



Design and development of a tactical harvest blocking/scheduling tool

Remsoft Inc.
332 Brunswick
Street, Fredericton,
NB, Canada
E3B 1H1
1-800-792-9468 or
1-506-450-1511
www.remsoft.com

Executive Summary

Forest management planning is deciding where and when to harvest timber or perform silvicultural treatments in order to achieve a desired flow of products and benefits from the forest. Strategic forest planning typically addresses the questions of what to do and when to do it, but the question of where to do it is deferred, more appropriately addressed at the tactical level. Ultimately, forest operations must take place on the forest landscape, and because of economic requirements, they generally must occur within relatively large, contiguous blocks instead of dispersed patches.

Despite the importance and magnitude of the problem, harvest blocking and scheduling is still carried out using a largely manual process. Geographic information systems are designed to assist in spatial analysis, but as general purpose tools they do not typically address specific tasks such as blocking. Remsoft (1994) conducted a study that compared available spatial forest planning tools to manual blocking and scheduling efforts. Although the blocking and scheduling tools generated acceptable solutions, the report noted a number of deficiencies associated with them. Continued research into spatial planning led us to hypothesize that the separate blocking and scheduling processes could lead to less-desirable solutions than would be the case if the processes were combined into a single blocking/scheduling tool.

Since 1994, Remsoft has continued to investigate improved methods of spatial planning based on the Jamnick-Walters approach. Because of severe limitations to the Block model, we developed a replacement for it called Stanley. Whereas Block would favor timing choice deviations to address adjacency conflicts, Stanley partitions the scheduling problem into two sub-problems. The first partition deals with harvest units that have adjacent harvest units and the second deals with those that have no adjacent harvest units. Two new heuristics were implemented in Stanley to take advantage of the partitioned scheduling problem that significantly improved its performance over Block (Cogswell 1995).

Remsoft staff have conferred with numerous potential users of Crystal, Block and Stanley to determine how best to improve the overall blocking/scheduling process. Many of these suggestions were incorporated into the new prototype, Stanley Version 2. Most importantly, the Stanley prototype completely subsumes the functionality of Crystal and the previous version of Stanley with vastly improved allocation and scheduling algorithms. It recognizes multiple harvest actions for simultaneous allocation and scheduling, it is cognizant of stands that have been already been assigned harvest periods, it implements and supports multiple harvest actions for simultaneous allocation and scheduling, it is not restricted to

Final Report

submitted to
Canadian Forest
Service, Pacific
Region
March 1996

even-flow assumptions and the user can select one or many different controlling variables, including volume, area treated and monetary returns (present net value).

The Stanley Version 2 prototype is based entirely on new programming code, completely subsuming the functionality of the existing Crystal and Stanley Version 1 programs. Beyond the functionality of the existing programs, additional capabilities have been added that provide significant improvements to solution values. Stanley Version 2 has five phases of operation during each iteration. In phase 1, the Stanley performs essentially the same functions that were part of the Crystal program: stands are randomly selected from the eligible stand list and harvest units are developed by an accretion algorithm. Stanley offers substantially faster operation than Crystal did, and the number of planning periods that can be blocked has been increased from 8 in Crystal to 25. In phase 2, Stanley begins to schedule the harvest units created in phase 1 by partitioning the allocation problem into harvest units with and without neighbors. In phase 3, Stanley addresses output levels from the scheduled harvest units and attempts to balance flows to closely match those from the strategic schedule, targeting those periods which deviate most from the strategic harvest schedule. Phase 4 attempts to consolidate any discontinuous blocks by incorporating new stands so that the fragments become part of one contiguous block. Finally, phase 5 selectively deallocates blocks to reduce harvest levels in the highest periods, thus reducing periodic flow fluctuations.

Because the capabilities of Stanley now include those previously performed by Crystal, the program requires more input data than the previous version. The activity list details the harvest actions that are to be blocked. Since Woodstock generates this file automatically in DBF format, it was a logical choice to use this file directly. An equivalent file can be generated by users who do not use Woodstock. Stanley must be cognizant of harvest operability limits that are specific to different stand types. Since the operability guidelines are inherent to the strategic harvest schedule, it makes sense to use the strategic harvest scheduling model solution as the source of this information. The stand (polygon) description list is usually derived from a GIS attribute file and contains a unique identification label for each stand, the landscape thematic attributes that identify the development type each stand belongs to, the current age in periods, and the area of the stand. An empty field for storing the harvest period that Stanley assigns to each stand is also required. The stand (polygon) description list contains double entry list of adjacent or proximate stands identified by their unique identification labels. The stand (polygon) extents list is optional and contains the unique stand identification label and northeast and southwest x-y coordinates defining the bounding box or extent for each stand.

To illustrate the capabilities of the Stanley prototype, several case studies are presented with problem parameters consistent with planning regulations in use in British Columbia and elsewhere. The appendix contains data structures for Stanley input files.

In addition to Remsoft's internal testing and development, the Stanley prototype has been undergoing operational feasibility testing at Fraser Inc. and will be installed at the Fundy Model Forest in the near future. Remsoft staff presented two papers dealing with spatial planning issues directly related to the Stanley project and additional publications based on this work are anticipated in the near future.

Introduction

Rationale

Forest management planning is deciding where and when to harvest timber or perform silvicultural treatments in order to achieve a desired flow of products and benefits from the forest. Strategic forest planning typically addresses the questions of what to do and when to do it, but the question of where to do it is deferred, more appropriately addressed at the tactical level. Ultimately, forest operations must take place on the forest landscape, and because of economic requirements, they generally must occur within relatively large, contiguous blocks instead of dispersed patches.

To ensure that the objectives of the strategic planning model are maintained during implementation, harvest blocks established on the forest must be composed of the stand types scheduled for treatment in the strategic model – if they do not, the assumptions of the strategic model may not hold. Future harvest levels may be jeopardized by harvesting stands before they accumulate the harvest volumes expected or by not harvesting decadent stands in a timely fashion, thereby losing volume to mortality. Furthermore, long-term harvesting and silviculture costs may have to increase to compensate for poor decisions made previously.

The importance of well-designed harvest blocks cannot be overstated: every forest company in Canada must detail their planned operations and have them approved. Because it can take weeks or even months to develop a single block layout, the cost of developing management and operating plans is significant. Furthermore, new restrictions and constraints on forest operations for environmental reasons only complicate the problem of harvest blocking, making it that much more difficult to arrive at a solution which is satisfactory to the company, the provincial regulatory bodies and the public.

Despite the importance and magnitude of the problem, harvest blocking and scheduling is still carried out using a largely manual process. Geographic information systems are designed to assist in spatial analysis, but as general purpose tools they do not typically address specific tasks such as blocking. Even value-added GIS products such as ArcForest (ESRI Canada) do not provide specific forest planning tools like harvest scheduling models or blocking and scheduling programs which are the basis of forest management planning. Within the literature, there are numerous examples of work

studying the problem of scheduling harvest blocks under spatial constraints but there is little work on the problem of block generation itself.

Literature Review

The United States forestry community has a long history of forest planning and sophisticated forest planning models based on linear programming. Most of the research effort regarding the scheduling of harvest blocks under adjacency constraints has focused on embedding adjacency constraints within a mixed-integer programming framework (Meneghin et al. 1988; Torres-Rojo and Brodie, 1990; Jones et al. 1991; Yoshimoto and Brodie, 1994). There are several drawbacks to this approach. Integer programming models are notoriously difficult to solve and practical limits on the size of solvable problems prevent application on real-world data sets. Furthermore, the adjacency constraints can only be formulated if each harvest unit is distinct – you cannot combine two smaller blocks to form a single larger block that still meets the maximum opening size requirements. Thus, harvest blocks must either be predefined, or they are limited to single stands, which may not be feasible harvest units.

Baskent (1990) developed an inventory projection model which aggregates stands into feasible harvest blocks as it schedules a long-term management schedule. As an inventory projection model, the scheduler is driven by a set of rules which are applied in a sequential manner over time. As such, the model is incapable of anticipating the impacts of earlier actions and because it cannot undo previous actions, it can easily lead to very sub-optimal solutions. Furthermore, the allocation process becomes progressively more difficult with each planning period as the number of available options necessarily decreases with time.

Simulated annealing has several potential advantages over integer programming, including the ability to model a large number of stands and no requirement for linearity. Given sufficient computational effort, simulated annealing models will theoretically converge to an optimal solution. Although the time to convergence may be very long, very good feasible solutions can often be found quickly. Lockwood and Moore (1993) gave an example application of simulated annealing for a large number of stands where each stand could only be considered for one treatment over the planning horizon. Regenerated stands were not considered in this model and harvest blocks and harvest units consisted of single stands.

Again, there is no ability to group small stands into larger units.

Nelson *et al.* (1991) used a hierarchical planning approach to develop spatially feasible harvest schedules. First, a stratum-based LP approach is used to solve the long-term scheduling problem subject to forest-wide constraints but without specific spatial detail. Then, in a separate step, the cut is allocated to blocks for the first few periods of the long-term planning horizon and scheduled using a Monte Carlo integer programming (MCIP) model. Beyond the problem of manually delineating blocks, the difficulty with this approach is that the only explicit linkage between the strategic and tactical planning models is the allowable harvest and thus the tactical solution may be incompatible with longer term goals.

