0.1	1
400	0.05	99.2	4.68	38/40	23.9	100	0.00	40/40	19.8	3
	0.20	99.2	3.47	36/40	30.8	100	0.00	40/40	21.7	3
500	0.05	99.8	1.15	39/40	46.5	100	0.00	40/40	31.5	7
	0.20	99.6	1.85	35/40	53.6	99.8	0.15	39/40	44.1	10
600	0.05	98.0	7.71	36/40	80.9	99.3	1.4	38/40	92.2	15
	0.20	98.0	9.19	32/40	91.6	99.0	2.7	37/40	118.3	14

†: CPU Time on a Pentium 4, 2.2GHz and 512MB of RAM.
♯: CPU Time on a Pentium 4, 2.0GHz and 2GB of RAM.

Comparison of computational results on 320 DNA sequences from Blazewicz et al. [10] (Source: GenBank database). Average values computed over 40 instances per class.

Conclusion

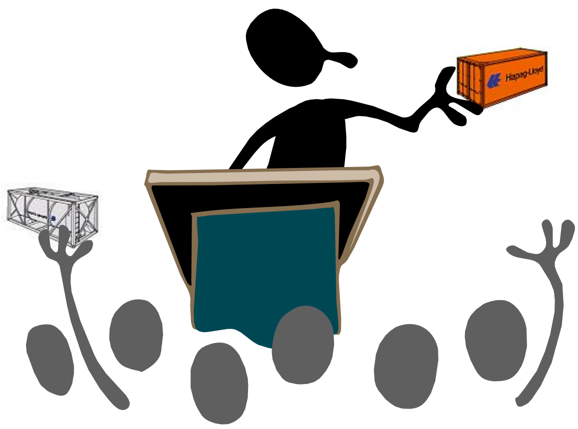
During our work related to Container terminals we found a “nice“ (simplified) problem which may be treated with metaheuristics.

Idea: Try a new metaheuristic.
(Corridor method; Sniedovich and Voß, 2006)

Better results compared to the best found in the literature.
Proof of concept (for the corridor method)?
(Embedding into current simulation studies for developing strategies for the DRMG.)

IWI HAMBURG

Remarks? Questions?



IWI HAMBURG

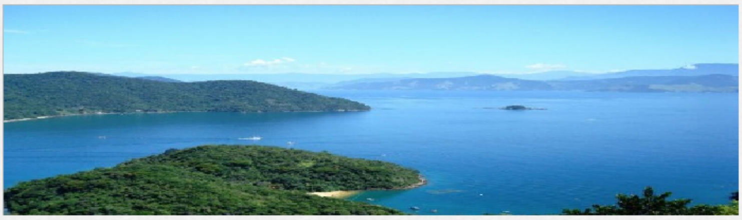
Matheuristics'2012

Fourth International Workshop on Model-Based Metaheuristics

September 16 to 21, 2012 - Angra dos Reis, Rio de Janeiro, Brazil

type your search

Home Venue Organization Submissions Registration Invited lectures Program Contact us



September 16 to 21, 2012 :: Angra dos Reis, Rio de Janeiro, Brazil

The Matheuristics workshop series is proposed as a primary forum for researchers working on exploiting mathematical programming techniques in a (meta)heuristic framework, granting to mathematical programming approaches the problem robustness and time effectiveness which characterize metaheuristics, or exploiting the mathematical programming model formulation in the customization of a metaheuristic for specific or general problems.

The **Matheuristics'2012** workshop will be held at Portogalo Suite Hotel, located in Angra dos Reis, Brazil, from **September 16 to 21, 2012**.

http://www.ic.uff.br/matheuristics2012

IWI HAMBURG

Some new talks (Example): Local Branching (Fischetti, Lodi)

The local branching framework

For a given positive integer parameter k, we define the *k-OPT neighborhood* $\mathcal{N}(\bar{x}, k)$ of $\bar{x}$ as the set of the feasible solutions of (P) satisfying the additional *local branching constraint*:

$$\Delta(x, \bar{x}) := \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \bar{S}} x_j \leq k$$

where the two terms in left-hand side count the number of binary variables flipping their value (with respect to $\bar{x}$) either from 1 to 0 or from 0 to 1, respectively.

IWI HAMBURG

Local Branching

A commonly used heuristic idea in MIP context is the so-called hard *variable fixing* or *diving*:

1. the solution of a continuous relaxation x^* is "analyzed";
2. some of its nonzero variables are heuristically rounded-up to the nearest integer (if non-integer) and then fixed to this value;
3. the method is iterated until either a feasible solution is found or the problem is infeasible.

The obvious question related to this mechanism is however:

How should one choose the actual variables to be fixed?

The idea is simple. In a binary problem in which a current feasible solution $\bar{x}$ is given, impose a soft *variable fixing* constraint, fixing a relevant number of variables without losing the possibility of finding good feasible solutions:

$$\sum_{j=1}^{n} \bar{x}_j \, x_j \geq \lceil 0.9 \sum_{j=1}^{n} \bar{x}_j \rceil$$

IWI HAMBURG