`--enable-debug`	compile all projects with debug op
`--enable-debug-symphony`	compile only SYMPHONY project
`--enable-msvc`	Under MSys2, compile so that exe
`--enable-static`	build static libraries
`--enable-static-executable`	create a complete static executable
`--enable-gnu-packages`	compile with GNU packages compile interactive optimizer with
`--disable-cgl-cuts`	disable generic cut generation
`--enable-sensitivity-analysis`	compile in the sensitivity analysis
`--enable-root-only`	process only the root node
`--enable-frac-branching`	compile in the fractional branching
`--enable-tests`	perform additional sanity checks (
`--enable-tm-tests`	perform more tests
`--enable-trace-path`	additional debugging options
`--enable-cut-check`	additional debugging options
`--enable-statistics`	additional statistics
`--enable-pseudo-costs`	enable some experimental pseudo-
`--enable-draw-graph`	enable IGD graph drawing applica
`--with-XXX-incdir`	specify the directory with the hea where XXX is one of LP solver pa xpress
`--with-XXX-lib`	specify the flags to link with the li where XXX is one of LP solver pa xpress
`--with-lp-solver[=lpsolver]`	specify the LP solver in small lette
`--with-application`	compile the application library
`--enable-openmp`	compile in OpenMP features
`--with-pvm`	compile in parallel architecture (as installed and the variable PVM_R
`--without-cg`	compile without cut generator mo
`--without-cp`	compile without cut pool module
`--without-lp`	compile without LP solver module
`--without-tm`	compile without tree manager mo

```
[frame=lines]
int main(int argc, char **argv)
{
   sym_environment *env = sym_open_environment();
   sym_parse_command_line(env, argc, argv);
   sym_load_problem(env);
   sym_solve(env);
   sym_close_environment(env);
```

```
[frame=lines]
int main(int argc, char **argv)
{
   sym_environment *env = sym_open_environment();
   sym_parse_command_line(env, argc, argv);
   sym_load_problem(env);
   sym_set_int_param(env, "find_first_feasible", TRUE);
   sym_set_int_param(env, "node_selection_strategy", DEPTH_FI
   sym_solve(env);
   sym_set_int_param(env, "find_first_feasible", FALSE);
   sym_set_int_param(env, "node_selection_strategy", BEST_FIR
   sym_warm_solve(env);
```

```
[frame=lines]
int main(int argc, char **argv)
{
   warm_start_desc *ws;
   sym_environment *env = sym_open_environment();
   sym_parse_command_line(env, argc, argv);
   sym_load_problem(env);
   sym_set_int_param(env, "node_limit", 100);
   sym_set_int_param(env, "keep_warm_start", TRUE);
   sym_solve(env);
   ws = sym_get_warm_start(env);
   sym_set_int_param(env, "node_limit", -1);
   sym_warm_solve(env);
   sym_set_obj_coeff(env, 0, 100);
   sym_set_obj_coeff(env, 200, 150);
   sym_set_warm_start(ws);
   sym_warm_solve(env);
```

i

i^{th}

```
[frame=lines]
int main(int argc, char **argv)
{
   sym_environment *env = sym_open_environment();
   sym_parse_command_line(env, argc, argv);
   sym_load_problem(env);
   sym_set_obj2_coeff(env, 0, 1);
   sym_mc_solve(env);
```

```
[frame=lines]
int main(int argc, char **argv)
{
   OsiSymSolverInterface si;
   si.parseCommandLine(argc, argv);
   si.loadProblem();
   si.branchAndBound();
```

S

$c \in \mathbf{R}^S$

$\hat{s} \in S$

$\hat{s}$

n

$S_1, \ldots, S_n$

$$\cup_{i=1}^{n} S_i = S$$

Bounding Operation

Input: A subproblem $\mathcal{S}$, described in terms of a "small" set of ineq
$\mathcal{S} = \{x^s : s \in \mathcal{F} \text{ and } ax^s \leq \beta \; \forall \, (a, \beta) \in \mathcal{L}'\}$ and α, an upper bound
value.

Output: Either (1) an optimal solution $s^* \in \mathcal{S}$ to the subproblem, (2)
optimal value of the subproblem, or (3) a message `pruned` indicating
should not be considered further.