Dallain (1989) devised a block scheduling tool (*Block*) based on Monte Carlo integer programming (MCIP). *Block* is capable of scheduling harvest under opening size and adjacency constraints and it maximizes the average even-flow harvest volume for up to six planning periods. Walters (1991) developed a harvest blocking tool (*Crystal*) that disaggregates stratum-based harvest schedules and allocates the harvest prescriptions to stands, forming feasible harvest blocks.

Jamnick and Walters (1993) also used a hierarchical planning approach but with two important differences. First, the blocks were generated using an automated blocking algorithm (Walters 1991) instead of manual delineation. Second, only those development types scheduled for harvest in the strategic planning model were eligible for blocking. Thus, a much stronger link between the strategic and tactical planning models was established than was used by Nelson *et al.* (1991).

Walters and Feunekes (1994) used the Jamnick-Walters approach to develop spatially feasible block harvest schedules for two Crown Licenses in New Brunswick. Compared with results found by the Licensees using a manual blocking and scheduling approach, the automated blocking and scheduling tools gave equally good or better solutions, along with significant savings in time and cost.

Overall, there has been very little research effort applied to the problem of developing harvest blocks, though there has been a significant effort in addressing the scheduling issue.

Remsoft's continuing research and development in spatial planning tools

Remsoft (1994) conducted a study that compared available spatial forest planning tools to manual blocking and scheduling efforts. Although the *Crystal* and *Block* tools generated acceptable solutions, the report noted a number of deficiencies associated with them:

Crystal

Crystal was shown to work well despite a few shortcomings. The inability to block out more than a single harvest prescription is a potential problem although it was not an overwhelming limitation in this study. On the other hand, compared to the manual approaches used by the Licensees, Crystal is a vast improvement. It does not require a particular harvest scheduling model; we used Woodstock in this study, but FORMAN+1 or any other stratum-based planning model could have been used to determine the harvest schedule. Many alternative block configurations can be generated by Crystal in a single day, and since it is possible to accommodate previously determined blocks, there is ample opportunity user intervention into the blocking process. Most important, unlike the manual process which is subject to human error, Crystal provides a consistent, reproducible method of generating harvest blocks

The allocation algorithm used in Crystal was sensitive to the blocking parameters specified (particularly minimum block size) and to the nature of the land base on which it was working. Although we used the same sets of parameters on both Licenses, for a given set of parameters Crystal always allocated a higher proportion of scheduled area for License 4 than it did for License 8.

Block

Block performed adequately but required a significant amount of effort and custom programming to be able to use it efficiently. Despite its awkward input file structure and the inability to easily accommodate non-clearcut harvest prescriptions, Block produced good results in this study.

*The assumption that smaller blocks would yield fewer conflicts for a given land base than a larger one (Clements *et al.* 1991) was borne out in this study: when the minimum block size increased from 10 to 20 ha, the number of unharvested blocks more than doubled.*

One of our assumptions about Block did not materialize, however. We assumed that Block would group together blocks eligible for harvest in the same period, thus avoiding adjacency conflicts. However, it appears that the Block algorithm tends to deviate timing of harvest to avoid adjacency conflicts with only incidental grouping of compatible blocks.

(Remsoft 1994)

Since 1994, Remsoft has continued to investigate improved methods of spatial planning based on the Jamnick-Walters approach. One of the biggest difficulties with the *Block* model is the inefficiency of the pure MCIP algorithm. Although it is easy to randomly generate new sequences for harvesting, the probability is high that one element within the schedule is infeasible (for example, a single block harvested during a green-up restriction period). In such cases, it might be easy to correct the schedule, but *Block's* MCIP algorithm discards the entire schedule and generates a new one. The end result is that a very large number of attempts is required to generate a spatial harvest schedule under even modest flow constraints. Using a hybrid MCIP - heuristic approach, we have devised a new algorithm which substantially reduces the number of attempts required to generate feasible solutions.

The other major limitation of *Block* that Remsoft (1994) found was its tendency to favor changes in harvest timing over block aggregation. In many cases, it would be easy to combine two adjacent harvest units into a single block and still satisfy opening size and green-up delay constraints. However, the *Block* algorithm consistently scheduled two adjacent harvest units in separate planning periods, resulting in potentially large reductions in harvest volumes. To address this shortcoming, we partitioned the scheduling problem into two sub-problems. The first deals with harvest units that have adjacent harvest units and the second deals with those that have no adjacent harvest units. Two new heuristics were implemented in *Stanley* to take advantage of the partitioned scheduling problem.

The first heuristic consisted of two phases: a *make-big* phase and a *balance-flow* phase. In the *make-big* phase, a harvest unit is selected randomly and adjacent harvest units area checked to determine if they are eligible for harvest. If no adjacent harvest units have been scheduled for harvest, the harvest unit is tentatively assigned the preferred harvest period that *Crystal* used when it created the harvest unit. Alternatively, if adjacent harvest units have been

already scheduled, the harvest unit is assigned to a valid planning period corresponding to one or more of the neighboring harvest units. Adjacent harvest units scheduled for harvest in the same planning period are considered a single harvest block and must meet opening size requirements and during the *make-big* phase checks are made to ensure compliance. Refer to Figure 1a.

In the *balance-flow* phase, the timing choices assigned to harvest units are modified so that overall harvest flows are within the specified tolerance (e.g., typical flow constraints might be $\pm 5\%$). Timing choices are adjusted only if the change does not cause opening size or adjacency violations. Since the *balance-flow* phase can work at odds with the *make-big* phase, the *make-big* phase is performed first and the *balance-flow* phase is initiated after a random proportion of the harvest units are assigned tentative harvest periods.

Because harvest units without neighbors are isolated, there is no need to be concerned about adjacency or opening size constraints. Therefore, the purpose of the second heuristic is simply to balance harvest flows by scheduling harvest units in periods where the largest shortfalls in harvest volume exist. Refer to Figure 1b.

Crystal uses stand topology to build harvest blocks by accretion. The process used is relatively simple:

1. an eligible stand is randomly selected
2. the potential harvest period(s) for the stand is(are) checked
3. a search is made for adjacent stands that are also eligible for harvesting in the same period(s)
4. one stand is selected for inclusion in the new block
5. a new search is made for stands adjacent to the selected stands
6. process is repeated until the desired block size is reached or it is determined that no more stands are available.

Graphically, the process is depicted in Figure 2.

When *Crystal* is finished, it writes the preferred harvest period and harvest unit number assigned to each stand into a DBF file. It uses the stand adjacency information to generate an adjacency list for harvest units, also in DBF format. *Stanley* uses both these files, plus harvest unit yield information to schedule harvest units and form final harvest blocks.

Cogswell (1995) used *Woodstock*, *Crystal*, and *Stanley* in an iterative re-planning process to test the spatial feasibility of timber harvest schedules.

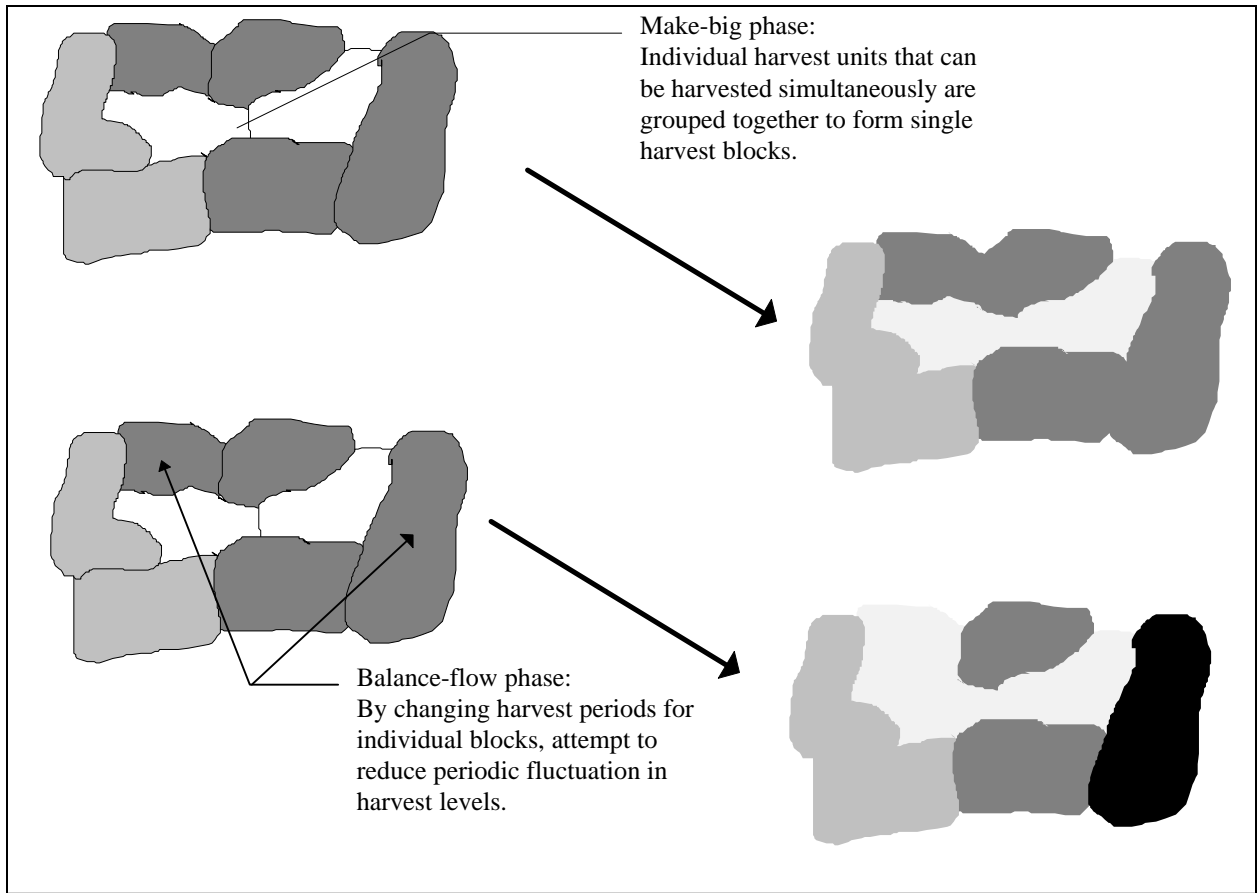


Figure 1a. Partitioned allocation problem, Stanley Version 1: stands with eligible neighbors.

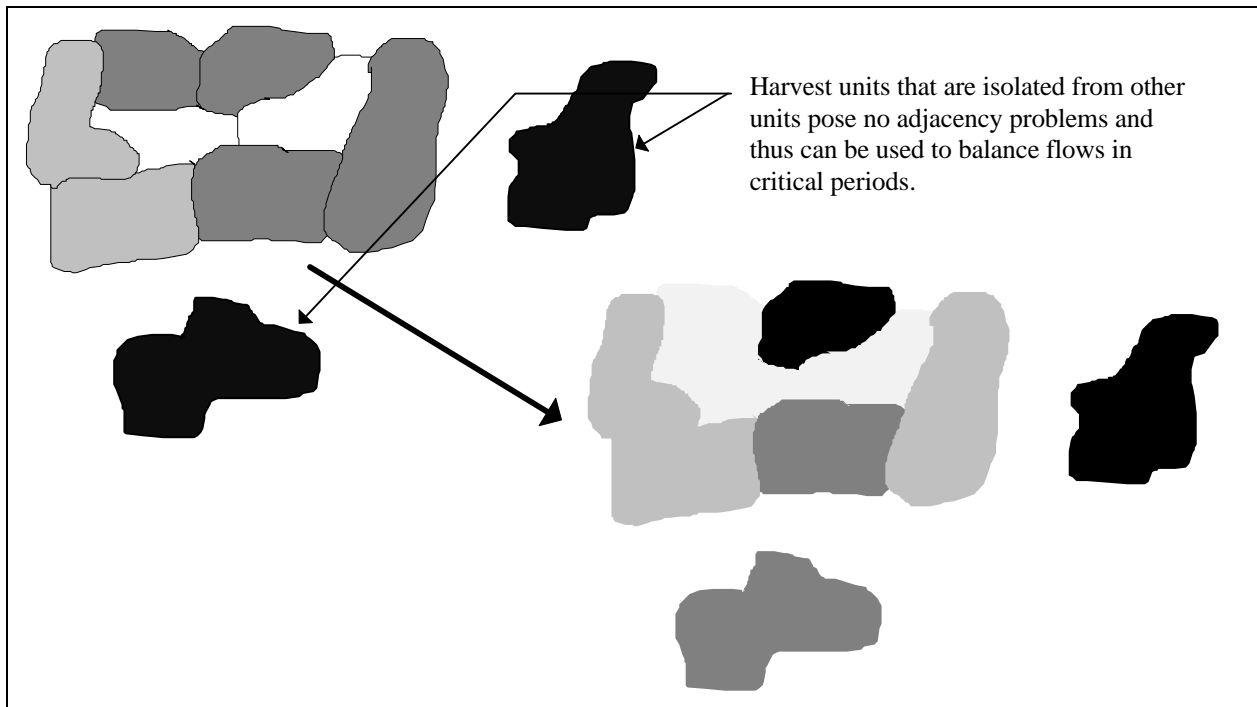


Figure 1b. Partitioned allocation problem, Stanley Version 1: stands with no eligible neighbors.

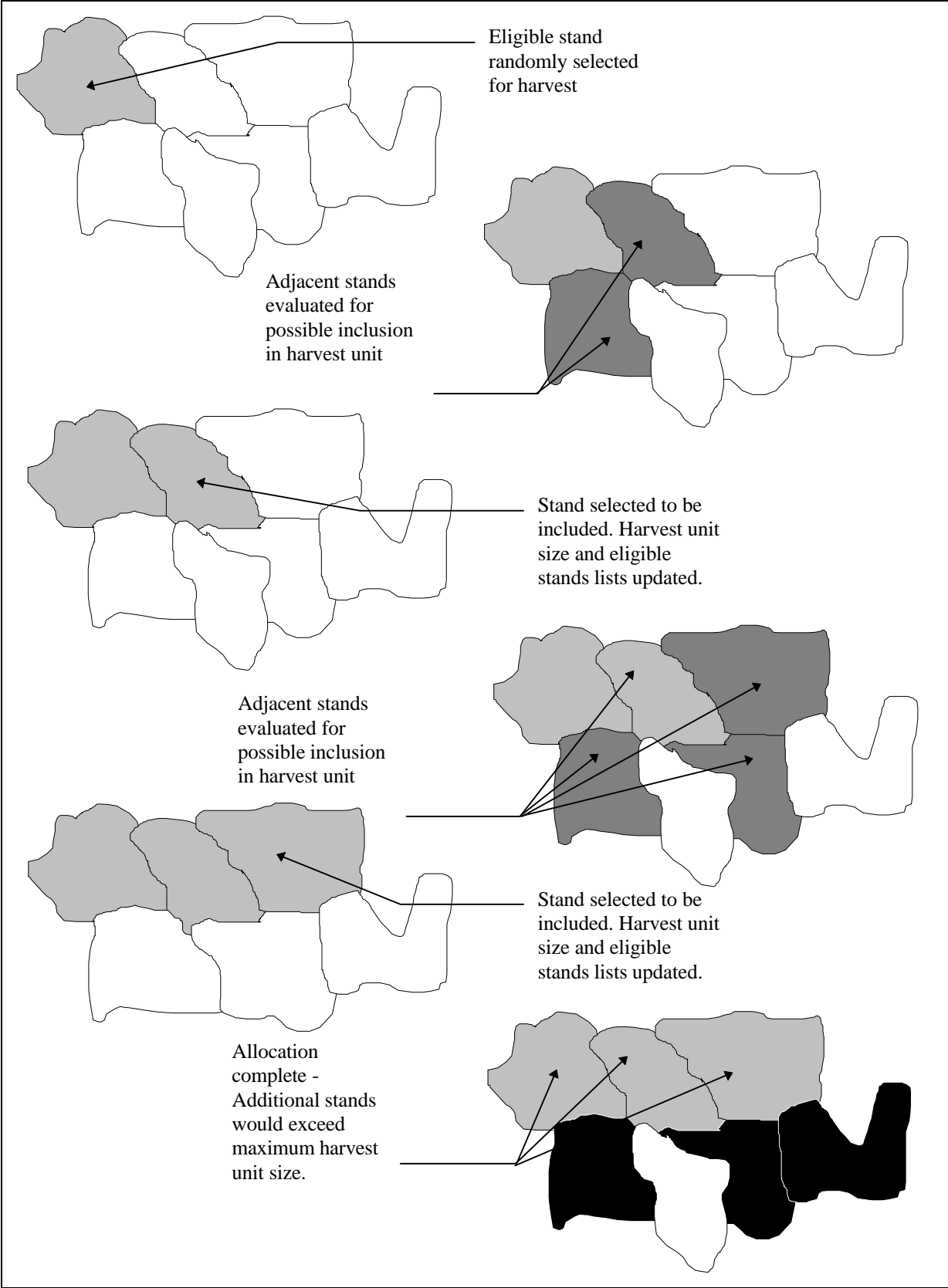


Figure 2. Graphical depiction of the allocation algorithm used in Crystal and Stanley Version 1.

Compared to *Block*, *Stanley* was far more efficient: solution times for a given set of feasible solutions were much shorter and the solutions themselves were significantly better than those produced by *Block*.

Continued testing of *Crystal* and *Stanley* led us to hypothesize that the separate blocking and scheduling processes could lead to less-desirable solutions than would be the case if the processes were combined. For example, it is conceivable that a *Crystal* block configuration might allocate a large proportion of the area scheduled for harvest in a strategic wood supply model, yet that same configuration may present a great deal of difficulty when opening size and adjacency constraints are applied by *Stanley*. There is nothing in the literature that suggests how to evaluate a block configuration for potential spatial conflicts other than to test it under a complete set of spatial constraints. Thus, the required procedure is to first generate numerous block configurations with *Crystal* and test many scheduling alternatives for each configuration using *Stanley*. However, conducting this type of analysis using *Crystal* and *Stanley* is somewhat cumbersome.

In recent years, forest companies have come to include multiple harvest actions in their strategic harvest schedules, and to consider harvest flow policies other than even-flow. *Crystal* and *Stanley* were both designed for clearcut harvests. Although *Crystal* is insensitive to harvest volume fluctuations in the strategic harvest schedule, *Stanley* assumes that even-flow is the desired flow policy. Remsoft (1994) suggests several workarounds for multiple harvest actions, but ultimately these tools are too limited for scheduling multiple harvest activities.