Step 1. Set $\mathcal{C} \leftarrow \mathcal{L}'$.

Step 2. Solve the LP $\min\{cx : ax \leq \beta \; \forall \, (a, \beta) \in \mathcal{C}\}$.

Step 3. If the LP has a feasible solution $\hat{x}$, then go to Step 4.
output `pruned`. This subproblem has no feasible solutions.

Step 4. If $c\hat{x} < \alpha$, then go to Step 5. Otherwise, STOP and
subproblem cannot produce a solution of value better than α.

Step 5. If $\hat{x}$ is the incidence vector of some $\hat{s} \in \mathcal{S}$, then $\hat{s}$ is the
this subproblem. STOP and output $\hat{s}$ as s^*. Otherwise, apply separ
heuristics to $\hat{x}$ to get a set of violated inequalities $\mathcal{C}'$. If $\mathcal{C}' = \emptyset$, then
on the value of an optimal element of $\mathcal{S}$. STOP and return $\hat{x}$ and
Otherwise, set $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$ and go to Step 2.

$$CP = (E, \mathcal{F})$$

E

$$\mathcal{F} \subseteq 2^E$$

$c \in \mathbf{R}^E$

$\mathcal{F}$

P

L

$$\mathcal{P} = \{x \in \mathbf{R}^n : ax \leq \beta \ \ \forall \ (a, \beta) \in \mathcal{L}\}.$$

Branching Operation

Input: A subproblem $\mathcal{S}$ and $\hat{x}$, the LP solution yielding the lower bou

Output: $S_1, \ldots, S_p$ such that $\mathcal{S} = \cup_{i=1}^{p} S_i$.

Step 1. Determine sets $\mathcal{L}_1, \ldots, \mathcal{L}_p$ of inequalities such that $\mathcal{S} =$
$\beta \ \forall \ (a, \beta) \in \mathcal{L}_i\}$ and $\hat{x} \notin \cup_{i=1}^{n} S_i$.

Step 2. Set $S_i = \{x \in \mathcal{S} : ax \leq \beta \ \ \forall \ (a, \beta) \in \mathcal{L}_i \cup \mathcal{L}'\}$ where $\mathcal{L}'$ is t
used to describe $\mathcal{S}$.

Generic Branch and Cut Algorithm

Input: A data array specifying the problem instance.

Output: The global optimal solution s^* to the problem instance.

Step 1. Generate a “good” feasible solution $\hat{s}$ using heuristics. Set

Step 2. Generate the first subproblem $\mathcal{S}^I$ by constructing a small
valid for $\mathcal{P}$. Set $A \leftarrow \{\mathcal{S}^I\}$.

Step 3. If $A = \emptyset$, STOP and output $\hat{s}$ as the global optimum s^*. Ot
$\mathcal{S} \in A$. Set $A \leftarrow A \setminus \{\mathcal{S}\}$. Process $\mathcal{S}$.

Step 4. If the result of Step 3 is a feasible solution $\overline{s}$, then $c\overline{s} < c\hat{s}$. Se
and go to Step 3. If the subproblem was pruned, go to Step 3. Other

Step 5. Perform the branching operation. Add the set of subproblem
go to Step 3.

$$\begin{array}{lrcl} \text{vmin} & [cx, dx], & & \\ \text{s.t.} & Ax & \leq & b, \\ & x & \in & \mathbb{Z}^n. \end{array}$$

p

q

$$cq \leq cp$$

$$dq \leq dp$$

$$0 \leq \alpha \leq 1$$

$(\alpha c + (1 - \alpha) d) x.$

α

$$x^c$$

$$\alpha = 1$$

x^d

$\alpha = 0$

$$\max\{\alpha(cp - cx^c), (1 - \alpha)(dp - dx^d)\}.$$

(i, j)

$i < j$

$\leq$

$\geq$

C++ Interface	**C Interface**	**Description**
OsiSymSolverInterface	sym_open_environment	create a new environment.
loadProblem	sym_load_problem	load the problem read trough an M
branchAndBound	sym_solve/sym_warm_solve	solve the MILP problem from scratc from a warm start if loaded.
resolve	sym_warm_solve	re-solve the MILP problem after so
initialSolve	sym_solve	solve the MILP problem from scratc
multiCriteriaBranchAndBound	sym_mc_solve	solve the multi criteria problem.
setInitialData	sym_set_defaults	set the parameters to their defaults.
parseCommandLine	sym_parse_command_line	read the command line arguments.
findInitialBounds	sym_find_initial_bounds	find the initial bounds via the user
createPermanentCutPools	sym_create_permanent_cut_pools	save the global cuts.
loadProblem	sym_explicit_load_problem	load the problem through a set of a
getWarmStart	sym_get_warm_start	get the warm start description.
setWarmStart	sym_set_warm_start	set the warm start description.
getLbForNewRhs	sym_get_lb_for_new_rhs	find a lower bound to the new rhs p using the post solution info.
getUbForNewRhs	sym_get_lb_for_new_rhs	find an upper bound to the new rhs using the post solution info.
getLbForNewObj	sym_get_lb_for_new_rhs	find a lower bound to the new obj p using the post solution info.
getUbForNewObj	sym_get_lb_for_new_rhs	find an upper bound to the new obj using the post solution info.
reset	sym_close_environment	return the allocated memory.
setIntParam	sym_set_int_param	set the integer type OSI parameter.
setSymParam(int)	sym_set_int_param	set the integer type SYMPHONY p
setDblParam	sym_set_dbl_param	set the double type OSI parameter.
setSymParam(double)	sym_set_dbl_param	set the double type SYMPHONY p
setStrParam	sym_set_str_param	set the string type OSI parameter.
setSymParam(string)	sym_set_str_param	set the string type SYMPHONY pa
getIntParam	sym_get_int_param	get the value of the integer type OS
getSymParam(int &)	sym_get_int_param	get the value of the integer type SY
getDblParam	sym_get_dbl_param	get the value of the double type OS
getSymParam(double &)	sym_get_dbl_param	get the value of the double type SY
getStrParam	sym_get_str_param	get the value of the string type OSI
getSymParam(string &)	sym_get_str_param	get the value of the string type SY
isProvenOptimal	sym_is_proven_optimal	query the problem status.
isProvenPrimalInfeasible	sym_is_proven_primal_infeasible	query the problem status.
isPrimalObjectiveLimitReached	sym_is_target_gap_achieved	query the problem status.
isIterationLimitReached	sym_is_iteration_limit_reached	query the problem status.
isTimeLimitReached	sym_is_time_limit_reached	query the problem status.
isTargetGapReached	sym_is_target_gap_achieved	query the problem status.
getNumCols	sym_get_num_cols	get the number of columns.
getNumRows	sym_get_num_rows	get the number of rows.
getNumElements	sym_get_num_elements	get the number of nonzero elements.
getColLower	sym_get_col_lower	get the column lower bounds.
getColUpper	sym_get_col_upper	get the column upper bounds.
getRowSense	sym_get_row_sense	get the row senses.
getRightHandSide	sym_get_rhs	get the rhs values.
getRowRange	sym_get_row_range	get the row range values.
getRowLower	sym_get_row_lower	get the row lower bounds.
getRowUpper	sym_get_row_upper	get the row upper bounds.
getObjCoefficients	sym_get_obj_coeff	get the objective function vector.