Features of *Stanley* Version 2

Remsoft staff have conferred with numerous potential users of *Crystal*, *Block* and *Stanley* to determine how best to improve the overall blocking/scheduling process. Many of these suggestions were incorporated into the new prototype.

Recognize manual blocking efforts

Forest companies do not undertake forest planning from a clean slate – operations must continue and activities that were planned during the prior planning exercise will be carried out regardless of any new analyses. Therefore, *Stanley* is cognizant of stands that have been already been assigned harvest periods. For harvest periods prior to the first *Stanley* planning period, sequential integers less than 1 are used to indicate the appropriate period (i.e., a block assigned

harvest period 0 was harvested in the period *immediately before* the first planning period and a block assigned to period -3 was harvested *four* periods before the first *Stanley* planning period). This numbering convention is necessary to recognize the green-up delays that may still be in force from previous harvest activities. *Stanley* avoids violating the green-up restrictions by blocking around the existing harvest blocks and assigning appropriate harvest periods.

If the user specifies harvest periods for stands that are within the *Stanley* planning horizon, he or she can instruct *Stanley* to work with these pre-allocated blocks in one of two ways. First, the configuration and harvest periods for these blocks may be fixed and *Stanley* will simply allocate stands to fit these blocks as best it can. The alternative is to allow *Stanley* to increase the size of a block by incorporating additional stands but leave the timing choices unchanged. We did not offer the possibility of *Stanley* changing both the timing and configuration of pre-allocated blocks because we felt such functionality had limited value.

Allow multiple harvest actions

The current versions of *Crystal* and *Stanley* Version 1 assume that only clearcut harvests will be performed. *Stanley* Version 2 supports multiple harvest actions for simultaneous allocation and scheduling.

In order to implement this capability, we had to address basic issues of planning. For example, a spruce stand may be eligible for clearcut harvesting or a uniform shelterwood harvest but which one takes precedence during the allocation process? *Stanley* attempts to distribute the harvest prescriptions over the allocation process evenly. If a stand is selected for blocking that is eligible for more than one harvest prescription, *Stanley* evaluates its progress in meeting the area targets for each prescription in that stand type. Thus, if relatively less progress has been made toward allocating prescription A than prescription B, the algorithm will allocate the stand to prescription A.

A more complicated issue is one that arises from multiple-entry harvests such as shelterwood harvests or two-pass cuts. How do you recognize the volume outcomes from the regeneration cut and overstory removals that occur in different periods, and how do you force the second removal to occur? In *Woodstock* models, multiple entry harvests are often represented as discrete actions, but in practice they are linked and represent a single blocking choice. As yet, we have

not devised a method for dealing with harvest volumes arising from intermediate and final harvests in multiple entry harvest system. However, the objective function used in *Stanley* only considers the outputs arising from the initial harvest of a multiple entry system, and by blocking only the first entry the location of such activities is fixed. If the blocked solution generated by *Stanley* is later run through *Woodstock*, the transitions associated with the first entry harvests will occur, and the volumes arising from later entries will be properly accounted for. The drawback to this approach is that it will not work if the middle or final harvest entries are subject to opening size restrictions but the initial harvest entry is not.

Allow for objective functions that are not only volume maximization

Stanley Version 1 assumes that the output objective is to maximize the average harvest level under even flow constraints. The new algorithm is not restricted to even-flow assumptions and the user can select one or many different controlling variables, including volume, area treated and monetary returns (present net value).

When even-flow is no longer required, evaluating deviations from the strategic schedule becomes more difficult. For example, exceeding the harvest area targets in one period by 200 ha is a small problem when 10,000 ha are scheduled to be harvested, but the same amount is a huge deviation in another period where 500 ha are scheduled to be harvested.

Stanley evaluates the percentage of target attained in each period and verifies that the deviation in targets attained period to period is less than a given percentage. For example, if the user specifies a tolerance of $\pm 5\%$ and *Stanley* attains targets of 87% and 94% for the first two periods, the solution will not be considered acceptable since the maximum variation is more than 5%. We are continuing to investigate better methods of evaluating solutions that are intuitive and robust under a variety of different flow policies.

In the current versions of *Crystal* and *Stanley*, it is assumed that the same objectives are used in the strategic and tactical plans. However, if maximization of present net value is the objective, significant timing choice deviations from the strategic schedule may have little impact on PNV even if the product mix is substantially different. So, if product mix is more important, it may be better to use a volume objective in the blocking/scheduling step even though

the strategic schedule maximized present net value. *Stanley* cannot make these kinds of decisions about objectives (nor should it) but analysts must be aware of the subtleties associated with strategic and tactical planning to use the program effectively. *Stanley* provides different control structures that affect blocking and scheduling performance but how these structures work may not be intuitively obvious.

Simplify the interface between the strategic harvest schedule, Crystal and Stanley

Although *Crystal*, *Block* and *Stanley* Version 1 are not difficult to use, they do not work together seamlessly. *Block* input files use a format that was easy to implement as a graduate thesis project, but for general usage it is difficult to use. *Crystal* generates the block configuration and adjacency relationships needed by *Stanley* but the development of block yields for scheduling remains problematic. The linkages between strategic and tactical models and between the final blocks and the GIS map files are also somewhat cumbersome, depending on specific field names and formatting to make it work. All of these issues have been addressed in the new algorithm. Wherever possible, input data and solution files are in DBF format so that the solutions generated by *Stanley* are easily imported into GIS for mapped displays. In a separate R&D initiative, Remsoft is cooperating with ESRI Canada to develop smoother interfaces between *Woodstock*, *Stanley* and ESRI's ArcForest and ArcView products. If successful, the development of this linkage has the potential to greatly improve overall data management and will make the integration of spatial data into the planning process a much easier and less intimidating task.

A philosophical question that we had to wrestle with was, *does it make sense to have completely generic planning tools that work with any strategic planning model?* *Crystal*, *Block* and *Stanley* were all designed as stand-alone programs but the consensus from users is that they are too difficult to use as a system. Since we do not have control over the development of other planning models, we cannot anticipate changes to syntax nor can we take advantage of special structures they may offer. From our standpoint, it seemed clear that we should emphasize functionality over portability and take advantage of structures available to us in *Woodstock*. In particular, this alleviated difficulties associated with operability of stands when timing choice deviations were required as well as the issue of generating block yields. However, *Stanley*

does not specifically require the use of *Woodstock*, although using it can simplify matters greatly.

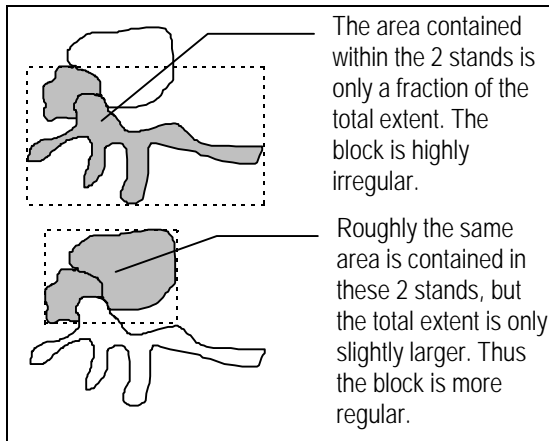


Figure 3. Using extents to control harvest unit configurations (*Stanley*, Phase 1).

Improve the allocation process

Although *Crystal* appears to generate feasible harvest units most of the time, we have observed systematic behaviors that result in poor block configurations. This behavior is related to the topological map information that is the basis for adjacency relationships. *Stanley* uses more sophisticated blocking rules to ensure that blocks are reasonably compact and not arbitrarily oriented along a particular axis. If the user provides stand extent values along with adjacency information, *Stanley* will use this information to select stands for allocation that result in less irregularly shaped blocks. Refer to Figure 3.

Stanley Version 2

The *Stanley* Version 2 prototype is based entirely on new programming code, completely subsuming the functionality of the existing *Crystal* and *Stanley* Version 1 programs. Beyond the functionality of the existing programs, additional capabilities have been added that provide significant improvements to solution values.

Algorithm

Stanley Version 2 has five phases of operation during each iteration. In phase 1, *Stanley* performs essentially the same functions that were part of the *Crystal* program: stands are randomly selected from the eligible stand list and harvest units are developed by an accretion algorithm. *Stanley* offers substantially faster operation than *Crystal* did, and the number of planning periods that can be blocked has been increased from 8 in *Crystal* to 25. *Stanley* uses a more

sophisticated blocking algorithm that can take advantage of adjacency, proximity and extents information about stands – *Crystal* assumed that all stands in the adjacency list were physically contiguous. The data files used by *Crystal* were linked by a numerical code to associate yields and adjacency tables to the harvest prescriptions to be blocked. In *Stanley*, this linkage has been entirely replaced by a landscape thematic attribute scheme modeled after *Woodstock*. Not only is this more intuitive, but it reduces the amount of data processing required to use the program.

In phase 2, *Stanley* begins to schedule the harvest units created in phase 1. *Stanley* Version 1 partitioned the allocation problem into harvest units with and without neighbors; the new prototype does this as well. Initially, harvest units with neighbors are randomly selected and assigned to the preferred harvest period for the development types within the unit. However, as the allocation process continues, harvest units that are proximate to previously scheduled blocks must be considered and if they are assigned different planning periods, a green-up delay may be required.

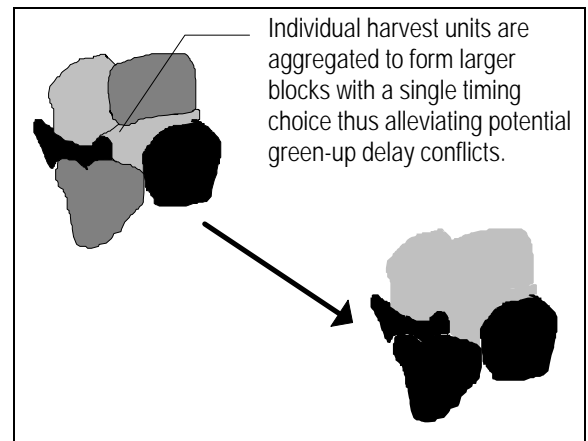


Figure 4. Harvest unit aggregation to avoid adjacency constraint violations (*Stanley* Phase 2).

One way to mitigate this green-up delay is by combining the two harvest units into one and assigning them both to the same harvest period as illustrated in Figure 4. However, if the combined area of the harvest units exceeds the maximum opening size, they cannot be combined. The phase 2 algorithm applies a heuristic to determine if the harvest units should be combined.

In phase 3, *Stanley* addresses output levels from the scheduled harvest units and attempts to balance flows

to closely match those from the strategic schedule. In an ideal situation, the areas allocated to each harvest prescription by *Stanley* would correspond exactly to the timing choices and development types found in the strategic harvest schedule. However, *Stanley* allocates whole stands to prescriptions and the only way to ensure that the strategic area targets are met exactly is to completely allocate a development type class to a single prescription. Since harvest scheduling models tend to split classes over several prescriptions, some deviation from the strategic harvest schedule is expected at the tactical level.

In phase 3, *Stanley* uses two methods for balancing harvest flows, targeting those periods which deviate most from the strategic harvest schedule. The first method involves changing the harvest timing choice of harvest units. Those that contribute to a surplus condition are shifted to a planning period corresponding where there is a deficit. Refer to Figure 5. *Stanley* uses the operability definitions for the activity being applied in the harvest unit *and* the spatial relationships to neighboring harvest units to determine if the new planning period is a feasible choice.

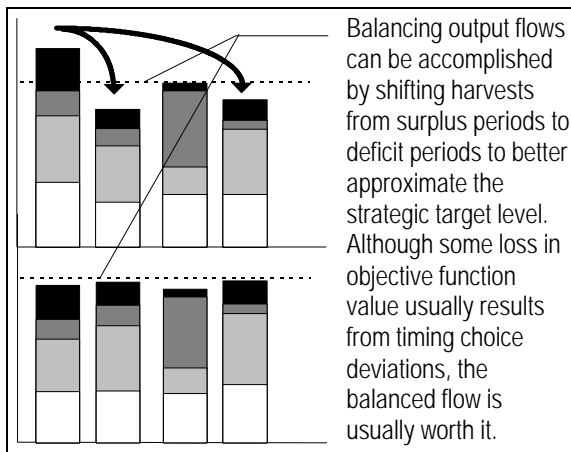


Figure 5. Timing choice shifts used to balance harvest flows (*Stanley* Phase 3).

If a simple timing choice change is not feasible, *Stanley* may then disaggregate the harvest unit back into component stands and then reallocate the individual stands to the residual harvest units. While an entire harvest unit must be left unharvested because it violates opening size or adjacency constraints, it may be possible to reassign its component stands to neighboring units leaving only one or two stands unharvested to satisfy the spatial constraints, as illustrated in Figure 6.

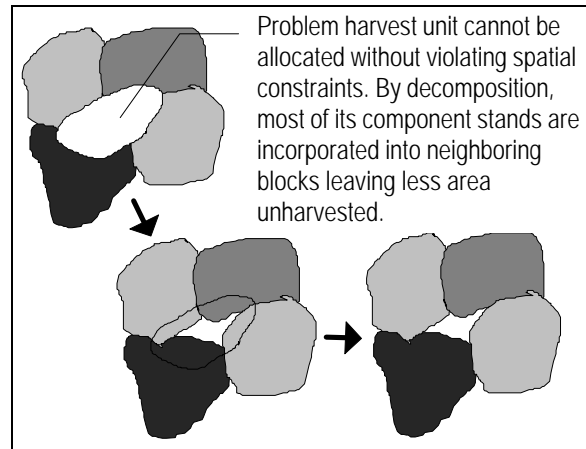


Figure 6. Harvest unit deallocation used to balance harvest flows (*Stanley* Phase 3).

Phase 4 was developed to address the situation where a harvest block is composed of non-contiguous harvest units. This situation obviously arises when a harvest road crosses through a harvest block – in effect the block is sliced into two non-contiguous pieces. However, the situation may also arise when two or more proximate harvest units are scheduled for harvest in the same period, but their combined areas do not exceed the maximum opening size limit. So long as the adjacency restrictions for the combined set of harvest units is met, they may be treated as a single opening.

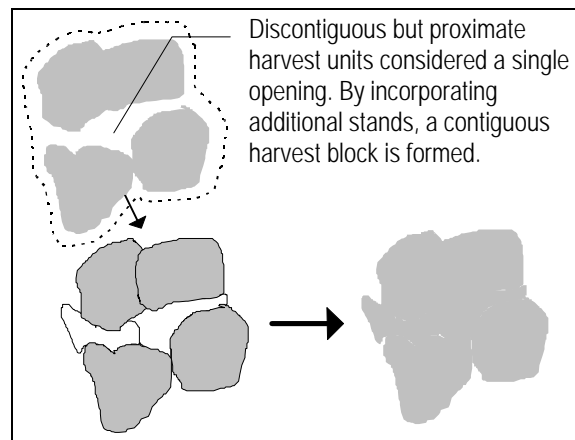


Figure 7. Stand aggregation to consolidate discontinuous harvest units (*Stanley* Phase 4).

There are some drawbacks to doing this: the area tied up in the green-up corridor is obviously greater than would be the case for a single contiguous harvest block. Accessibility to the individual harvest units may also be problematic and could be ameliorated by consolidating them. Phase 4 attempts to do this by incorporating new stands such that the fragments

become part of one contiguous block (see Figure 7). As in Phase 2, checks are made to ensure that adjacency restrictions and opening size limits are not violated.

The last phase of the *Stanley* algorithm simply attempts to balance output flows by sacrificing blocks. When the objective function value is higher than the current best solution but the flow fluctuations exceed the specified tolerance, *Stanley* will begin deallocating blocks until the flow fluctuations come within range. Since sacrificing blocks necessarily reduces the objective function, it may turn out that the objective function value will be reduced below that of the current best solution.

Data Requirements

Because the capabilities of *Stanley* now include those previously performed by *Crystal*, the program requires more input data than the previous version. Structures and formats of the input data files are given in the appendix.

Activity List

The activity list details the harvest actions that are to be blocked. Since *Woodstock* generates this file automatically in DBF format, it was a logical choice to use this file directly. An equivalent file can be generated by users who do not use *Woodstock* by maintaining the table structure and filling in the appropriate fields corresponding to their strategic harvest schedule. *Stanley* takes advantage of reduced cost information arising from LP solutions to select (where possible) timing choice alternatives that have the least impact on objective function value.

Stand Operability List

Although *Stanley* is primarily concerned with allocating harvest periods to stands, it must be cognizant of harvest operability limits that are specific to different stand types. Since the operability guidelines are inherent to the strategic harvest schedule, it makes sense to use the strategic harvest scheduling model solution file as the source of this information.

Stand (Polygon) Description List

The stand (polygon) description list is usually derived from a GIS attribute file and contains a unique identification label for each stand, the landscape thematic attributes that identify the development type each stand belongs to, the current age in periods, and the area of the stand. An empty field for storing the

harvest period that *Stanley* assigns to each stand is also required. Other fields may be included in the file for reference by the user, but they are not explicitly required by *Stanley*.

Stand (Polygon) Adjacency List

The stand (polygon) description list contains double entry list of adjacent or proximate stands identified by their unique identification labels. There are three fields in the table: the first two contain the ID labels for each pair of adjacent or proximate stands and the third field is simply a logical field that differentiates adjacent stands from proximate ones. A proximate stand is one that is not physically contiguous to another but has a boundary point that is within a given distance of that stands nearest boundary. Remsoft has developed a utility program to generate the adjacency list from ESRI ArcInfo and ArcView data files, but this list may be produced using for any geographic information system.

Stand (Polygon) Extents List

The stand (polygon) extents list is optional and contains the unique stand identification label and northeast and southwest x-y coordinates defining the bounding box or extent for each stand. This information is usually available via a GIS query and it is used by *Stanley* to control block shape.

Case studies

The first set of case studies are designed to illustrate the enhanced blocking features of *Stanley*. A uniform forest composed of identical cells is used to represent scenarios where the *Stanley* algorithms have been modified to achieve better performance in blocking. The second set of case studies illustrates how *Stanley* can address non-linear flow constraints and the third set demonstrates multiple harvest actions. The last case study represents a real forest under realistic flow and adjacency constraints.