C++ Interface	**C Interface**	**Description**
getObjSense	sym_get_obj_sense	get the objective sense.
isContinuous	sym_is_continuous	query the variable type.
isBinary	sym_is_binary	query the variable type.
isInteger	sym_is_integer	query the variable type.
isIntegerNonBinary	-	query the variable type.
isFreeBinary	sym_is_binary	query the variable type.
getMatrixByRow	-	get the constraint matrix by row orien
getMatrixByCol	-	get the constraint matrix by column o
getInfinity	-	get the infinity definition of SYMPHO
getColSolution	sym_get_col_solution	get the current best column solution.
getRowActivity	sym_get_row_activity	get the current row activity.
getObjValue	sym_get_obj_val	get the current best objective value.
getPrimalBound	sym_get_primal_bound	get the primal upper bound.
getIterationCount	sym_get_iteration_count	get the number of the analyzed tree n
setObjCoeff	sym_set_obj_coeff	set the objective function vector.
setObj2Coeff	sym_set_obj2_coeff	set the second objective function vect
setColLower	sym_set_col_lower	set the column lower bounds.
setColUpper	sym_set_col_upper	set the column upper bounds.
setRowLower	sym_set_row_lower	set the row lower bounds.
setRowUpper	sym_set_row_upper	set the row upper bounds.
setRowType	sym_set_row_type	set the row characteristics.
setObjSense	sym_set_obj_sense	set the objective sense.
setColSolution	sym_set_col_solution	set the current solution.
setContinuous	sym_set_continuous	set the variable type.
setInteger	sym_set_integer	set the variable type.
setColName	sym_set_col_names	set the column names.
addCol	sym_add_col	add columns to the constraint matrix.
addRow	sym_add_row	add rows to the constraint matrix.
deleteCols	sym_delete_cols	delete some columns from the constrai
deleteRows	sym_delete_rows	delete some rows from the constraint
writeMps	-	write the current problem in MPS for
applyRowCut	-	add some row cuts.
applyColCut	-	add some column cuts.
SymWarmStart(warm_start_desc *)	sym_create_copy_warm_start	create a SYMPHONY warm start by
SymWarmStart(char *)	sym_read_warm_start	create a SYMPHONY warm start rea
getCopyOfWarmStartDesc	sym_create_copy_warm_start	get the copy of the warm start struct
writeToFile	sym_write_warm_start_desc	write the loaded warm start to a file.

✕

$\leq$

$\geq$

0th

1st

$\texttt{rhs} + \texttt{range}$

$=$

ϵ

-1

th

2

5

1

z_i^+, z_i^-

i

$$s_i = \alpha \times \min\{z_i^+, z_i^-\} + (1 - \alpha) \times \max\{z_i^+, z_i^-\}$$

α

$[0, 1]$

$n^{\mathrm{t}h}$

n

C++ Interface	**C Interface**	**Valu**
OsiSymVerbosity	verbosity	-user
OsiSymWarmStart	warm_start	-boole
OsiSymNodeLimit OsiMaxNumIteration OsiMaxNumIterationHotStart	node_limit	-user
OsiSymFindFirstFeasible	find_first_feasible	-boole
OsiSymSearchStrategy	node_selection_rule	LOW HIGH BREA DEPT
OsiSymUsePermanentCutPools	use_permanent_cut_pools	-boole
OsiSymGenerateCglGomoryCuts	generate_cgl_gomory_cuts	-boole
OsiSymGenerateCglKnapsackCuts	generate_cgl_knapsack_cuts	-boole
OsiSymGenerateCglOddHoleCuts	generate_cgl_oddhole_cuts	-boole
OsiSymGenerateCglProbingCuts	generate_cgl_probing_cuts	-boole
OsiSymGenerateCglCliqueCuts	generate_cgl_clique_cuts	-boole
OsiSymGenerateCglFlowAndCoverCuts	generate_cgl_flow_and_cover_cuts	-boole
OsiSymGenerateCglRoundingCuts	generate_cgl_rounding_cuts	-boole
OsiSymGenerateCglLiftAndProjectCuts	generate_cgl_lift_and_project_cuts	-boole
OsiSymKeepWarmStart	keep_warm_start	-boole
OsiSymTrimWarmTree	trim_warm_tree * -boolean-	
OsiSymDoReducedCostFixing	do_reduced_cost_fixing	-boole
OsiSymMCFindSupportedSolutions	mc_find_supported_solutions	-boole
OsiSymSensitivityAnalysis	sensitivity_analysis	-boole
OsiSymRandomSeed	random_seed	-user
OsiSymDivingStrategy	diving_strategy	BEST COM COM
OsiSymDivingK	diving_k	-user
OsiSymDivingThreshold	diving_threshold	-user
OsiSymGranularity	granularity	-user
OsiSymTimeLimit	time_limit	-user
OsiSymGapLimit	gap_limit	-user
OsiObjOffset	-	-user
OsiProbName	problem_name	-user