Case 1 - Adjacency versus proximity

A uniform, square forest was subdivided into a 20 x 20 grid and *Stanley* was applied to maximize even flow harvest for four planning periods. The minimum harvest block size was one cell and no maximum size limit was established. A one period harvest delay constraint was applied, such that a block adjacent to a block harvested in period one could be harvested no earlier than period three. *Stanley* tries to make blocks as large as possible while maintaining even flow from period to period.

Figure 8.
Stanley case
study 1:
proximate and
corner point
adjacent cells
omitted from
adjacency
list.

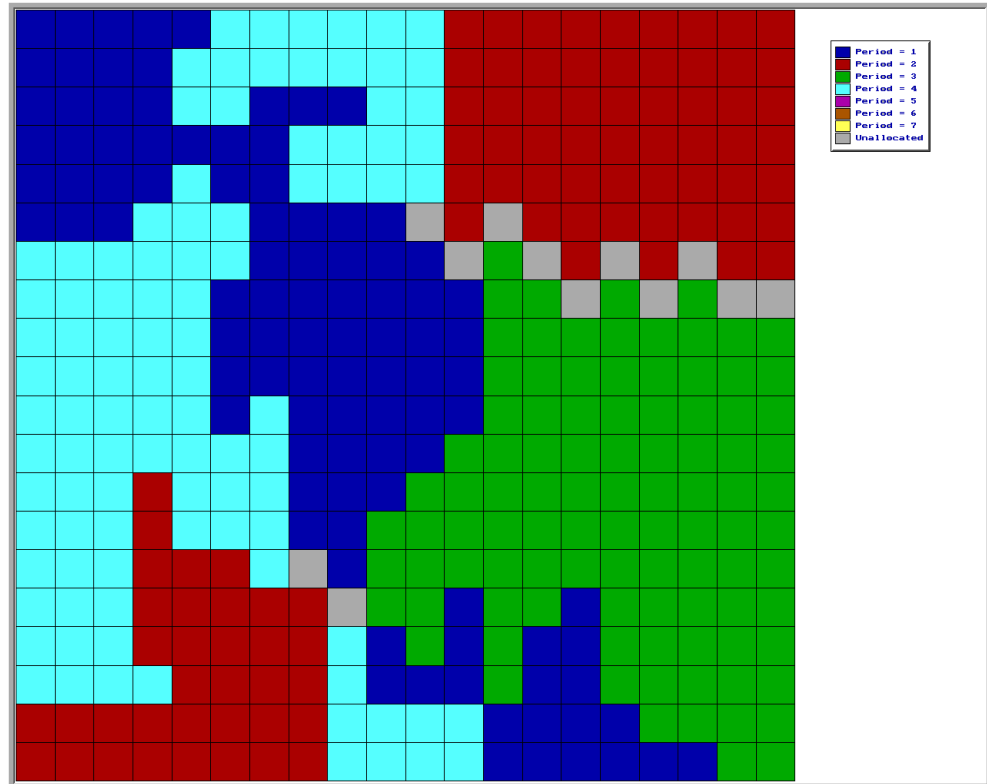
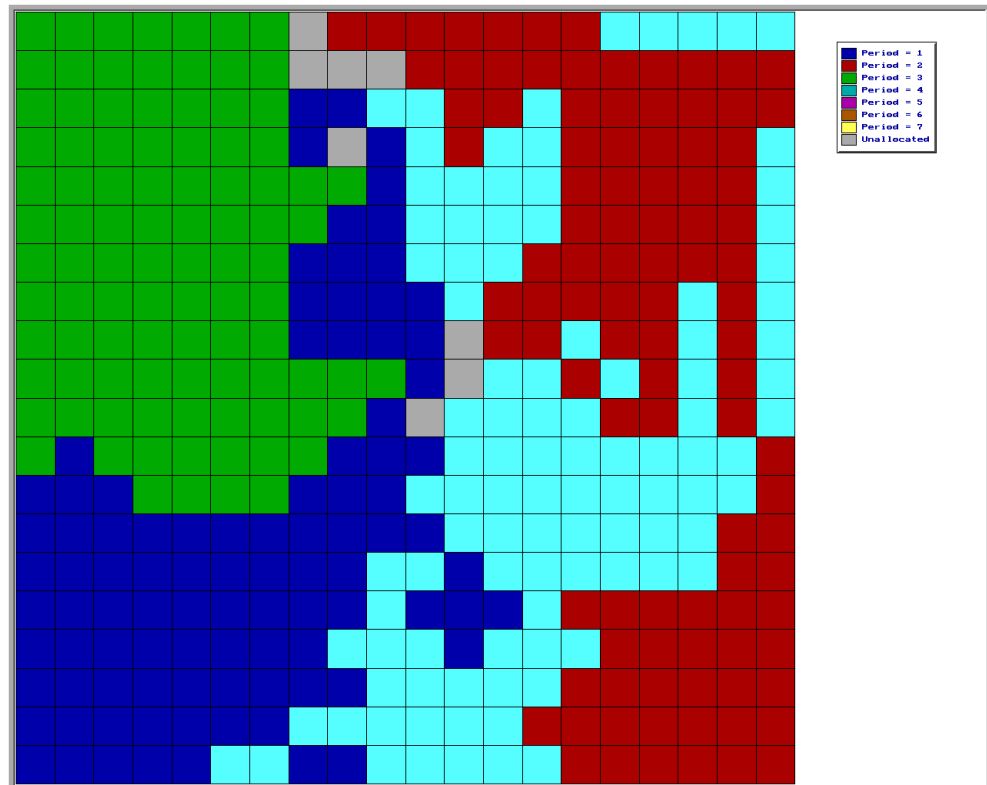


Figure 9.
Stanley case
study 2:
proximate and
corner point
adjacent cells
included in
adjacency
list.



In Figure 8, you can readily see that *Stanley* was quite successful in allocating the harvest to meet the green-up delay requirements. Since corner point adjacencies were not included in the adjacency list, *Stanley* left cells unharvested between the red block in the upper right corner and green block below it, to act as buffers between the two harvest blocks. Similarly, the blue J-shaped block located in the lower middle of the forest is considered a separate block from the nearby larger blue block. However, both of these instances clearly violate the intent of the adjacency restrictions.

When the corner point adjacencies were included in the adjacency list *Stanley* yielded a very different block pattern, as illustrated in Figure 9. Notice that in this block configuration there are no instances where the adjacency restrictions are violated – cells are either left unharvested or there is a minimum one period delay for the intervening cells between blocks. However, the harvest blocks themselves are less desirable than those in Figure 8.

For example, all of the blue cells constitute a single large harvest block, but there are isolated pieces that are problematic from an operational standpoint. Any attempt to connect these small islands with the main portion of the block could mean completely severing another island from its parent block – for example, the blue and cyan blocks on the lower boundary of the forest. In other cases, connecting these islands to their parent blocks with corridors would result in adjacency violations as well.

Stanley avoids the problems associated with strict adjacency and proximity by employing only adjacency relationships when creating harvest units. Later, when it is scheduling harvest units, *Stanley* uses both the adjacency and proximity relationships to ensure that spatial constraints are properly met.

Case 1 - Stand extents for shape control

The *Stanley* blocking algorithm is designed to select stands which correspond to stand types selected for harvest in a strategic harvest scheduling model. If more than one polygon exactly matches the stand types being sought for blocking, the algorithm selects the first polygon it finds. A problem arises if the stands in the forest are subdivided, either by a systematic overlay grid or if they are derived from raster GIS format. Because there are multiple neighboring polygons or cells to choose from, the algorithm tends to continue searching for adjacent neighbors in the same direction.

To illustrate the problem of blocking without shape controls, we applied *Stanley* to a completely uniform

5000 ha forest that was scheduled for harvest over 4 periods under even-flow constraints. Since all stands are identical, any random starting point can yield a feasible harvest unit as long as its neighbors have not already been allocated. In order to simulate realistic planning requirements used in British Columbia and elsewhere, we applied minimum and maximum harvest block sizes of 20 and 120 ha respectively along with a 1 period green-up delay. In Figure 10, it is readily apparent that the blocks are oriented in specific orientation arising from the numbering scheme used to identify polygons. *Stanley* uses an indexing scheme to rapidly locate feasible polygons but any polygon identification scheme that is systematic will result in some form of systematic block configuration.

By utilizing the extents information available in a GIS database, *Stanley* tries to create harvest units that are more reasonable. A polygon *extent* is simply an x-y coordinate pair that defines a rectangle that completely contains the polygon. As polygons are incorporated into a harvest unit, an extent is calculated for the entire harvest unit. When considering equally feasible polygons for inclusion, *Stanley* compares the impact of adding each candidate and selects the polygon which yields the more regular shape. *Stanley* was again applied to the harvest scheduling problem described earlier, except that polygon extents were utilized to provide shape controls.

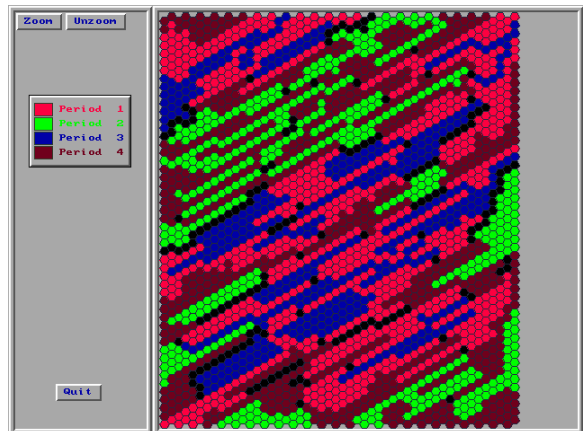


Figure 10. *Stanley* solution without block shape controls applied. Cells shaded black are left unharvested.

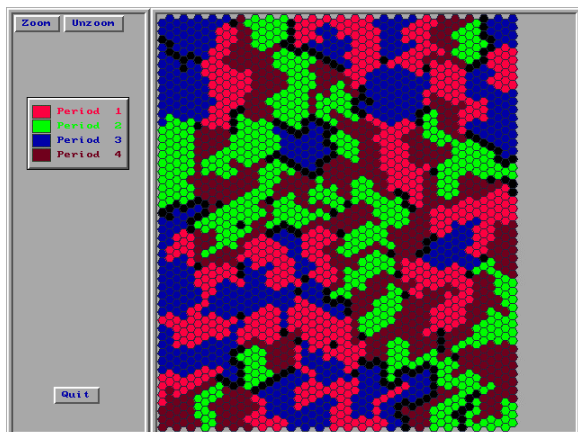


Figure 11. *Stanley solution with block shape controls applied. Cells shaded black are left unharvested.*

In Figure 11, the configuration of harvest blocks is much better – there are relatively few *tentacles* and harvesting is clustered. Although it is not readily apparent from the maps, the harvest flow from the schedule with shape controls is much better than the schedule without shape controls. The striated harvest block pattern of Figure 10 illustrates a greedy algorithm in terms of spatial flexibility.

Because the first blocks allocated are largely unconstrained by previous allocations, they tend to be linear. This linearity makes later allocations more difficult because perpendicular arrangements become increasingly difficult even though they may be exactly what is needed to address adjacency constraints. Eventually, there are no options left to create a feasible harvest unit and the forest area is left unharvested. Thus, the average harvest level is reduced and the harvest flows are more erratic.

Case 2 - Nonlinear output flows

While the flow of products from a strategic model may be strictly even, it is usually expected that some variation in periodic harvest will arise from blocking. A question arises, however, as to how much variation is to be allowed, and on what basis is the variation calculated. The simplest type of constraint simply requires that harvest be within 5% of a fixed amount, usually derived from the strategic harvest level. The difficulty with this approach is that the specified target harvest level may not be feasible. A more complicated constraint requires that harvest flow fluctuations be less than 5% period to period. Although this avoids the infeasibility problem of the first approach, there is no guarantee that harvests will not continually decline over time. A third alternative would be to require that periodic harvests fall within

5% of the average harvest for all periods. Like the second approach, there is no guarantee that harvests will not continually decline, but the decline will be limited to at most 10% over the entire planning horizon.

When harvest levels are no longer required to be even, the issue of flow fluctuation becomes less clear. For example, how much variation does one allow when strategic harvest flows decline each period by up to 10% and a 5% tolerance on flow fluctuations is specified? To illustrate the problem, suppose *Stanley* generated a solution with output levels of 104 and 90 in the first 2 periods to fit a strategic schedule that called for output levels of 100 and 90. One interpretation might be that a 5% tolerance would allow declines of up to 15% and as little as 5%, and therefore the solution is acceptable (14% decline). Another interpretation might be that the expected decline is 10 units and a 5% tolerance permits declines ranging from 9.5 to 10.5 units, and therefore the solution is not acceptable. Currently *Stanley* interprets the 5% tolerance in terms of the two periods which deviate most from the strategic harvest schedule (i.e., if one period is overallocated by 4% and another period is underallocated by 1%, the 5% tolerance is just met).

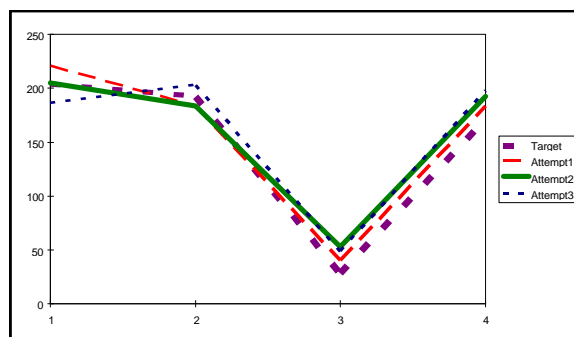


Figure 12. *Stanley harvest schedule output levels compared to strategic output levels.*

Another issue to be resolved is how to compare solutions and choose the one that is best. Based on an even-flow strategic harvest schedule, maximizing the average volume over all periods may be a reasonable objective but how much weight do you place on periodic fluctuations in harvest flows? Furthermore, if the strategic schedule is not even-flow or worse, non-linear, an average harvest level over all periods makes little sense. *Stanley* uses a maximization objective and selects the solution which produces the highest total output over the planning horizon. In Figure 12, 3 block solutions are compared to the projected output levels from the strategic harvest schedule – *Stanley* would retain the solution

corresponding to the red dashed line because the sum of outputs over 4 periods is highest.

Case 3 - Previously allocated blocks

In designing *Stanley*, two types of preallocated blocks needed to be considered. The more important of these are blocks corresponding to harvests that have taken place recently, and therefore green-up restrictions in their proximity are still in place. *Stanley* can allocate stands to harvest units within the corridors, but it must not schedule these blocks for harvest during the green-up periods.

The less important type of preallocated block represents stands that were scheduled for harvest in the near future based on operational considerations, but as yet they have not been harvested. For these blocks *Stanley* may incorporate additional stands to alleviate harvest flow or adjacency problems but it cannot choose to leave preallocated blocks unharvested.

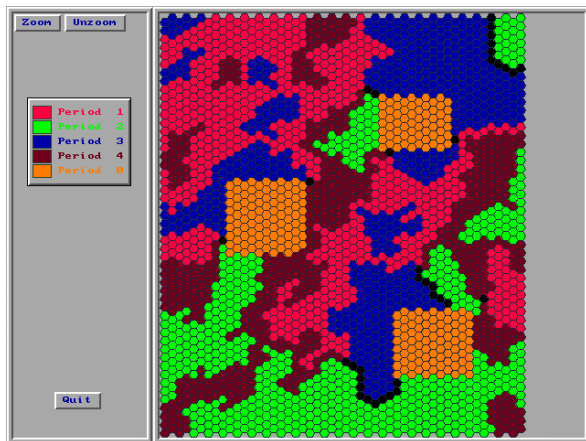


Figure 13. *Stanley* solution, with pre-existing harvest blocks with associated green-up delays.

Figure 13 depicts a planning situation where previously harvested blocks are present and green-up delays are still required within a given distance of their boundaries. In this simple example, the minimum distance is equal to the length of an edge on each hexagon. For *Stanley* to properly account for the buffer distance, the proximity information is crucial. Proximity relationships should be calculated for the maximum distance anticipated for green-up buffers. *Stanley* will work without differentiating proximity from adjacency, but the results may be disappointing, as was illustrated in the first case study.

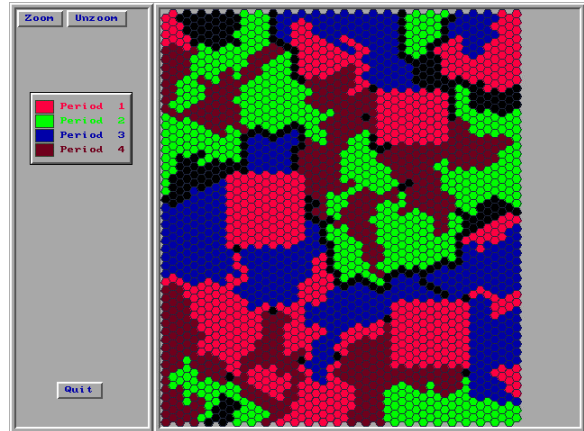


Figure 14. *Stanley* solution with 3 first period blocks preallocated (red rectangles). Cells shaded black are left unharvested.

Figure 14 illustrates a *Stanley* block schedule where 3 blocks were preallocated to period 1. The scheduling algorithm was successful in meeting adjacency constraints by scheduling adjacent blocks in other periods. Block augmentation was turned off, but *Stanley* could also have mitigated adjacency difficulties by augmenting the preallocated blocks with additional stands, if the option had been used.

Case 4 - Multiple harvest actions

Stanley is able to allocate and schedule harvest activities that are *compatible*; because they are subject to the same opening size and adjacency restrictions it would be possible to combine two or more of these activities within a single harvest block. If harvest activities are not compatible, they need to be allocated in separate *Stanley* runs or through manual procedures. In the future, the ability to simultaneously allocate non-compatible activities in a single *Stanley* run will be implemented.

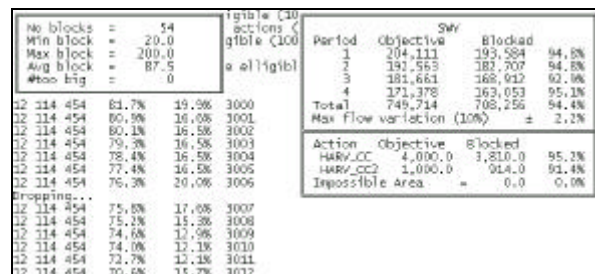


Figure 15. Screen capture of a *Stanley* run, with two compatible harvest actions to be allocated and scheduled.

Stanley can differentiate harvest actions that contribute to the objective function value and those that do not. It will take advantage of these actions in order to make feasible blocks, but any stands

allocated to these actions will not affect the objective function value. Figure 15 is a screen capture of *Stanley* as it is running and the relative success in blocking each harvest action is reported.

In Figure 16, the solution to the aforementioned *Stanley* run with two compatible harvest actions is depicted. While colors correspond to particular periods, the darker shades represent the first harvest action and the lighter shades, the second harvest action. Note that within harvest blocks, *Stanley* does not attempt to cluster the two actions.

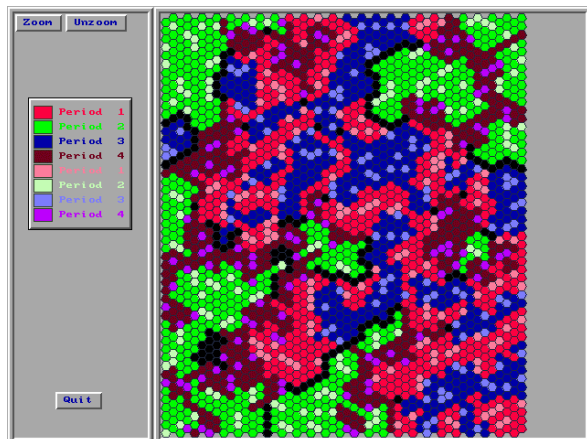


Figure 16. *Stanley* solution for two compatible harvest actions. Stands shaded black are left unharvested.

Case 5 - Large scale forest planning

In many respects, the case studies presented so far represent worst case scenarios. Actual forests are not composed of a single stand with uniform species and age characteristics. In reality, forests will be naturally subdivided on species, site quality, riparian zones and numerous other features. While some of these boundaries represent obstacles to blocking and scheduling, others provide natural breaks that facilitate the process. The purpose of the artificial forests was to detect systematic errors in the *Stanley* algorithms that would not be easily detected in a real forest data set. Similarly, the purpose of a real forest test data set is to observe how well the algorithms function under realistic operating conditions.

The last case study is modeled loosely on a paper by Nelson and Hackett (1992) that dealt with the problem of scheduling blocks under adjacency and opening size restrictions in British Columbia. On a 4000 ha forest composed of 7291 stands (average 5.4 ha), we applied opening size restrictions of 10 ha minimum and 100 ha maximum, an adjacency corridor of no less than 300 m and green-up delays of

3 periods (15 years). The strategic harvest schedule had a 20 period planning horizon and we scheduled the first 10 periods using *Stanley*.

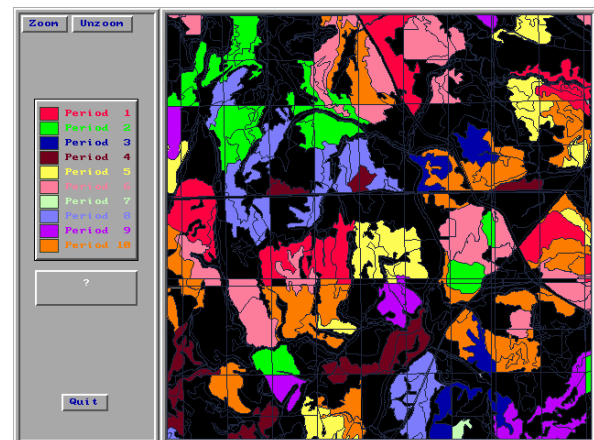


Figure 17. *Stanley* solution for a real forest using 300 m adjacency corridors, 3 period green-up delays and maximum opening sizes of 100 ha. Stands shaded black are left unharvested.

The combination of relatively long green-up delays and large adjacency corridors exacted a heavy penalty on the objective function value, and this was expected (about 35-40% reduction from non-spatial strategic harvest levels). Given that the forest itself was rather small, there is not the range of age classes available that would be characteristic of a commercial forest land base and this lack of flexibility severely limits *Stanley's* ability to find feasible solutions. Nevertheless, the blocked solution however looks entirely reasonable, with an average block size of just over 40 ha (see Figure 17).

Literature cited

- Baskent, E.Z. Spatial wood supply modelling: concept and practice. M.Sc.F. thesis, Faculty of Forestry, University of New Brunswick, Fredericton.
- Cogswell, A. 1995. Evaluating the spatial feasibility of future timber harvest allocations using a repeated planning process. B.Sc.F. thesis, Faculty of Forestry, University of New Brunswick, Fredericton.
- Dallain, P.L. 1989. An operational, spatially feasible harvest scheduling model. M.Sc.F. thesis, Faculty of Forestry, University of New Brunswick, Fredericton.
- Jamnick, M.S., and Walters, K.R. 1993. Spatial and temporal allocation of stratum-based harvest schedules. *Can. J. For. Res.* **23**: 402-413.

Jones, J.G., Meneghin, B.J. and Kirby, M.W. 1991. Formulating adjacency constraints in linear optimization models for scheduling projects in tactical planning. *For. Sci.* **37**: 1283-1297.

Lockwood, C. and Moore, T. 1993. Harvest scheduling with spatial constraints: a simulated annealing approach. *Can. J. For. Res.* **23**: 468-478.

Meneghin, B.J., Kirby, M.W. and Jones, J.G. 1988. An algorithm for writing adjacency constraints efficiently in linear programming models. **IN** The 1988 Symposium on Systems Analysis in Forest Resources, 29 March - 1 April 1988. Asilomar, CA. *Edited by* B. Kent and L.S. Davis. USDA For. Serv. Rocky Mt. For. Range Exp. Stn. Gen. Tech. Rep. RM-161. pp 46-53.

Nelson, J., Brodie, J.D., and Sessions, J. 1991. Integrating short-term, area-based logging plans with long-term harvest schedules. *For. Sci.* **37**: 101-122.

Nelson, J. and Hackett, J.S. 1992. An economic analysis of timber harvesting regulations in the Tsitika Valley.

Remsoft Inc. 1994. An assessment of tools for strategic and tactical forest management planning. New Brunswick Dept. Nat. Res. Energy, Fredericton, NB.

Torres-Rojo, J. and Brodie, J.D. 1990. Adjacency constraints in harvest scheduling: an aggregation heuristic. *Can. J. For. Res.* **20**: 978-986.

Walters, K.R. 1991. Spatial and temporal allocation of strata-based harvest schedules. M.Sc.F. thesis, Faculty of Forestry, University of New Brunswick, Fredericton.

Walters, K.R. and Feunekes, U. 1994. A hierarchical approach to spatial planning: a report card. **IN** Proceedings of the GIS'94 Symposium, Vancouver, BC. February, 1994.

Yoshimoto, A. and Brodie, J.D. 1994. Comparative analysis of algorithms to generate adjacency constraints. *Can. J. For. Res.* **24**: 1277-1288.

Appendix - File formats

For the most part, *Stanley* input data is stored in DBF files, but some information is stored in ASCII text format. There are no proprietary data formats.

Structure for global stand database:

E:\STANLEY\DEM01\F0REST.DBF				
Number of data records:		2500		
Date of last update		: 03/28/96		
Field Index	Field Name	Type	Width	Dec
1	AREA	Numeric	13	3
2	AGE	Numeric	3	
3	REMSOFT_ID	Character	8	
4	CUT_PERIOD	Numeric	3	
5	PREBLOCK	Character	1	
6	TH1	Character	6	
7	TH2	Character	2	
8	TH3	Character	2	
9	TH4	Character	4	
** Total **			43	

Structure for strategic harvest schedule choices database:

E:\STANLEY\DEM01\M0DEL.DBF				
Number of data records:		36		
Date of last update		: 03/27/96		
Field Index	Field Name	Type	Width	Dec
1	VARIABLE	Character	8	
2	L1	Character	6	
3	L2	Character	1	
4	L3	Character	2	
5	L4	Character	4	
6	DEATH_AGE	Numeric	3	
7	BIRTH_AGE	Numeric	3	
8	BORN	Numeric	3	
9	DIED	Numeric	3	
10	ACTION	Numeric	3	
11	SOLN_VALUE	Numeric	12	4
12	SWV	Numeric	6	1
** Total **			55	

Structure for stand adjacency database:

E:\STANLEY\DEM01\STANDADJ.DBF				
Number of data records:		14602		
Date of last update		: 03/28/96		
Field Index	Field Name	Type	Width	Dec
1	REMSOFT_ID	Character	11	
2	REMSOFT_AD	Character	11	
3	PROXIMAL	Character	1	
** Total **			24	

Structure for stand extents database:

E:\STANLEY\DEMO1\EXTENTS.DBF				
Number of data records: 2500				
Date of last update : 02/13/96				
Field	Field Name	Type	Width	Dec
Index				
1	REMSOFT_ID	Character	8	
2	MINX	Numeric	15	5
3	MINY	Numeric	15	5
4	MAXX	Numeric	15	5
5	MAXY	Numeric	15	5
**	Total	**	69	